

**NETWORK ARCHITECTURE DEFENSE: HOLISTIC SECURITY  
PATTERN-BASED MODEL**

**Castro Auma Yoga**

A Thesis Submitted to The Board of Postgraduate Studies in Partial Fulfillment of The Requirement for The Award of The Degree of Doctor of Philosophy in Information Technology Security and Audit of Jaramogi Oginga Odinga University of Science and Technology

©2024

# DECLARATION AND APPROVAL

## Declaration

This is my own work and has not been presented for any award in any other university or institution

Signature: \_\_\_\_\_

Castro Yoga

**I162/4608/2014**

\_\_\_\_\_

Date

## Approval

This proposal/thesis has been submitted with our approval as the university supervisors.

Signature: \_\_\_\_\_

**Prof. Anthony J. Rodrigues,**

Department of Computer Science and Software Engineering.

School of Informatics and Innovative Systems.

Jaramogi Oginga Odinga University of Science and Technology

\_\_\_\_\_

Date

Signature: \_\_\_\_\_

**Prof. Silvance Abeka,**

Department of Information Systems Technology.

School of Informatics and Innovative Systems.

Jaramogi Oginga Odinga University of Science and Technology

\_\_\_\_\_

Date

## **ACKNOWLEDGEMENT**

I would like to express my deepest gratitude to all those who have contributed to the completion of this thesis. This journey would not have been possible without the support and encouragement of numerous individuals.

First and foremost, I am profoundly thankful to my supervisor, Prof. Anthony Joachim Rodrigues and Prof. Silvanice Onyango Abeka, whose guidance and expertise have been invaluable throughout the entire research process. Your insightful feedback and unwavering commitment to my academic development have been instrumental in shaping this work.

I am grateful to my family for their unconditional love and encouragement. Your unwavering support has been a constant source of motivation, and I am truly blessed to have you by my side.

Special thanks go to my friends and colleagues who provided valuable insights and a supportive environment. Your camaraderie has made the challenges more manageable and the successes more meaningful.

This thesis is a testament to the collective effort of those mentioned above, and I am sincerely thankful for the impact each has had on this academic endeavor.

## **DEDICATION**

Dad, Mom, Dave, Jerry & Rodger thanks for the encouragement and support I hope I have  
made you proud

Caro, I do not take for granted your sacrifice for all those years

Imani, Manuella, CJ, & Adrian you guys are the reason for all this

Angie, I wish you were here to see this.

## ABSTRACT

Network security experts face numerous challenges in protecting networks despite implementing defense strategies. The complexity of networks, coupled with a scattered approach to security implementation, adds to the difficulties. Currently, different security solutions employ distinct mechanisms without a cohesive approach to the entire system. Although similar problems exist at each level of security, a holistic strategy is lacking, resulting in different models being applied in various parts of the network architecture. To effectively secure a network, a coordinated and holistic approach is essential. The study's primary goal was to develop a holistic security pattern-based model for defending network architecture. To achieve this the study looked at the techniques and threats employed in attacking the network architecture, assessed the models, frameworks and artifacts that guide in the design and development of a secure network architecture. Overall, the study was guided by pattern theory, the constructs employed in the development of the model included the OSI network architecture model, the cisco three-layer hierarchical model, CAPEC attack pattern Repository, STRIDE threat Model and Risk Management Framework. The study adopted Simulation research design approach to design and conduct experiments to obtain results. To test the model the study utilized a secondary dataset UNSW-NB15 which was subjected to Kaggle machine learning platform. For ease of testing, the model was split into three stages with their respective input, process and output component, with each output serving as an input to the subsequent stage. The first stage was to determine the attacks per surface of the network architecture this involved classifying and clustering attacks according to the layers, for classification a stacking ensemble approach composed of select KBest feature selection algorithm, a KNeighbors, RandomForest and GaussianNB classifiers and Logistic regression Meta learner was utilized, for clustering KMeans clustering algorithm was utilized. The second stage was to identify relevant attacks while third was to generate defense patterns. The findings reveal that a significant percentage of attacks targeted the Host layer (50.5%), followed by the User layer (30.5%) and the Media layer (19%). The distribution of attacks is categorized by types, with exploits constituting the majority (48%), followed by generic attacks (22.7%), fuzzers (12.2%), reconnaissance (7.69%), DoS (Denial of Service) (5.02%), backdoor (3.01%), analysis (0.6%), shellcode (0.33%), and worms (0.11%). Additionally, the study identified and evaluated two attack patterns (worms and backdoors) not present in the CAPEC repository. The evaluation was based on their forces and the STRIDE model. Overall, the research emphasizes the importance of a holistic approach to network security and presents a model that integrates various frameworks and constructs to enhance defense against cyber threats.

# TABLE OF CONTENTS

DECLARATION.....	i
ACKNOWLEDGEMENT .....	ii
DEDICATION.....	iii
ABSTRACT .....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES .....	xi
LIST OF TABLES .....	xiii
LIST OF ABBREVIATIONS AND ACRONYMS. ....	xv
CHAPTER ONE .....	1
INTRODUCTION.....	1
1.1. Background Information.....	1
1.2. Statement of the problem. ....	6
1.3. Objectives of the Study .....	7
1.3.1. Main Objective of the Study.....	7
1.3.2. Specific Objectives. ....	7
1.4. Significance of the Study. ....	7
1.5. Scope of the Study .....	8
1.6. Assumption of The Study .....	8
1.7. Rationale of the Study.....	8
CHAPTER TWO .....	10
LITERATURE REVIEW.....	10
2.1. The History of Patterns .....	10
2.1.1. The Notion of Patterns.....	12
2.1.2. Describing and documenting patterns.....	14
2.1.2.1. GoF format .....	14
2.1.2.2. Alexandrian format.....	15
2.1.2.3. POSA format .....	16
2.1.2.4. Coplien format.....	17
2.1.3. Elements of patterns .....	18
2.1.4 Challenges for Creation of Patterns .....	19
2.1.5. Benefits Accrued from Development of Patterns .....	21
2.1.6. Inadequacies of Patterns.....	22

<b>2.2. Design Patterns.....</b>	<b>24</b>
<b>2.3. Anti-patterns.....</b>	<b>26</b>
<b>2.4. Attacks on Networks and Havoc Caused.....</b>	<b>26</b>
<b>2.4.1. Techniques Employed in Attacking Networks .....</b>	<b>29</b>
<b>2.4.2. Havoc of Network Attacks.....</b>	<b>30</b>
<b>2.5. The OSI Model and Its Roles in Securing Networks .....</b>	<b>31</b>
<b>2.5.1. The OSI Model.....</b>	<b>32</b>
<b>2.5.2. Necessity of Having Security at Each Layer of the OSI Model.....</b>	<b>36</b>
<b>2.5.3. OSI Layers Threats.....</b>	<b>36</b>
<b>2.5.3.1 Physical Layer Security threats .....</b>	<b>36</b>
<b>2.5.3.2. Data Link Layer Security Threats (Switch Security) .....</b>	<b>37</b>
<b>2.5.3.3. Network Layer Security Threats (Router Security) .....</b>	<b>38</b>
<b>2.5.3.4. Transport Layer Threats.....</b>	<b>39</b>
<b>2.5.3.5. Session Layer Threats.....</b>	<b>40</b>
<b>2.5.3.6. Presentation Layer Threats.....</b>	<b>41</b>
<b>2.5.3.7. Application Layer Threats.....</b>	<b>41</b>
<b>2.6. Network Security Architecture Role in Securing Networks.....</b>	<b>43</b>
<b>2.7. Network Security Design.....</b>	<b>44</b>
<b>2.8. Frameworks Guiding Security of Networks.....</b>	<b>46</b>
<b>2.8.1. NIST Security Framework .....</b>	<b>46</b>
<b>2.8.2. COBIT Security Framework .....</b>	<b>46</b>
<b>2.8.3. ISO/IEC Standards framework .....</b>	<b>46</b>
<b>2.8.4. HITRUST Cybersecurity Framework (CSF).....</b>	<b>47</b>
<b>2.8.5. Internet of Things (IoT) Cybersecurity Alliance (IOTCA).....</b>	<b>47</b>
<b>2.8.6. MITRE ATT&amp;CK.....</b>	<b>48</b>
<b>2.9. Threat Model and Threat Modeling.....</b>	<b>48</b>
<b>2.9.1. Threat Modeling Approaches.....</b>	<b>50</b>
<b>2.9.1.1. LINDDUN .....</b>	<b>51</b>
<b>2.9.1.2. OCTAVE.....</b>	<b>52</b>
<b>2.9.1.3. STRIDE.....</b>	<b>54</b>
<b>2.9.1.4. DREAD.....</b>	<b>56</b>
<b>2.9.1.5. PASTA.....</b>	<b>57</b>
<b>2.9.1.6. CVSS.....</b>	<b>59</b>

2.9.1.7. <b>TRIKE</b> .....	60
2.9.1.8. <b>Attack Trees</b> .....	62
<b>2.10. Security Risk Assessment and Network Security</b> .....	63
<b>CHAPTER 3</b> .....	66
<b>METHODOLOGY</b> .....	66
<b>3.1. Research Design</b> .....	66
<b>3.2. Phase 1: Model Development</b> .....	66
3.2.1. <b>Literature Review</b> .....	66
3.2.2. <b>Conceptualization of holistic security pattern-based model</b> .....	66
<b>3.3 Phase 2: Simulation</b> .....	66
3.3.1 <b>Selection of Simulation Tools</b> .....	66
3.3.2 <b>Scenario Design</b> .....	66
3.3.3 <b>Implementation of the Model</b> .....	67
3.3.4 <b>Metrics Definition</b> .....	67
<b>3.4 Ethical Considerations</b> .....	67
<b>3.5. Dataset</b> .....	67
<b>3.6. The Study Test Environment</b> .....	69
3.6.1. <b>Kaggle Test Bed</b> .....	69
3.6.2. <b>Python</b> .....	69
<b>3.7. Performance Metrics</b> .....	69
<b>CHAPTER FOUR</b> .....	71
<b>MODEL FORMULATION</b> .....	71
<b>4.1. Introduction</b> .....	71
<b>4.2. Extending the OSI model</b> .....	72
<b>4.3. Model Development</b> .....	75
4.3.1. <b>Attack Context Checking</b> .....	75
4.3.1.1. <b>Three Layer Network Security Domain (TNLSD) Component</b> .....	75
4.3.1.2. <b>Anti-Goal Identification</b> .....	76
4.3.2. <b>Attack Surface Identification</b> .....	78
4.3.3. <b>CAPEC Repository</b> .....	80
4.3.3.1. <b>Attack Patterns</b> .....	81
4.3.3.2. <b>Retrieving Relevant Attack Patterns Component</b> .....	83
4.3.4. <b>Risk Assessment Component</b> .....	86



4.3.5.	Network Architecture Security Pattern Based Model .....	90
CHAPTER 5 .....		91
MODEL TESTING AND RESULTS .....		91
5.1.	Model Testing .....	91
5.1.1.	First Stage of Model .....	91
5.1.2.	The Second Stage of the Model. ....	92
5.1.3.	Third Stage of the Model. ....	93
5.2.	Study Data Set -UNSW-NB_15 .....	94
5.2.1.	Dataset Description .....	94
5.2.2.	UNSW-NB15 Features Categories.....	96
5.2.3.	UNSW-NB15 Data Types.....	97
5.2.4.	UNSW-NB15 Attack Type Categories.....	97
5.3.	Machine Learning Testbed.....	98
5.3.1.	Importation of Machine Learning Libraries and Modules .....	98
5.3.2.	Data Preprocessing.....	99
5.3.3.	Importation of Dataset.....	99
5.3.4.	Renaming of Columns Headers. ....	100
5.3.5.	Boolean Filtering of Malicious Logs .....	100
5.3.6.	Merging the Dataset.....	102
5.4.	Analysis of the Malicious Combined Dataset .....	102
5.4.1.	Malicious Count Within the Dataset .....	102
5.4.2.	Identification of Missing Values.....	103
5.4.3.	Protocol Distribution in the Malicious Traffic .....	104
5.4.3.1.	Top Three Malicious Protocols .....	104
5.4.4.	Attack Distribution. ....	108
5.4.5.	Targeted Ports Per Attack.....	111
5.4.6.	Main Dependency Protocols Timings Per State .....	112
5.4.7.	Mean Of Time Distribution in Malicious Packets.....	113
5.4.8.	Attack Category Time Duration in The Malicious Dataset.....	113
5.4.9.	Attack Category Time Duration For TCP, UDP, UNAS And Others.....	113
5.4.10.	Mean Duration of Attacks Per Protocol.....	114
5.5.	Data Cleaning .....	115
5.5.1.	Identification And Filtering of Missing Values. ....	115

5.5.2.	Filling of the Missing Values .....	116
5.5.3.	Filling Missing Values with The Median.....	116
5.5.4.	Forward Filling of Missing Values .....	117
5.5.5.	Backward Filling of Missing Values.....	117
5.5.6.	Filling Of Missing Values in The Mal Dataset with Created Values .....	117
5.6.	Machine Learning algorithms.....	118
5.6.1.	Feature Selection -SelectKBest Algorithm.....	118
5.6.2.	Base Learners -KNeighbors, Random Forest, GaussianNB.....	118
5.6.3.	Meta-Learner (Logistic Regression).....	119
5.6.4.	K-means Clustering .....	119
5.6.5.	Feature Scaling .....	119
5.7.	Ensemble Technique Approach.....	120
5.7.1.	Stacking Ensemble .....	121
5.7.2.	Feature Selection .....	121
5.7.3.	Base Learners .....	122
5.7.4.	Meta Learner .....	123
5.7.5.	Clustering Algorithm Based on KMeans .....	124
5.7.6.	Performance Metric .....	126
5.8.	Feature Engineering and Feature Selection.....	126
5.8.1.	Training and Testing Splitting .....	126
5.8.2.	The Select KBest Feature Selection Algorithm .....	127
5.8.3.	Importation of the Machine Learning Algorithms. ....	129
5.8.4.	Preparation for the Machine Learning Algorithms.....	129
5.8.5.	Analysis of Attacks to Their Surfaces. ....	131
5.8.6.	Visualization of Attacks to Their Surface.....	132
5.9.	Identification and Retrieval of Stride Based Attacks from CAPEC Repository.....	133
5.9.1.	Filtering Of UNSW-NB_15 Attacks from CAPEC. ....	133
5.9.2.	Merging of Attacks In UNSW-NB_15 Found In CAPEC.....	136
5.9.3.	Mapping UNSW-NB_15 Attacks Identified CAPEC To STRIDE Threat Model	137
5.10.	Mapping Process of CAPEC Attacks to STRIDE.....	137
5.11.	Risk Analysis of the Filtered Attacks. ....	139
5.11.1.	Quantitative Risk Analysis.....	140
5.11.2.	Qualitative Risk Analysis.....	143

5.12. Defense Mechanism for the Attacks .....	144
<b>CHAPTER 6</b> .....	148
<b>DISCUSSIONS AND FINDINGS</b> .....	148
6.1. Summary of Findings.....	148
6.2. Generation of Worms and Backdoor Patterns for CAPEC Repository.....	150
6.2.1. General Worm Pattern .....	151
6.2.2. General Backdoor Attack Pattern .....	155
6.3. Qualitative Analysis of Worms and Backdoor Patterns.....	158
6.3.1. Evaluating Worm and Backdoor Patterns on The Basis Of Their Forces.....	159
6.3.2. Evaluation of Worm and Backdoor Patterns Based On STRIDE.....	160
<b>CHAPTER 7</b> .....	162
<b>CONCLUSION AND RECOMMENDATIONS</b> .....	162
7.1. Conclusions.....	162
7.2. Contribution of the Research.....	162
7.3. Recommendations.....	163
<b>REFERENCES</b> .....	164
<b>APPENDIX</b> .....	195
Appendix 1. Code Boxes .....	195
Appendix 2: Machine learning libraries and modules.....	212
Appendix 3: Board of Postgraduate Research Approval .....	215
Appendix 4:Ethical approval .....	217

## LIST OF FIGURES

Figure 2. 1. Interaction of pattern elements source. adapted from (Noble, 1998).....	13
Figure 2. 2. The relation between the pattern elements (Barhoom, 2015). .....	18
Figure 2. 3. Relationship between Patterns. (Author). .....	19
Figure 2. 4. Depicts the number of incidence attacks in relation to the year .....	28
Figure 2. 5. Showing the media and host layers Source: <a href="http://www.cables-solutions.com/wp-content/uploads/2016/06/OSI-Model.png">http://www.cables-solutions.com/wp-content/uploads/2016/06/OSI-Model.png</a> .....	34
Figure 2. 6. The OSI Layers Data Flow .....	35
Figure 2. 7. OSI Models and Attacks. (Manninen, 2018) .....	36
Figure 2. 8. OSI Addressing Security at Each Layer. (Holl, 2003).....	42
Figure 2. 9. Linddun Phases. ....	51
Figure 2. 10. Octave Process. ....	53
Figure 2. 11. Three-step technique to evaluating organizational threat profiles .....	54
Figure 2. 12. Summary of DREAD threat model.....	56
Figure 2. 13. Stages of PASTA .....	59
Figure 2. 14. Categories of CVSS. ....	60
Figure 2. 15. TRIKE methodology .....	61
Figure 4. 1. OSI and User Interaction .....	73
Figure 4. 2. Extended OSI Model. ....	74
Figure 4. 3. Three-layer network Security domain (TLNSD). ....	76
Figure 4. 4. Attack surface Identification. ....	80
Figure 4. 5. The method for developing abuse cases based on Microsoft threat modeling and attack patterns. ....	82
Figure 4. 6. Identifying existing attack pattern. ....	83
Figure 4. 7. Retrieving Relevant Attack Pattern and Deriving Alternative Attack Pattern.....	86
Figure 4. 8. Risk Management Process .....	87
Figure 4. 9. Risk Management process model .....	88
Figure 4. 10. ISSRM process (Mayer, 2009) .....	89
Figure 4. 11. Information System Security Risk Management (ISSRM) metamodel .....	89
Figure 4. 12. Network Architecture Security Pattern Based Model.....	90
Figure 5. 1. First stage of the model.....	92
Figure 5. 2. Second Stage of the Model .....	93
Figure 5. 3 Third Stage of the Model. ....	93
Figure 5. 4. UNSW-NB15 Dataset Description. ....	94
Figure 5. 5. The Testbed Visualization for UNSW-NB15. ....	95
Figure 5. 6. Architecture for Generating UNSW-NB15 Data Set.....	96
Figure 5. 7. Percentage of malicious log in Un_1 Un2 Un3 and Un4.....	102
Figure 5. 8. Distribution of Normal vs. Malicious Traffic. ....	103
Figure 5. 9. Top Ten Malicious Protocol Count. ....	104
Figure 5. 10. Distribution of Malicious Protocol. ....	107
Figure 5. 11. % distribution of malicious protocol .....	108
Figure 5. 12. Treemap Visualization of Attack Distribution.....	109
Figure 5. 13. Attack Distribution UDP Protocol.....	109

Figure 5. 14. Attack distribution TCP protocol.....	110
Figure 5. 15. Attack Distribution UNAS Protocol.....	110
Figure 5. 16. Attack Distribution Other Protocol.....	111
Figure 5. 17. Stacking Ensemble Approach.....	121
Figure 5. 18. Meta learner prediction flows.....	123
Figure 5. 19. KMeans Clustering flow.....	125
Figure 5. 20. Attack Distribution for Each Surface of Attack.....	132
Figure 5. 21. CAPEC/STRIDE database Schema.....	138

## LIST OF TABLES

Table 1. 1.Difficulties in defending against attacks. ....	2
Table 2. 1.Well-known pattern forms. (Author). ....	14
Table 2. 2.GOF pattern Format. Adapted from (Gamma et al., 1995).....	15
Table 2. 3.Alexandrian pattern Format. Adapted from (Alexander 1979).....	16
Table 2. 4.POSA format.....	17
Table 2. 5.Coplien pattern format. Adapted From (Coplien,2002).....	17
Table 2. 6.STRIDE Threat Categories definition and property violated.....	55
Table 2. 7.Summary Threat Modeling Methods Features.....	63
Table 3. 1.Dataset properties and their value ranges (Ring et al., 2019) .....	68
Table 3. 2.Study dataset selection criteria.....	68
Table 5. 1.Data Set Statistics.....	96
Table 5. 2.Feature Attributes.....	97
Table 5. 3.Data Types. ....	97
Table 5. 4.Attack type categories.....	98
Table 5. 5.Output of .head( ) on the dataset showing column headers with no name. ....	100
Table 5. 6.Dataset with New Column Headers. ....	100
Table 5. 7.Sample of malicious logs from un_1 .....	101
Table 5. 8.Sample of malicious logs from un_2 .....	101
Table 5. 9.Sample of malicious logs from un_3 .....	101
Table 5. 10.Sample of malicious logs from un_4 .....	102
Table 5. 11.Missing Values in The Malicious Dataset. ....	104
Table 5. 12.Sample Filtering malicious TCP traffic. ....	105
Table 5. 13.Sample Filtering malicious UDP traffic.....	105
Table 5. 14.Sample Filtering malicious Unas traffic. ....	105
Table 5. 15.Output of the top 3 significant malicious protocols. ....	106
Table 5. 16.Low significant malicious protocols. ....	106
Table 5. 17.Other Protocols. ....	107
Table 5. 18. Protocol Distribution of Tmalpt and Nunas. ....	107
Table 5. 19. Percentage of Attack Distribution within the Malicious Traffic.....	108
Table 5. 20. Sample of Ports with 100 Attacks and Above.....	112
Table 5. 21. Statistics of the Malicious Logs per Protocol.....	112
Table 5. 22. Main dependency protocols timings per state .....	112
Table 5. 23. Mean of Time Distribution in Malicious Packets. ....	113
Table 5. 24. Attack Category Time Duration in the Malicious Dataset. ....	113
Table 5. 25. Attack category time duration for TCP UDP UNAS. ....	114
Table 5. 26. Mean Duration of Attacks per Protocol .....	114
Table 5. 27. Missing values in malicious dataset .....	115
Table 5. 28. Missing values statistics.....	116
Table 5. 29. Sample Missing values filled with Median. ....	117
Table 5. 30. Performance metrics scores.....	126
Table 5. 31. Columns with the object data type .....	127
Table 5. 32. Confirmation of columns with the object data type. ....	128

Table 5. 33. Selected features for machine learning algorithm.....	129
Table 5. 34. Mapping to attacks surfaces.....	130
Table 5. 35. Sample of Mapping the outputs to the SelectKBest algorithm data.....	131
Table 5. 36. sample output for the attack surfaces.....	131
Table 5. 37. Attack Distribution for Each Surface of Attack.....	133
Table 5. 38. CAPEC Dataset Features.....	133
Table 5. 39. Generic attacks from CAPEC.....	134
Table 5. 40. Exploits attacks from CAPEC.....	134
Table 5. 41. Fuzzers' attacks from CAPEC.....	135
Table 5. 42. DOS attacks from CAPEC for dos and dos2 respectively.....	135
Table 5. 43. Reconnaissance attacks from CAPEC.....	135
Table 5. 44. Analysis attacks from CAPEC.....	135
Table 5. 45. Shell attacks from CAPEC.....	136
Table 5. 46. UNSW-NB_15 found in CAPEC.....	136
Table 5. 47. UNSW-NB_15 Attacks Category vis a vis CAPEC Attack Patterns.....	137
Table 5. 48. Mapping of STRIDE to CAPEC.....	139
Table 5. 49. Risk Analysis of Filtered Attacks.....	140
Table 5. 50. Quantitative Risk Analysis of the Filtered Attacks.....	140
Table 5. 51. Risk Score of the Filtered Attacks.....	141
Table 5. 52. Quantitative risk analysis of all the filtered attacks.....	141
Table 5. 53. Quantitative Risk Score of all the Filtered Attacks.....	142
Table 5. 54. Qualitative Risk Mapping Score of all the Filtered Attacks.....	143
Table 5. 55. High, Medium, low Risk attacks.....	144
Table 5. 56. Defense Mechanisms for the Filtered Attacks.....	145
Table 6. 1. Worm Attack Pattern and Defense.....	151
Table 6. 2. General Backdoor Attack Pattern.....	155
Table 6. 3. Evaluating Worm and Backdoor Patterns on The Basis Of Their Forces.....	160
Table 6. 4. Evaluating the Worm and Backdoor Patterns Using STRIDE.....	161

## **LIST OF ABBREVIATIONS AND ACRONYMS.**

ACCS-Australian Centre for Cyber Security.

ACM- Association of Computer manufactures.

AFRL- Air Force Research Laboratory.

APT- Advanced Persistent Threats.

ARP-Address Resolution Protocol.

BSIMM- Building Security in Maturity Model.

CAPEC-Common Attack Pattern Enumeration Classification.

CIA- Confidentiality, Integrity, and Availability.

COBIT- Control Objectives for Information and Related Technologies.

COTS- Components off-the-shelf.

CPU- Central Processing Unit.

CRAMM- CCTA Risk Analysis and Management Method.

CSTG- Cyber Systems and Technology Group.

CSV- Comma-Separated Values.

CVE- Common Vulnerabilities and Exposures.

CVSS- Common Vulnerability Scoring System.

CWE- Common Weakness Enumeration.

DARPA-Defense Advanced Research Projects Agency.

DDoS-Distributed Denial of Service.

DFD- Data Flow Diagram.

DHCP- Dynamic Host Configuration Protocol.

DOS-Denial of Service.

DREAD- Damage, Reproducibility, Exploitability, Affected users, Discoverability

EBIOS-Expression des Besoinset Identi\_cationdes Objectifs de Sécurité.

GoF- Gang of Four.

HCI- Human-Computer Interaction.

HIPAA- Health Insurance Portability and Accountability.



HITRUST- Health Information Trust Alliance.

HTTP-Hyper Text Transfer Protocol.

ICMP- Internet Control Message Protocol.

IDS- Intrusion Detection System.

IEC- International Electro technical Commission.

IOTCA- Internet of Things Cybersecurity Alliance.

ISACA- Information Systems Audit and Control Association.

ISO- International Standards Organization.

ISSRM-Information System Security Risk Management.

KDD-Knowledge Discovery and Data Mining.

LINDDUN-Linkability, Identifiability, Non-Repudiation, Detectability, Disclosure of Information, Unawareness, Non-compliance.

MAC-Mac Address Control.

MEHARI-Methode Harmoniséed' Analyse du Risque Informatique.

MIT-Massachusetts Institute of Technology.

MitM-Man in the Middle.

MITRE ATT&CK- MITRE Adversarial Tactics, Techniques, and Common Knowledge.

MITRE- Massachusetts Institute of Technology Research and Engineering.

NIC- Network Interface Cards.

NIST- The National Institute of Standards and Technology.

OCTAVE- Operationally Critical Threat, Asset, and Vulnerability Evaluation.

OOPSLA- Object-Oriented Programming, Systems, Languages & Applications.

OSI- Open System Interconnect Model.

OWASP-Open Web Application Security Project

PASTA-Process for Attack Simulation and Threat Analysis

PHI- Protected Health Information.

PLOP- Pattern Language of Programs.

POSA- Pattern-Oriented Software Architecture.

PTTES- Patterns for Time-Triggered Embedded Systems.

R2L-Root to local.

SDL- Security Development Lifecycle.

SRA- Security risk assessment.

SSL- Secure Socket Layer.

STP- Spanning Tree Protocol.

STRIDE- Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privileges.

TCP- Transmission Control Protocol.

TF-IDF- Term Frequency-Inverse Document Frequency.

TLNSD- Three Layer Network Security Domain.

U2R- User to Root.

UDP-User Datagram Protocol.

UML-Unified Modified Language.

VLAN- Virtual Local Area Network.

# CHAPTER ONE

## INTRODUCTION

### **1.1. Background Information.**

Networks are inflicted by innumerable attacks both internal and external, which usually result in damaging effects. These attacks can be shattering for both the users and the organization as they can result to consequences such as exposure to litigation and civil suits, loss of productivity, attacks from cyber terrorists, data and identity theft and even loss of reputation (Ciampa, 2017b). With the fast growth of network capable portable devices, mobile apps services, cloud computing services, the need on the networks is increasing day by day (Yao et al., 2014). The scenario in terms of efficiency has also literally opened up the avenue for attackers to propagate numerous attacks against network resources (Verizon 2013 Annual Review, 2013). Motives for these attacks appear equally diverse like personal reasons, prestige, criminal, commercial, or ideological in nature (G. Kumar, 2016).

Despite crafting defense strategies, there still exist a myriad of challenges in protecting networks against attacks, this is largely attributed to greater sophistication of attacks where attack tools vary their behavior so that the same attack appears different each time; Speed of attacks where attackers can launch attacks against millions of computers within minutes; user confusion where users are required to make difficult security decisions with little or no instructions at all hence creating vulnerabilities to network systems (Ciampa, 2011). Table 1.1 elaborates difficulties in defending against attacks.

Attacks can also be traced back to well-known security problems and vulnerabilities such as not hashing passwords within a database, storing sensitive information on servers within the DMZ. These are clear indications that there is either a poor understanding of security issues or that the priority given to security within networks is low (Schumacher et al., 2006).

Security awareness is lacking among enterprise planners, system architects and developers, and operations managers. They therefore place a lot of reliance on security experts to comprehend their security issues and offer security solutions. To meet the demand, there aren't enough security experts, though. Additionally, the security experts frequently find themselves reusing the same solutions for every business or system development project. They are

wasting their time by doing this and are prevented from tackling more difficult problems (Schumacher et al., 2013a).

**Table 1. 1.Difficulties in defending against attacks.  
(Ciampa, 2017a).**

<b>Reason</b>	<b>Description</b>
Universally connected devices	Attackers from anywhere in the world can send attacks.
Increased speed of attacks	Attackers can launch attacks against millions of computers within minutes.
Greater sophistication of attacks	Attack tools vary their behavior so the same attack appears differently each time.
Availability and simplicity of attack tools	Attacks are no longer limited to highly skilled attackers.
Faster detection of vulnerabilities	Attackers can discover security holes in hardware or software more quickly.
Delays security updating	Vendors are overwhelmed trying to keep pace updating their products against the latest attacks.
Weak security update distribution	Many software products lack a means to distribute security updates in a timely fashion.
Distributed attacks	Attackers use thousands of computers in an attack against a single computer or network.
Introduction of BYOD	Organizations are having difficulty providing security for a wide array of personal devices.
User confusion	Users are required to make difficult security decisions with little or no instruction.

For an organization, the expense of repairing these vulnerabilities and the risk involved with them after implementation are substantial. Although there are many best practices available to address the problem of security vulnerabilities, these approaches are sometimes challenging to reuse since the best practices are implementation-specific. Therefore, there is a larger need to comprehend the underlying causes of security problems in networks, where they come from, and what can be done to alleviate them (Dougherty et al., 2009).

Technology has connected the world through networks, which are getting more complicated and widely dispersed. Networks are now connected to the cloud, apps, and other Internet of Things devices, as opposed to being primarily constructed around dispersed data centers in the past (Shah, 2018) .These networks facilitate the operation of applications, the majority of which are distributed in nature and backed by web services, interfaces, and agents, making them complicated and challenging to comprehend, create, and manage. Future predictions indicate that this trend will continue, with 125 billion connected devices estimated to be in use by 2030. (Campbell, 2019).

Given these complexities, errors have become common and vulnerabilities are on the rise coupled with the fact that these networks and resources within it store valuable information

which attract attacks. According to AlgoSec (2013), most security administrators have fallen into the trap of bolting on more and more security layers and rules in attempts to shield the network and vital assets from cyber-attacks. As a result, the network environment has become more complicated, which has raised the possibility of dangers arising from human mistake and incorrect setup. The situation has also worsened with the increase in the use of mobile devices with enhanced capabilities such as smart phone which is now becoming the leading target of attackers (Ruggiero & Foote, 2011) (Salerno et al., 2011).

Although the importance of security is widely acknowledged, it is still an afterthought in many projects. That is why a secure design should be a critical component when developing and deploying a system. When a system is not designed with security in mind it is definitely bound to fail in this age of cybercrime (Schumacher et al., 2013a). Olagunju et al., (2013), points out that the root cause of most infrastructural failures such as buildings, bridges and roads can be traced back to poor architectural designs. The same can be underscored in the world of Information Technology where some attacks have been traced back to poorly designed software, poorly designed network, and use of protocols not critically designed with security in mind (Kocher et al., 2004).

According to Yoder and Barcalow (1998) developers build systems without security in mind. This is attributed to the fact that they focus more on trying to learn the domain rather than how to protect the system. They tend to put a lot of emphasis on satisfying user needs than security. When the time comes to deploy these systems, it quickly becomes apparent that adding security is much harder than just adding a password protected login screen. The challenge with security is that security has always been considered an afterthought. Even within the conventional system development life cycle security is considered a non-functional requirement rather than a functional requirement which means it is a function that is taken into consideration after deployment rather than being incorporated right from the requirement soliciting phase (Galloway, 2019).

In a similar vein, according to (Schumacher et al., 2006), the enterprise context and requirements that shape system security are not explicitly addressed and are not incorporated into system architectures. As a result, it is necessary to start addressing security up-front rather than using the "repair-service" approach that is commonplace at the moment. Security

problems should be addressed at every stage of the system development process, (Devanabu & S, 2000). Ignoring security issues or deferring them until later phases of system development could be risky because it is difficult to retrofit security into a system later on (Riaz et al., 2007). As a result, it is critical to prevent mistakes since a well-thought-out design that includes security considerations will make it easier to react to changing security requirements (Yoder & Barcalow, 1998).

According to (Schumacher, 2003a) It is difficult to get security right if one does not start from the design level, noting that whereas in systems development process there are mechanical aids to detect coding errors there is no such aids for detecting design errors making it difficult to get security right when the system is already commissioned. Furthermore, for a secure system there is need for security requirements to be analyzed, security flaws to be discovered, designed at the early stages of systems development process in order to avoid a catastrophe.

According to (Fernández et al., 2010), a good design process is essential for producing secure systems. The basic premise behind their technique is that security principles should be implemented at every stage of the system lifetime, and that compliance with security principles should be checked at each stage rather than being a consideration after deployment.

Several approaches to building secure systems exists of which the prominent ones are based on *secure coding*: they include Microsoft's Security Development Lifecycle (SDL) (Microsoft, 2019), OWASP's CLASP (C.L.A.S.P., 2019) and Building Security in Maturity Model (BSIMM) (McGraw & Young, 2016). (Gregoire et al., 2007) made a detailed comparison of approaches and looked for similarities and differences, as well as suggested improvements. They concluded that while code-based security is a valuable approach in designing and build secure systems, it cannot produce secure systems by themselves, but can be a good complement to model-based methods.

Some of the well-known model approaches such as UMLsec (Jürjens, 2005); CORAS (den Braber et al., 2007); SecureTropos (Mouratidis & Giorgini, 2007) and Goal-Risk (Ansar & Khan, 2018) have also been proposed as security approaches building secure system. But still they are not reliable, not well defined on the basis of security features and design patterns (Rehman & Mustafa, 2013).

In summary, security is frequently overlooked in system design and implementation. Rather than retrofitting, it is necessary to handle security from the start. Many security breaches may be traced back to well-known security issues that continue to crop up. The primary objective in these circumstances is to improve functionality and performance rather than to reduce risk. Therefore, in line with these challenges we need to find a way to conceptualize, design, build and maintain a network that can try and withstand attacks basing it on pattern theory. Pattern Theory is a research discipline that was conceptualized by professor Ulf Grenander in the late 1960s and has undergone further development by Grenander and the Pattern Theory group at Brown University (Grenander et al., 2007). Pattern theory is a way to approach patterns through mathematical formalism '*a way of reasoning about patterns*', it can be presented using analytical tools or computational methods. On pattern theory, emphasis is pegged on structure of the patterns themselves and not the recognition of patterns.

Patterns are a well-established system development strategy. They are designed to collect domain-specific expert knowledge in the form of documentation with a particular structure and proven solutions for recurrent problems (Schumacher et al., 2006).

(Appleton, 2000) defines a pattern as an identified nugget of instructional information that contains the basic structure and understanding of a successful family of proven solutions to a recurrent problem that develops inside a certain context and system of forces. From the definition it is seen that patterns can be applied within any discipline going by various research done over the previous years and though it has its roots in the architectural field it has received recognition in the field software engineering. Currently patterns have been applied in diverse discipline e.g. psychology, pedagogy, enterprise development and telecommunication (Hamid, 2014)

The benefits of "reusability" are one of the reasons for its widespread appeal. Patterns, in general, are organized texts prepared by professionals to give tested and verified answers to frequently occurring issues in a certain environment. The power of such documentation is that knowledge and experience are no longer isolated to the minds of specialists, but are documented in a form that can be easily accessed and shared (Lakhani & Faisal, 2015).

Pattern-based approach has gained more attention recently in systems engineering by addressing new challenges that were not targeted in the past (Hamid, 2014). They have been

applied extensively within the system architecture for real-time embedded systems, middleware's and distributed systems (Douglass, 2002), and it is now gaining momentum in the field of information security engineering (Schumacher, 2003a) by promoting the use of patterns in the form of reusable design artifacts.

The Pattern based approach examines three different challenges: first is “mining”, which involves discerning patterns from existing systems, the second challenge is “hatching” which involves selection of the appropriate pattern; and lastly “application” which is the effective use of pattern during the system development process. These three challenges are dependent on diverse expertise not limited to mathematics, graph theory, logic, stochastic modeling, and hardware and software design(Sietz, 2011).

Patterns provide a strong foundation for building security and dependability into systems. Neumann advocates for the necessity for 'principled' systems, which are founded on sound conceptual approaches and patterns that allow for the implicit application of principles (Neumann, 2004) .

## **1.2. Statement of the problem.**

One of the most important reasons why we do have challenges in securing network architecture is their complex nature (Bocetta, 2019). Another important reason is that most developers and administrators build and apply security in a “helter-skelter” manner and when securing a network, the different parts are secured using specific products or mechanisms.

According to (Szabo et al., 2015) the Network Architecture model has a fundamental design problem, in that it allows different layers to work without the knowledge of each other and information flows up and down to the next layer as the data is being processed. Thus, if one layer is hacked communication can be compromised without the subsequent layer noticing anything wrong.

Currently each security solution uses different distinct mechanisms even though the problems being solved at each level are largely the same (Small, 2012). There hardly exists a holistic approach in regards to the complete system. If attempted, one may find that different models may be applied in different parts of the network architecture. Securing a network requires a holistic approach if it is to guard against attacks and the practice of relying on security



components, cannot make the whole network secure if they do not work in a coordinated way and protect all parts of the system.

There is a great need for a holistic approach, Fernandez, (2009) describes it *as* “Covering all architectural levels and all units” for you to implement a secure network architecture. It is important to have a system view of the network architecture, and there is a great need to unify the security approach targeting these layers, and these can be achieved by using patterns since it can provide a holistic view of security, which is a fundamental principle to build secure systems (Fernandez, 2009).

### **1.3. Objectives of the Study**

#### **1.3.1. Main Objective of the Study**

The main objective of the study was to develop a holistic security pattern-based model for the network architecture and test its adoption towards the enhancement of network security.

#### **1.3.2. Specific Objectives.**

1. To assess the techniques, models, frameworks that guide in the design and development of a secure network architecture.
2. To conceptualize and develop security pattern-based model for the network architecture
3. To test the conceptual security pattern-based model
4. To evaluate the generated patterns contribution to network security assurance.

### **1.4. Significance of the Study.**

The study's findings on the pattern model will have significant practical implications for organizations. By adopting this model, organizations can strengthen their network security, reduce vulnerabilities, and improve their incident response capabilities, ultimately safeguarding critical assets and data.

Additionally, this study will make valuable contributions to the body of knowledge in network security by exploring and documenting the effectiveness of defensive pattern models. It can serve as a solid foundation for further studies in the rapidly evolving field of cybersecurity.

Not only will network security professionals and practitioners benefit from the study's insights and recommendations, but the defensive pattern models themselves also provide guidance on improving security postures. The findings of this study can inform and shape best practices in the field.

### **1.5. Scope of the Study**

The study is specifically centered around the domain of network security. It addresses challenges, vulnerabilities, and threats within network architectures and aims to provide solutions to enhance overall security. The primary focus is on the development and evaluation of defensive pattern model. The model is conceptualized to offer proactive and comprehensive security measures, and its effectiveness will be assessed in the context of network security. The scope also involves a holistic approach to network security. The holistic pattern model is designed to address security challenges across multiple layers of network architecture, offering a comprehensive and coordinated strategy. In summary, the scope of the study is focused on addressing network security challenges through the development and evaluation of defensive pattern models, with an emphasis on a holistic approach within practical organizational contexts.

### **1.6. Assumption of The Study**

The study assumes that current challenges in network security, stemming from the complexity of networks and a fragmented security approach, necessitate a coordinated and holistic defense strategy. To address this, the study developed a security pattern-based model by integrating various frameworks and constructs. The assumption is grounded in the belief that the proposed model, tested on the Intrusion detection dataset, can fill gaps in existing security models and offer a comprehensive defense against diverse cyber-attacks

### **1.7. Rationale of the Study**

This study is motivated by the ongoing challenges faced by network security experts in protecting complex networks, attributed to the fragmented nature of current security strategies. The rationale is grounded in the need for a comprehensive and coordinated security approach. The proposed model, based on security patterns and integrating various frameworks, aims to fill the existing gaps. The research employs a simulation approach, to

furnish empirical evidence for the effectiveness of the model. The analysis of cyber-attacks across network layers and the identification of new attack patterns contribute to the study's rationale, emphasizing the importance of a holistic approach to enhance network security resilience.

## **CHAPTER TWO**

### **LITERATURE REVIEW**

This chapter is entitled “Literature Review”. This section first provides an in-depth study on patterns in the first three sections namely (1) History of Patterns, (2) Design Patterns, and (3) Anti-Patterns. It then discusses attacks noted on networks and the resultant damage due to them under (4) Attacks on Networks and Havoc Caused. Thereafter, a synopsis of the OSI model and the part it plays in securing networks is provided in (5) The OSI Model and its Roles in Securing Networks. Aspects of network security such (6) Network Security Architecture Role in Securing Networks, (7) Network Security Design, and (8) Frameworks Guiding Security of Networks are then discussed. The discussion in (9) Threat Model and *Threat Modeling* provides an understanding of how models are developed to standardize and ease disaster planning and preparedness processes in the face of security threats. Finally, the section on (10) *Security Risk Assessment and Network Security* looks at the process(es) of identifying security risks and vulnerabilities and some of the methodologies useful in this process

#### **2.1. The History of Patterns**

A pattern is a regularity that can be seen in nature or in designs made by people. From a prescriptive point of view, a pattern is a template that can be used to make new instances. From a descriptive point of view, the parts of a pattern that repeat in a predictable way can be seen and identified (H. Zhu, 2014) .

Patterns explain and predict regularities in a subject area, just like theories do in science. The pattern roots can be clearly traced in the world of architectural design of structures and documentation of the architectural best practices and lessons learnt. Abstracting these problems solutions and showing how they can be resolved through a series of related augmented steps, successfully defines best practices that can be used to solve a particular problem and allows us to clearly reason about them. The focus of patterns is not pegged so much on technology but rather on documenting and supporting sound engineering and design processes (Appleton, 2000).

Patterns became more popular in computer science after the "Gang of Four" (GoF) book came out in 1994. ( e Gamma et al., 1995). In the same year, the first Pattern Languages of

Programming (PLoP) Conference was held. The next year, the Portland Pattern Repository was set up to keep track of patterns (Repository, 2014) .

In his seminal books (Alexander, 1977) and (Alexander, 1979a) laid the foundation of the “pattern concept” in the context of architectural design and building. With his team they spent more than twenty years developing an approach to civil architectures using patterns (Ammar & Hany, 2003), identifying more than two hundred and fifty patterns that are instrumental up-to-date in planning of regions, towns, neighborhoods, buildings and rooms, ending with detailed construction plans. The team was also instrumental in identifying that a pattern should consist of a context-problem-solution trichotomy structure, known as the Alexandrian form.

(Beck, 1987) began to try out the idea of using patterns in programming. Specifically, they used Smalltalk, which is a prototyping environment for user interfaces, to design user interfaces. At the ACM Conference OOPSLA, they showed what they had done. This work was important because it led to patterns that helped a lot in making user interface designs less complicated. Since then, Beck, Cunningham, and others have written many more patterns based on this work (Buschmann et al., 1996) .

In 1991 Peter Coad when reviewing the work by (Coplien, 1991) noted that the book contained valuable C++ best practices for abstractions, although the term pattern was not used and concepts within the book were not written in pattern style he was able to use the concept in coming up with patterns that can be applied in object oriented programming (Coad, 1992).

In 1991 and 1992 the pattern community met in Object-Oriented Programming, Systems, Languages & Applications conference (OOPSLA) where the Gang of Four (GoF) presented a compilation of patterns that were discussed by several key figures of the pattern community.

The pattern events of 1993 saw the formation of the hillside Group sponsored by Kent Beck and Grady Booch as the de-facto organization for the pattern community, it is from here that the patterns work by Erich Gamma and the group (GoF) was adopted and formed the reference point of patterns in the field of software engineering. In April 1994 the same group met again and planned the first ever conference on pattern language of programs (PLOP) and in the month May 1995 and the first proceedings of the conference was published (Coplien & Schmidt, 1995).

In 1995 the textbook (Gamma, et al., 1995), was considered the core reference of the pattern knowledge in the world of software engineering. It was latter followed the POSA book by the pattern community (Buschmann et al., 1996) which also played a critical reference book for patterns.

The PLoP conference was held every year in Allerton Park, University of Illinois at Urbana-Champaign, until 2004. Since then, it has rotated between Allerton Park and being held with the big computer science conference OOPSLA, the Agile Conference in 2009, and PUARL in 2018. From that time other pattern conferences have taken place such as AsianPLoP, ChiliPLoP, EuroPLoP & KoalaPLoP (Schumacher et al., 2006) amongst others. These conferences are important since they provide avenue for discussions on improvement of patterns and working groups on several pattern topics, and clearly demonstrates a continuous progress of the success of patterns and a steady growth of the pattern community since when it was first started by the hillside group in 2001.

The fact that patterns represent a "grass roots" effort to build upon and draw from the collective knowledge of talented designers is a key factor in their success. Rarely do new development projects address genuinely fresh challenges that necessitate completely original answers. Developers may occasionally come up with comparable answers on their own or frequently recollect an issue they previously handled in a different circumstance, reusing its essence and customizing its details to tackle the new difficulty. For both common and unique design difficulties, experienced developers may draw on a wide corpus of such solution designs. Their new application development is guided by this actual expertise.

Finding similarities between application-specific design challenges and their solutions leads naturally to the idea of patterns since they frame these answers and their connection to the problem in a more understandable way. A pattern may be defined as: A solution to a problem that occurs inside a particular context, from a very broad birds-eye perspective.

### **2.1.1. The Notion of Patterns**

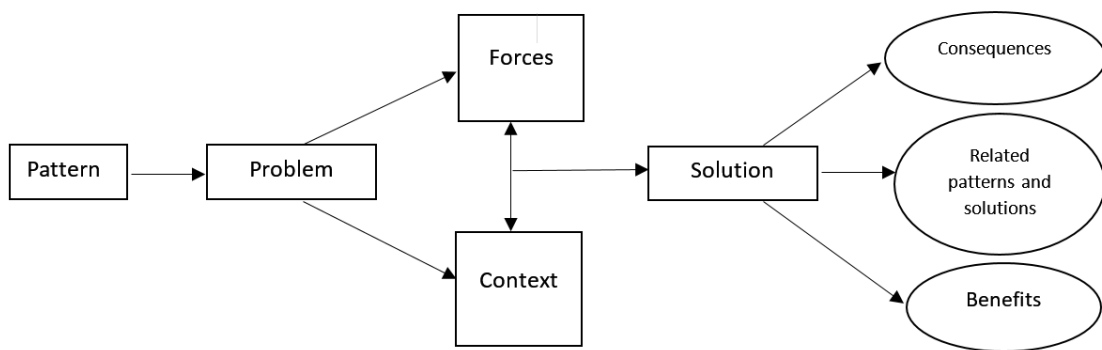
A "pattern" is defined as both a process and an object, where the "process" creates the "object." According to (Alexander, 1979), a "thing" can be represented as either code or a high-level design that shows an object's structure and behavior. In other words, a pattern is both a spatial arrangement of pieces that solves a specific issue or prevents a specific issue from arising and

a set of accompanying instructions to best produce this arrangement of elements (Schumacher et al., 2006).

Alexander’s idea of patterns has gained acceptance in several areas within the research community, in software engineering several authors and experts have provided their views on patterns as follows

GoF see patterns as a tool to record and share "best practices," or methods that have proven effective for seasoned designers (Gamma, et al., 1995). According to Appleton (2000), a pattern is an identified piece of educational knowledge that encapsulates the fundamental organization and wisdom of a successful family of tested solutions to a recurrent issue that develops inside a certain context and system of forces. It is described as a way to convey common sense and abstractions that are difficult to represent in other ways by Gabriel (1996). While Rising (1998) views it as artifacts that have been found in several systems already in place.

The above definitions clearly brings out the core concept of what a pattern is, but for it to be clear, avoid ambiguity, achieve convenience, and usage, as shown in Figure 2.1, a typical pattern should have the following elements: a name, a description of the problems it applies to, an analysis of the forces (the significant considerations and consequences of using the pattern), a sample implementation of the pattern's solution, references to known uses, and a list of patterns that are related to it.



**Figure 2. 1.Interaction of pattern elements source. adapted from (Noble, 1998)**

Figure 2.1 shows how the problem is typically explored in terms of its context and the applicable design factors that serve as the foundation for the solution. The solution's job is to

resolve the design pressures in such a manner that it creates advantages, some repercussions, and follow-on challenges that lead to the patterns' application.

### 2.1.2. Describing and documenting patterns

Patterns are usually described using templates, and there are several template formats to that effect. This format informs the design principle underlying the pattern and how to apply patterns to solve recurring problems. These templates have predefined sections that are used to document the different aspects of patterns. Depending on a pattern template one may find different pattern elements being used. For instance, one may find there is use of the word “Motivation instead of context” in a different template. It is worth noting that all known pattern templates contain the basic pattern structure and the basic elements (Schumacher, 2003b).

Different writers have different methods for documenting patterns. What both styles have in common is that design ideas are communicated in casual English initially, then elaborated with informative graphics and detailed code samples (H. Zhu, 2014) Table 2.1 shows several well-known pattern formats.

**Table 2. 1.Well-known pattern forms. (Author).**

<b>Pattern Format</b>	<b>Description</b>
GoF	Followed for the Gang of Four patterns (Gamma et al., 1995).
Alexandrian	Followed for the Alexander patterns (Alexander, 1979).
POSA	Followed for the Patterns of Software Architecture (Buschmann al., 1996).
Coplien	Followed for the James Coplien Patterns. (Coplien.,2000)

#### 2.1.2.1. GoF format

The GoF authors ( e Gamma et al., 1995) had published a text book called “Elements of Reusable patterns”. It has too many elements. However, it gives detailed understanding about the pattern from problem statement to related patterns. To understand and apply GoF patterns, the user should have knowledge in programming (C++), UML representations, and acquaintance with several systems in which GoF patterns are already used. Table 2.2 shows the GOF pattern Template.



Table 2. 2.GOF pattern Format. Adapted from (Gamma et al., 1995)

GoF Elements	Description
pattern name and classification	<i>Pattern Name:</i> A concise name of the pattern. This needs to be carefully selected because it will eventually become a component of the design vocabulary.  <i>Classification:</i> Creational, Structural, Behavioral
Intent	A short statement about what the design pattern does/ the problem it addresses/ the rationale behind the pattern
Also Known as	Other names for the pattern (if any)
Motivation	A scenario that illustrates design problem and how the structures in the pattern solve the problem
Applicability	This section presents situations where the design pattern can be used
Structure	A graphical representation of the classes in the design pattern
Participants	The classes and/or objects participating in the design pattern (and their responsibilities)
Collaborations	This section describes how the participants mentioned in the previous section carry out their responsibilities.
Consequences	What are the pros and cons of using the pattern? How the pattern does supports this objectives?
Implementations	Hints and techniques for implementing the pattern
Sample Code	Code fragments that illustrate how the pattern might be implemented
Known Uses	This section presents examples of the pattern encountered in real systems
Related Patterns	This section presents other patterns that are strongly related to the pattern, and the important references between the patterns.

### 2.1.2.2. Alexandrian format

The Alexandrian format is the first pattern form proposed by Alexander (1979) to guide users to generate solutions for the problems. Table 2.3 shows the Alexandrian format.

Table 2. 3.Alexandrian pattern Format. Adapted from (Alexander 1979)

Alexandrian Elements	Description
Name	meaningful name
Problem	The statement of the problem a pattern aims to solve, the problem description should give a designer the knowledge of the problem a pattern solves, which enables the designers to know when to apply given pattern.
Context	a situation giving rise to a problem, Context are the preconditions under which the problem and its solution seem to recur, and for which the solution is desirable, can also be viewed as the initial configuration of the system before the pattern is applied to it.
Forces	A description of relevant forces and constraints, represent a concrete scenario which provides the motivation for the use of pattern in certain context to solve a certain problem.
Solution	Proven solution to the problem, The description of the solution may indicate guidelines to keep in mind (as well as pitfalls to avoid) when attempting to create a concrete implementation of the solution.
Examples	sample applications of the pattern
Resulting context	(Force resolution) the state of the system after pattern has been applied A good pattern indicates what forces it leaves unresolved, or what other pattern must be applied, and how the context is changed by the pattern.
Rational	Explanation of steps or rules in the pattern describes where the pattern comes from, why it works, and why expert use it.
Related Patterns	static and dynamic relationship with other patterns
Known Use	occurrence of the pattern and its application within existing system helps in validating a pattern, i.e., verifying that pattern is indeed a proven solution to a recurring problem

### 2.1.2.3. POSA format

The POSA format. It is a structured form. POSA form is a lengthy form. It has less focus on applicability's, forces, program code, and structure of a pattern. Table 2.4 shows the POSA format

**Table 2. 4.POSA format**

<b>POSA Elements</b>	<b>Description</b>
PATTERN NAME	The name and a short summary of the pattern
Also known as	Other names for the pattern, if any exist
Example	A real-world example that demonstrates the existence of the problem and the need for the pattern. This helps to explain the problem and related forces better.
Context	Situations / settings where the pattern is applicable
Problem	Problem addressed by the pattern including a discussion of the related forces.
Solution	Fundamental solution to the underlying problem.
Structure	Discusses the structural aspects of the pattern
Dynamics	Typical scenarios describing the run-time behavior of the pattern
Implementation	These are guidelines for implementing the pattern
Example resolved	Further aspects for resolving the example not discussed in earlier sections are presented here
Variants	Very brief descriptions of alternative solutions to the pattern.
Known uses	Examples of the use of the pattern, taken from existing systems
Consequences	Pros and cons of applying the pattern
See also	References to related patterns

#### 2.1.2.4. Coplien format

The Coplien format as shown in Table 2.5 describes patterns in a short form, it has less focus on applicability's, consequences, when to use a pattern, structure, and implementation.

**Table 2. 5.Coplien pattern format. Adapted From (Coplien,2002)**

<b>Coplien Elements</b>	<b>Description</b>
PATTERN NAME	meaningful name
Problem	Problem addressed by the pattern
Context	Situations/settings where the pattern is applicable
Forces	Addresses the issue: "What makes the pattern so difficult?" Commonly a list of bullet points
Solution	The solution to the problem –this is generally layered with the most general interpretation at the highest level and providing more detail as one progresses through the section.
Resulting context	The pros and cons of the pattern
Rationale	This section describes how the pattern works, why it works and why it is "good".

### 2.1.3. Elements of patterns

Though there are several template formats that inform on how to apply patterns to solve recurring problems we are able to see that they share common elements as shown below.

**Name.** -This is what the pattern will be referred to it should be able to stress the action implied by the pattern, it is a handle that can be used to describe a design problem, its solutions, and consequences. The pattern name should be easy to refer and remember in a word or two, how to assign a name to a pattern is dictated by the conventions of a pattern community.

**Context.** -Describes the general environment and conditions under which the problem occurs and shows where and when the pattern will be applied.

**Problem-**At a glance it describes the problem to be addressed by the pattern by stating the goals and objectives the pattern wants to achieve. The problem is used to determine where and when the pattern will be applied.

**Forces-** Shows what considerations are to be accounted for when deciding on a solution to the problem?

**Solution.** -Describes a proven solution through a structure of the elements that make up the pattern, it provides the guidelines and what to avoid when attempting to implement a solution. The context in which a problem occurs should always determine the appropriate solution. Though a problem may have several solutions, in the case of patterns, it should contain one problem, one context, and one solution” (Harrison, 2006).

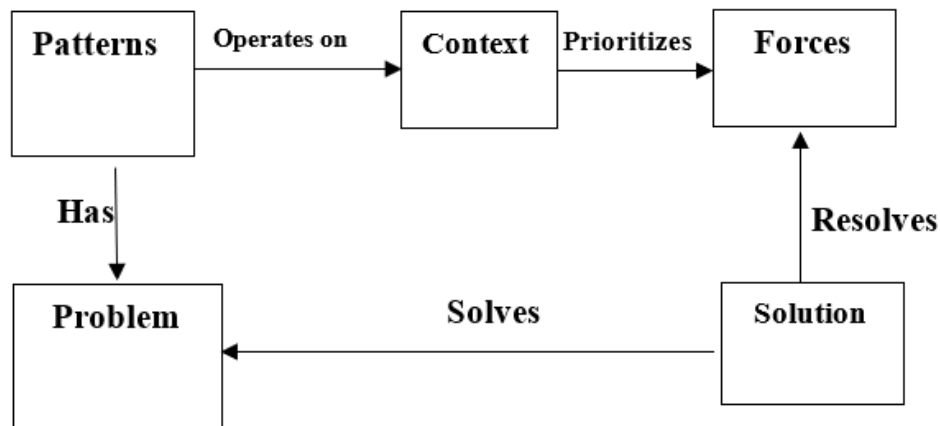


Figure 2. 2.The relation between the pattern elements (Barhoom, 2015).

**Related patterns-** though patterns focus one problem, one context, and one solution. They are not independent because they have relationship to other patterns (Buschmann et al., 1996). A given pattern's suggested solution is frequently implementable with the aid of additional patterns that address the problem's subproblems. A connection between consecutive patterns in a set of patterns is made by such a relationship.

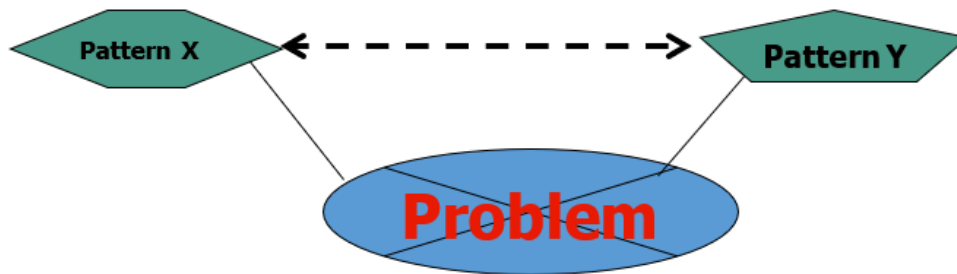


Figure 2. 3.Relationship between Patterns. (Author).

**Examples-** This is one or more of the pattern's sample applications, which help in understanding the pattern's use, the contexts in which it may be used, and how it alters a given situation.

**Resulting Context.** - Defines which forces have been resolved, which ones are still in play, and which patterns may now be applied, it is sometimes referred to as a resolution of forces. It also provides the results, post-conditions, and side-effects that come with using the pattern.

**Rationale.** -It is the account of how the pattern solves the problem, why it solves it and why it is good

**Known Uses.** -They validate patterns as a provable solution to a recurring problem by describing the known manifestations of a particular pattern and how it is applied in a particular systems context, hence they serve as instructional examples.

#### 2.1.4 Challenges for Creation of Patterns

Christopher Alexander's development of design patterns was prompted by a critical examination of deficiencies in traditional architectural and urban planning. The prevalent issues included a lack of user-centered design, resulting in environments that failed to meet the needs and experiences of inhabitants(Alexander, 1979b). Traditional approaches led to

monotonous and inhuman spaces, prompting Alexander to introduce patterns inspired by nature and traditional settlements to infuse diversity, complexity, and wholeness.

To counter the top-down nature of planning and enhance adaptability, Alexander conceived patterns as small, adaptive solutions fostering organic, bottom-up development. Addressing disconnected architecture, patterns were proposed as a tool to establish a design language promoting coherence and connectivity between individual elements and their environment (Talen, 2015). Additionally, patterns aimed to shift the focus from superficial aesthetics to the inherent qualities that enhance livability and align with human needs, countering the prioritization of style over functionality (Klingmann, 2010). In summary, Christopher Alexander's development of patterns aimed to address shortcomings in traditional architectural approaches by promoting a more humane, sustainable, and user-centric design philosophy rooted in the intrinsic patterns of human life and nature.

The genesis of design patterns in the domains of software engineering, specifically through the seminal works of the Gang of Four (GoF), Pattern-Oriented Software Architecture (Posa), and Coplien, was prompted by discerned inadequacies and challenges within the respective realms of software design, architecture, and team dynamics. Each group endeavored to address distinct yet interrelated issues, thus contributing to the establishment of a robust pattern-oriented paradigm.

The Gang of Four's (GoF) design patterns emerged in response to the prevalent difficulty in achieving reusability and flexibility in software components. The intricate web of dependencies and coupled codebases impeded adaptability, necessitating a paradigm shift (Vesiluoma, 2009). GoF patterns, such as the Factory Method, Abstract Factory, and Singleton, sought to furnish a repertoire of solutions to common design predicaments, facilitating the creation of malleable, maintainable software systems.

POSA, on the other hand, delved into the architectural facets of software engineering, recognizing the challenges associated with organizing large-scale software systems. The advent of architectural patterns, encapsulated in the POSA framework, aimed to provide architectural solutions for high-level design concerns (D. C. Schmidt et al., 2013). These patterns addressed the complexities of distributed systems and concurrent programming, offering guidance through constructs like the Broker pattern and Active Object pattern.

In parallel, Coplien's organizational patterns confronted challenges arising from communication breakdowns within software development teams and the shortcomings of conventional project management approaches. Coplien introduced patterns, such as Whole Team and Trust Your Team, to enhance team dynamics and communication efficacy. The organizational patterns also tackled issues in project management through constructs like the Project Charter and Osmotic Communication(Aydinli, 2015). In conclusion, the development of design patterns by GoF, Posa, and Coplien underscores a collective endeavor to rectify inadequacies within the realms of software design, architecture, and team collaboration. These patterns, each tailored to its specific domain, serve as a cohesive and structured response to the multifaceted challenges inherent in contemporary software engineering practices.

### **2.1.5. Benefits Accrued from Development of Patterns**

The introduction of patterns has yielded substantial benefits, influencing the way systems are conceptualized, designed, and implemented. This paradigm shift has reverberated across different facets of the system development lifecycle, fostering improved efficiency, maintainability, and scalability.

patterns, as advocated by the Gang of Four (GoF), have significantly enhanced the reusability and flexibility of software components. By encapsulating proven solutions to recurring design problems, patterns facilitate the creation of adaptable and modular systems. This adaptability ensures that software architectures can evolve seamlessly in response to changing requirements, thereby mitigating the rigidity often associated with traditional design approaches(Zhang, 2011).

Organizational patterns, as championed by Coplien, have played a pivotal role in fostering effective communication and collaboration within software development teams. Constructs such as the Whole Team and Trust Your Team patterns have ameliorated interpersonal dynamics, resulting in increased transparency, shared understanding, and a more cohesive team environment(Santos et al., 2015). This enhanced collaboration is instrumental in tackling complex projects and achieving collective goals.

Pattern-Oriented Software Architecture (Posa) has significantly contributed to the coherence and scalability of software architectures. Architectural patterns provide proven blueprints for organizing complex systems, ensuring that components interact seamlessly(Schumacher,

2003c). This not only improves the overall robustness of software architectures but also facilitates scalability as systems grow in size and complexity.

Patterns, across various paradigms, have demonstrated their prowess in simplifying maintenance tasks and promoting extensibility. The encapsulation of design decisions within patterns reduces the ripple effects of modifications, making it easier to update and enhance systems. This benefit is particularly pronounced in long-lived software projects, where the cost of maintenance and evolution can be substantial(Izurieta & Bieman, 2013).

The introduction of design patterns has contributed to the standardization of best practices in software engineering (E. Gamma et al., 1993). Patterns encapsulate the distilled wisdom of experienced practitioners, providing a shared vocabulary and set of guidelines(J. M. Smith, 2012). This standardization not only accelerates the learning curve for new developers but also establishes a common ground for communication among team members.

patterns offer a resilient framework for adapting to the rapidly changing technological landscape. As new technologies and paradigms emerge, the fundamental principles embedded in design patterns remain relevant (Mang & Reed, 2012). This adaptability ensures that software systems designed with patterns can withstand the test of time and technological evolution.

In conclusion, the introduction of design patterns in software engineering has engendered a transformative impact, enhancing the robustness, maintainability, and collaborative potential of software systems. Through a systematic encapsulation of best practices and solutions to recurring challenges, design patterns continue to serve as invaluable tools in the pursuit of excellence in software development.

### **2.1.6. Inadequacies of Patterns**

Just like other technologies pattern have some restrictions. The utmost crucial concern posed by skeptics about the usefulness of patterns is one of uniformity. Patterns have no official standards, and writers develop patterns in a variety of styles. They are informal statements of a problem and its remedy. Patterns can become classics, remain restricted to specific sectors, or simply fade and be forgotten, depending on their usefulness. A pattern's success is greatly reliant on its name and the kind of the information it contains (D. Schmidt, 1995). Patterns that are both overly comprehensive and packed with information have a greater chance of



losing sight of the solution's central purpose and essence. Keeping a pattern too short, on the other hand, may degrade the quality of the information that is offered and necessitate the utilization of extra pertinent information sources in order to comprehend the design challenge. (Agerbo & Cornils, 1998).

In projects that need collaborative cooperation, it is essential to provide an introduction to the pattern collection of interest and training to the entire team in order for the patterns to be utilized properly. This is the only way to ensure that the patterns will be used effectively (Unger & Tichy, 2000). This is important to keep in mind if practitioners are attempting to broaden their vocabularies through the use of patterns. It is imperative that technically difficult communications be supplied, even though this could call for additional work to be put into training team members.

The fact that patterns are dependent on the programming language is another argument against them. The vast majority of pattern collections are dependent on a particular programming language, such as C, C++, or Java. Because of this dependence, the applicability of the pattern is occasionally restricted to particular systems alone. Finally, it is possible that applying the pattern to every possible situation is not acceptable, especially when the solution is obvious. There are others who argue that the importance of design patterns is overstated (Cline, 1996). The usage of design patterns may be advantageous, neutral, or detrimental depending on the circumstances of application, as determined by the results of an experiment that compared patterns to straightforward solutions. The experiment was carried out so that the findings could be compared (Prechelt & Unger, 2001) Other people who do not like patterns have said things like, "a pattern is not a strict prescription to be followed (unlike certain modeling standards like UML), but it is more than just a general suggestion" (Vokac & M., 2004). Because UML does not provide step-by-step directions like a recipe, this is difficult to accept because patterns do.

In spite of the challenges mentioned previously, it is nevertheless a fact that patterns have the potential to be helpful for solving more complicated design problems. They do this by offering a method to reuse tested and proven solutions, which helps designers avoid reinventing the wheel. "Pattern interest has expanded in recent years, as indicated by the number of conferences and seminars organized every year across the world since 1994. Extending

outside the United States and Europe, new developments include AsianPLOP 2011 in Tokyo and GuruPLOP 2013 in India. This unique method is still in its early stages, although patterns have been successfully applied in a variety of disciplines recently, including formalization approaches (Taibi, 2007), online application design (Millett, 2010), and mobile application design (Neil, 2012), among others. As a result, it is realistic to expect them to become more important in the next years.

## **2.2. Design Patterns.**

It is noteworthy to note that Alexander's architectural concepts and ideas have recently had a far greater influence on industries other than architecture. This covers a wide range of industries, from organizational management to poetry, but in the context of this study, it focuses specifically on the subject of computer software design. "Chris (Alexander) is a renowned cult figure," Richard Gabriel, a well-known proponent of the software pattern approach, once said. (Eakin, 2003).

The necessity for software reuse within this group has had an impact on the adoption of patterns. Software developers frequently repeat ideas that have previously worked well for them, and as they gain more experience, their library of design knowledge expands and they become more skilled. However, this design reuse is typically limited to personal experience, and developers seldom share design expertise (Beck & Coplien, 1996). It is absurd how we software developers reinvent the wheel with every project, according to (Ganssle, 1992). The introduction of design patterns presented a chance to share the collective experience of the software community and get beyond the inefficiencies and resource waste of re-invention.

Although patterns were primarily used in object-oriented program design at first, they are now used in many other areas of software engineering. Organizational patterns are discovered through examining recurrent linkages and structural relationships within companies that support their performance. The pattern language (Cain & Coplien, 1996) that outlined "best practices" for efficient software development is one example, as are the collection of patterns for implementing novel concepts in an organization (Manns & Rising, 2000). Pedagogical patterns aim to promote effective teaching practices by capturing professional knowledge in the area of teaching and learning. (Bergin, 2000) and are two instances of pedagogical patterns that have been published (Fricke & Volter, 2000). The two distinctive qualities of software—

reliability and human factors—are the focus of patterns for telecommunications systems. A few publications that discuss patterns and pattern languages for usage in fields like telecommunications, distributed systems, middleware, etc. are those by (Adams & Coplien, 1996), (Rising, 1998) and (Hanmer, 2007).

Additionally, patterns have been effectively used in software development (Ambler, 1998), cognition (Gardner & Rush, 1998), interface design (Borchers, 1999), and software configuration management (Berczuk & Appleton, 2003).

The research community has broadly embraced and expanded on (Alexander, 1977) notion of capturing design experience through patterns, particularly in the area of software engineering. Some of the opinions about patterns that various professionals have are presented here.

Patterns appear as a result of the lessons gleaned from the application of a certain discipline. Experts in the subject gather these lessons and add information acquired from a study of the theoretical underpinnings of the domain to them. These specialists may then reshape patterns that can be utilized again in the field. These actions taken together result in the formation of a pattern (Petter et al., 2010). Comparing patterns to other learning strategies like algorithms and heuristics is intriguing. In order to solve a problem, an algorithm uses operations from a predetermined set of fundamental operations (addition, subtraction, multiplication, and division), and it does so in a finite number of operations (Aytug et al., 2003). According to this definition, an algorithm converges when it completes its task in a limited number of steps, meaning it always arrives at the correct solution. An algorithm may be used to a collection of mathematical connections or mathematical assertions that relate to the different parts of a system in order to derive a solution. While design patterns are unrestricted, these linkages essentially convey knowledge of how a certain system function.

Heuristic rules are those that are created via experience, judgment, and intuition. Heuristics do not represent the knowledge of the design, unlike relationships, and are a representation of rules by which a system may be run. Additionally, heuristics may not always provide the best or ideal option (unlike algorithms). Years of experience lead to the development of heuristic rules, which, unlike design patterns, are often far more private and individual and not accessible to the general public.

It is crucial to remember that patterns are meant to enhance rather than diminish the distinctiveness of the design. This is so that a pattern can offer a universal solution to a recurrent issue, one that can be used in various contexts without necessarily repeating itself (Cool & Xie, 2000). It is not the case that patterns are "one size fits all" since the process of adapting or applying the pattern permits modification at various phases during the software development.

Software practitioners picked up on the educational value of patterns, or "learning from experience," quickly. It does so because codifying good design approach aids in distilling and disseminating expertise, assisting others in avoiding development traps and pitfalls that are regularly experienced (Jézéquel et al., 2000).

### **2.3. Anti-patterns**

In 1995, Andrew Koenig introduced the concept of an anti-pattern. Anti-patterns are, as their name suggests, the opposite of patterns. If patterns are the best answers to repeating issues, anti-patterns are the worst solutions to the same recurring problems. Because it might be extremely helpful to know what does not work (and why), Koenig asserted that anti-patterns may be more important than "actual" patterns (Rising, 1998). An alternative definition of an anti-pattern is that it "describes how to get out of a terrible solution and then how to move from there to a good solution" (Appleton, 2000).

According to Coplien (2000) anti-patterns do not provide a resolution force as patterns do, and they are dangerous as teaching tools: good pedagogy builds on positive examples that students can remember, rather than negative examples. Anti-patterns might be good diagnostic tools to understand system problems" (Coplien, 2000). Further: "Anti-patterns do not provide a resolution force as patterns do, and they are dangerous as teaching tools: good pedagogy builds on positive examples that students can remember, rather than negative examples. Anti-patterns might be good diagnostic tools to understand system problems".

### **2.4. Attacks on Networks and Havoc Caused.**

The world today is characterized by security incidents, as evidenced by the increasing number of attacks on world citizens from terrorism, extremism, wars, political intolerance, muggings, and hijackings, to name a few. Networks and systems are not immune to these attacks, just as

citizens are. Phishing, identity theft, worms, viruses, denial of service, social engineering, and botnets are examples of what is wreaking havoc in the world of technology.

In today's world, where the internet and networks are relied on for data communication, transmission, and storage, security is a major concern, and there is an increasing need for networks to be equipped with protection mechanisms against both internal and external attacks (Fernandez & Pan, 2001) based on the insecurity that exists within them (Nishant, 2012).

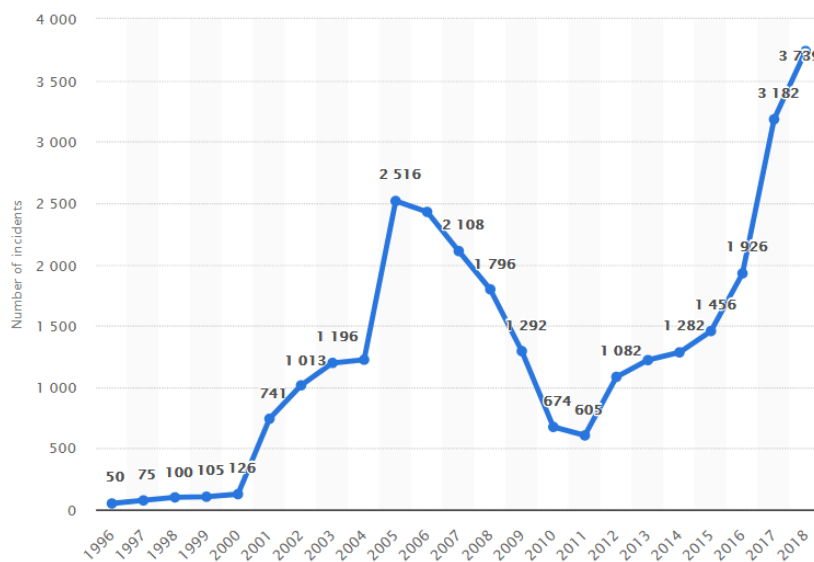
The rapid development of the internet and computing technologies has made society more reliant on network services than ever before (W. Baker et al., 2011). As the growth rate increases and attacks become more complex and sophisticated, enterprises and individuals are exposed to consequences such as civil suits, litigation, loss of productivity, cyber terrorist attacks, data and identity theft, and even reputational damage (Ciampa, 2017a).

Most organizations and businesses today rely on networked systems for competitive advantage, survival, and prosperity. As a result, the information contained within these systems is a valuable asset (Niekerk et al., 2006). Information on credit, health, professional work, business, education, and other important things are relied on within these systems, and because of their value, information within networks has become a growing target of attacks (Fernández, 2015).

Security threats to network systems have evolved dramatically, with news of security breaches, hacking, data disruption, and denial of service hitting the headlines almost weekly (Liebowitz, 2011) Uber, Heathrow Airport, the consulting accounting firm Deloitte, and the financial and credit reporting firm Equifax (Khosrowshahi, 2017) (Warburton, 2017) (Reuters, 2017) (Alferd & M, 2017) are examples of enterprises that experienced security breaches in 2017. Illegal exposure campaigns such as "Vault 7," a series of documents released by Wikileaks in March 2017, exposed the techniques the CIA had been gathering for use in cyber warfare, as well as their capabilities to exploit automobiles, Internet of Things (IoT) devices, personal computers, and smart phones. "Shadow Brokers," an underground group of hackers that emerged in August of 2016, were also notorious for publishing several zero-day exploits and were also responsible for the leaked exploit "EternalBlue," which was largely responsible for the WannaCry Ransomware. All of this clearly demonstrated that

compromises on every sector of the economy, whether government or private, are clearly not immune to attacks (Kevin, 2017).

Despite the fact that protective mechanisms and solutions are in place to ensure data security and to assist organizations in collecting, tracking, and reporting the status of known security issues. Every day, new threats and vulnerabilities are discovered. The continued growth of the number of attack incidents shows that we still have a long way to go and a lot of time and effort is required to reach the appropriate security levels Figure 2.4 shows the number of incidents reported to the CERT (Sas, 2019).



**Figure 2. 4.**Depicts the number of incidence attacks in relation to the year  
Source. (Statista,2019)

In 2018, a total of 3,739 incidents were reported, representing an exponential increase in attacks from 50 incidents in 1996 to 3,739 incidents in 2018. This pattern indicates that the number of cyber security incidents has been steadily increasing.

This raises the critical question of whether the security mechanisms in place to protect the ever-changing networks are sufficient to deter cyber criminals! Are attacks evolving at such a rapid pace that protective mechanisms are unable to keep up? Is there a challenge in the design of defensive mechanisms that they cannot handle providing network protection!

### **2.4.1. Techniques Employed in Attacking Networks**

Critical infrastructure is the target of several attacks using a range of strategies and tactics. This includes hacking into network systems, creating viruses and worms, attacking websites, launching denial-of-service attacks, or carrying out terrorist operations through electronic communications, according to (Bogdanoski, 2013).

The word "hacking" is used to refer to all unlawful access to computer systems and networks, including "cyber murder" and other similar crimes. A British hacker who entered the Liverpool hospital in 1994 and altered the doctor's prescription handed to the patient by the nurse is a classic case of "cybermurder". Many of these hackers employ "brute force," which entails a sequence of combinations of all feasible letters, numbers, and symbols, in an effort to discover the password that will enable them to carry out their malicious attack (Nagpal, 2002).

Attacks against any organization's or country's vital infrastructure may be carried out using the password sniffing approach. Software called a "password sniffer" is used to monitor a network and record passwords that are sent via the network adapter. The attacker has installed this method on several of the network systems that they want to compromise, including the phone system and network providers, in order to monitor all activity on an area network (Hassan et al., 2012). The sniffer program will automatically capture the data that user's input, such as login and password, while utilizing internet access methods like FTP or telnet. In 1994, hundreds of websites were impacted by one attack of this kind.

Sometimes attacks are launched via spam mail. Spam is the term for unsolicited mass message transmission. Although the attackers employ a variety of spam messages, email spam is the most frequently used one. Email spam occasionally poses as an advertisement for goods and services, and when opened by recipients, it instantly generates the recipient's username and password, which the attackers can use to access his or her email and carry out their attack (ITU, 2014).

On occasion, computer viruses are spread across a system in order to carry out undesirable functions including espionage, information generation, or even system failure. In his study, (MacKinnon et al., 2013) found that cyber weapons are primarily software tools used by cyber terrorists to wreak havoc on organizations and even nations in order for them to accomplish

their missions and goals. These software tools can manipulate computers, make intrusions into systems, and perform espionage by sending spying which can inform of viruses into a system.

#### **2.4.2. Havoc of Network Attacks**

It is impossible to overestimate the importance of data in carrying out various tasks inside an organization since it is the main instrument used to transfer data into information when businesses and nations manage it. According to (Koltuksuz, 2013), a network attack can undermine data integrity, causing the data to no longer be trusted, damaging its secrecy, and interfering with its availability. The increased frequency of network attacks into corporations and nations' data has created a slew of issues, including the loss of key and critical data that is typically difficult to recover.

The crucial infrastructure via which much of the company's activities is carried out and which also contains vital information is the pulse of every organization. It is self-evident that any attack on a country's key infrastructures has the potential to halt its economy. According Thuraisingham, (2004), an attack may be undertaken against any of the following targets. For the operation of the country, infrastructure comprises telephone lines, electronic, electricity, gas, reservoirs and water sources, food supply, and other crucial elements. All telephone connections might be shut down as a consequence of an attack on the software used by the telecommunications sector, and network attacks could target the software used by gas and electricity providers.

Attacks on networks have resulted in the closure or immobilization of several industry-focused businesses, which has some effect on the expansion of the global economy. According to Thuraisingham, (2004) estimate, network attacks could cost a company billions of dollars in the business sector. For example, a bank's network could be attacked or hacked, giving thieves access to their accounts without authorization and costing them millions or even billions of dollars. This would force the bank into bankruptcy and force it to close down.

Numerous innocent people have been killed by cyberterrorism, which has also caused financial hardship for many families and, occasionally, psychological harm to the afflicted individuals. According to Awan,( 2014), cyber-terrorism has taken the shape of attacks on computer systems, networks, and attacks that have led to "explosions of multiple plane crashes crises all over the world that have claimed many lives." These attacks have caused major



damage and the loss of life. It is extremely upsetting that cyber terrorism is taking lives at an exponential pace, and if prompt action is not taken to stop it, it will continue to take the lives of innocent people who ought to be advancing the world economy. Terrorists may even try to manipulate air traffic control by breaking in, which might lead to jet accidents or perhaps a deadly crash (Iqbal, 2004). Terrorists may potentially take control of a company's pharmaceutical computer system, changing some of the important prescriptions that the pharmacist may have created.

As trust may be seen as a tool that builds connections and confidence between companies and customers, it is a well-known fact that the growth and patronage of any business depend on the faith that its consumers have in such organization. Attacks on networks may intrude into other people's "cyberspace," claim (Saini & Rao, 2012), with the intention of discouraging and upsetting end users and customers who often visit the impacted page for commercial transactions. Customers will lose trust in the aforementioned internet site if it is invaded and attacked since it damages companies and makes users the victims of cyberterrorists. It is impossible to overestimate the value and significance of trust in the information technology era because without it, businesses and organizations would not be able to survive, much less thrive, in a cutthroat environment.

## **2.5. The OSI Model and Its Roles in Securing Networks**

Understanding the OSI model aids network administrators in understanding IT security which evolving and expanding daily. Threats to an organization can range from a misplaced backdoor to a nefarious adversary meticulously creating a Trojan to open ports on your web server (Shaw, 2022).

By examining the layers, we may identify the benefits and drawbacks of our networks. Despite having a fantastic set of antivirus software, our encryption techniques being up to dated. Knowing that a certain layer is weak helps us to comprehend how vulnerable our system is, how to split resources, and how to get specialist support if needed (Surman, 2002).

The most crucial point to keep in mind is that every layer has attacks either in development or ready to launch due to lax defense. Regardless of the layer, protecting your system from attackers is a never-ending process (Holl, 2003).

We can better comprehend the dangers that our networks may encounter by understanding the OSI model (Osakwe, 2020). Understanding the OSI model's segregated structure aids in our understanding of the overall but compartmentalized approach that must be used. The tiers can only be adequately protected if our networks are seen as independent parts. If the network can be divided into manageable components, as the OSI model enables us to do, we can share the risk. We have a better chance of addressing problems and protecting our assets when the risk is divided into smaller, more controllable parts (Q. Zhu & Basar, 2012).

Also, as technology advances, so do high-profile hacking tactics. As a result, security specialists are in high demand. However, security experts and analysts must first grasp the fundamentals of how network levels function, as well as the important components that may strengthen security at each tier (Bourgeois & Bourgeois, 2014).

The OSI model is a good place to start when it comes to learning about network security. Every layer of the framework is based on a particular protocol or method. Conceptually, the network architecture is divided into seven tiers comprising of Physical layer, Data link layer, Network layer, Transport layer, Session layer, Presentation layer and Application layer (Swire, 2018).

The effort of protecting a single computer, much alone a network, might be intimidating to the inexperienced. IT system managers must begin someplace, and that place should be with a grasp of the OSI model. The OSI model separates the network into easily discernible parts that may each be protected independently. Once every component is secure, a comprehensive security strategy is completed, and the threat of an attack is significantly reduced. But first, it is important to understand the OSI model.

### **2.5.1. The OSI Model.**

The International Organization for Standardization (ISO) developed the seven-layer networking architecture known as Open System Interconnection to be used for worldwide communications (Beal, 2021). It is important to note that the model is an ISO standard that specifies how the IT industry should develop networking protocols for computers. Protocols are sometimes referred to as languages, and these languages allow devices to communicate with one another. When everyone follows the same set of guidelines, putting the puzzle together becomes much easier (Jasud, 2017).

These layers are divided or rather modularized into two depending on the operational level within a network setup. These modules are referred to as the host and media Layers as shown in Figure 2.5 The Host layer combines the Application Presentation, Session and transport layer, while the media layer includes the network data link and presentation layers(Kavianpour & Anderson, 2017).

The media layers focus on the preparation, encoding, and transmission of data across the network. They are just concerned with the data transfer itself and not with the content or purpose of the data. They have hardware and software counterparts, with the former predominating as one moves from layer 1 to layer 4.

The host layers, on the other hand, are responsible for implementing networked applications and interfacing with the user. The higher-level protocols care less about the mechanics of data transport and instead rely on the lower levels to make it happen. Typically, these layers are realized as computer programs that operate on some sort of electronic device. Layers, Sublayers, and Layer Groups in the Open Systems Interconnection Reference Model (OSI, 2005).

In a hierarchical structure like this one, each layer handles its own specialized function and passes the results on to the layer above or below it for further processing. This means that data is sent from a higher level to a lower one. The data is directed to its final destination when it reaches the physical layer. When information reaches its final destination, it must first pass through several layers before it can be translated. An e-mail, for instance, starts at the Application layer and travels down the stack, across the wire, and back up the stack to the Application layer at the destination.

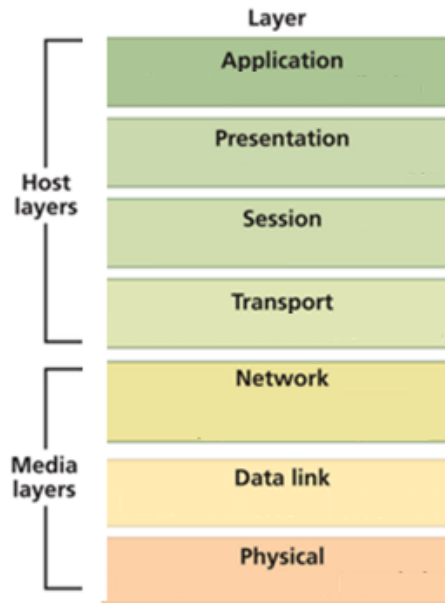


Figure 2. 5.Showing the media and host layersSource: <http://www.cables-solutions.com/wp-content/uploads/2016/06/OSI-Model.png>

Control is transmitted from one layer to the next within the source computer. Data flows down the hierarchy of the sending node and then up the hierarchy of the receiving node. Figure 2.6 depicts this information flow; observe how there is no way to skip a layer and how the procedure is mirrored on the following machine. Each layer is only capable of communicating with the layers above and below it as seen in Figure 2.6, the Physical Layer is capable of communicating with the Data Link Layer and the media itself note that there is no lower layer for Physical.

Created one layer at a time. This allows for scalability, as work in one layer can continue even if progress in a different layer is slowed. The process by which one layer of a network may connect with the proper layer at a remote location is called "encapsulation," and it occurs when data travels from one layer to the next.

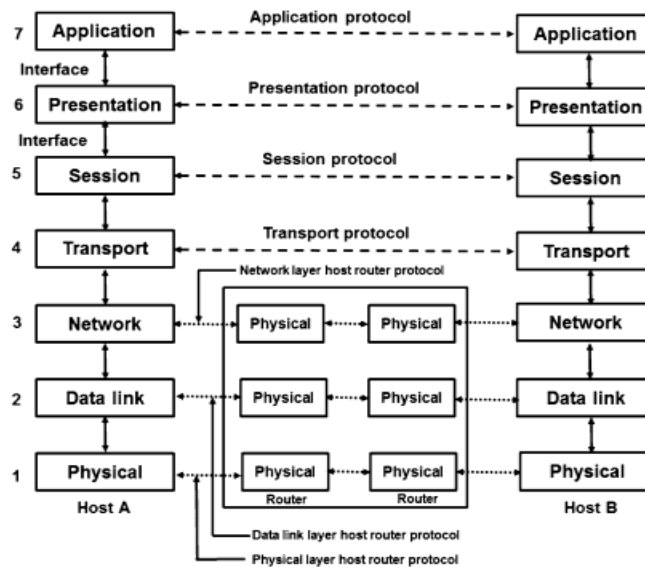


Figure 2. 6.The OSI Layers Data Flow  
Source (Y. Li et al., 2011)

The layers and their functions are described as follows:

- Physical Layer • Connect and disconnect connections, specify voltage and data rates, transform data bits into electrical signals, and choose between simplex, half-duplex, and full duplex transmission.
- Data link Layer • Waits for response for each sent frame while synchronizing, detecting, and fixing errors.
- Network Layer • Routes important signals, Separate incoming messages into packets. Serve as a network controller for data routing.
- Transport Layer - Determines if transmission should be parallel or single-path, and then determines if data should be multiplexed, divided, or segmented. Splits data down into smaller parts for effective handling.
- Session Layer –controls logging on and off, user authentication, billing, and session management between two systems in synchronized interaction.
- Presentation Layer – is in charge of the syntax and semantics of the information sent.
- The application layer- in charge of information file retransmission, login assistance, password verification, etc.

The model, together with examples of security risks for each of the model's levels, is depicted in Fig. 2.7. The OSI model's lowest levels are implemented in hardware and are thus vulnerable to physical attacks. The remaining layers are comprised of software and are thus only directly endangered by software risks.

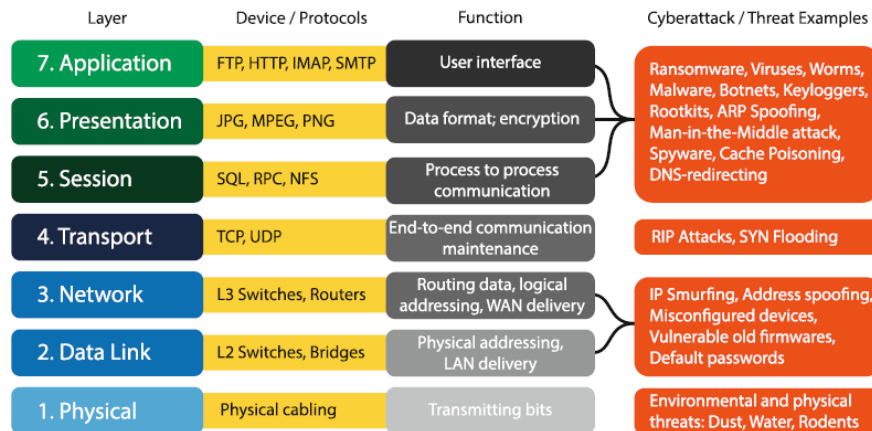


Figure 2. 7.OSI Models and Attacks. (Manninen, 2018)

## 2.5.2. Necessity of Having Security at Each Layer of the OSI Model.

OSI Defense in Depth to Increase Network Security explains how susceptible networks are and how Network administrators may lower risks (Dauch et al., 2009). Each layer of security is addressed via the OSI model technique.

## 2.5.3. OSI Layers Threats

### 2.5.3.1 Physical Layer Security threats

As earlier said the Physical Layer is used to provide the technical requirements for data communication it describes the network's physical characteristics, such as voltage levels, cable kinds, and interface pins. The bulk of risks are at this layer, include cutting physical cables, changing interface pins, natural disasters like earthquakes, fires, and floods that can cause short circuits, as well as other acts of human vandalism, can interfere with the electrical impulses that connect network nodes. Some of the threats/vulnerabilities in relation to access control include Access of the network to non-authorized personnel, inadequate physical security enabling unauthorized access, and unrestricted access to critical servers and lack of enforcement of password complexity policies(Shakiba-Herfeh et al., 2021).

### **2.5.3.2. Data Link Layer Security Threats (Switch Security)**

The second tier of the OSI model is called the data link layer, and it deals with how frames are delivered. It guarantees data transfer across a physical link in a secure manner. This layer is in charge of flow management, error notification, ordered frame delivery, network topology, and physical rather than logical addressing. Switches that handle protocols like the Spanning Tree Protocol (STP) and the Dynamic Host Configuration Protocol are frequently found on this tier (DHCP). LAN connectivity is provided by switches, and most attacks start within the LAN. Because of the layer's practical and useful design, there are flaws in the Data Link Layer. Frame-level exploits and vulnerabilities include, but are not limited to, sniffing, spoofing, broadcast storms, and unsafe or nonexistent virtual LANs. A network segment or the entire network may have significant issues due to improperly configured or defective network interface cards (NICs) (Mahmood et al., 2020).

ARP spoofing may be employed maliciously to take over the IP address of a system. Through the use of ARP spoofing, it is possible to force a switch to route traffic to a different VLAN by delivering ARP packets with carefully fabricated IDs. Without the top tier being aware of it, the security flaw at the lower layer affects security at the upper layer (Al Sukkar et al., 2016).

MAC flooding is the network switch attack that happens when a switch's MAC table reaches its maximum of 131,052 entries and then overflows. An attacker is able to collect network-sensitive information, such as passwords, by sniffing the inundated traffic (Daş et al., 2015).

A spanning tree protocol attack is A denial-of-service (DoS) attack that occurs when an attacker injects himself into a data stream. It starts with a physical intrusion, typically by a malicious person installing a rogue switch. The root priority is lowered by the attacker when a lower root priority is set, the link between two switches is severed. This makes the attacker's switch the "root" switch, giving the attacker complete access to all traffic going through all switches (Pearson, 2016).

A multicast brute-force attack looks for software flaws in the switch. By storming a switch with multicast packets, the attacker attempts to exploit every conceivable weakness. The objective is to determine whether a switch that receives a high quantity of layer 2 multicast traffic will "behave improperly." When routing joins different VLANs, the switch should

maintain traffic inside the original VLAN; but, if it does not, frames may leak into other VLANs. This attack is very speculative since it expects the switch to handle multicast packets improperly. To stop such an attack, the switch should contain each frame inside its appropriate broadcast domain. Switches, on the other hand, have typically been unable to counteract this kind of attack, adding it as another attack channel (Alzahrani et al., 2013).

Random frame-stress attacks come in a variety of flavors, but in general, they are brute-force attacks that randomly alter various fields of a packet while leaving the source and destination addresses alone. The objective of this attack is usually to determine how the switch software responds to packet fields that contain nonsensical or unexpected values(Zhao et al., 2011).

### **2.5.3.3. Network Layer Security Threats (Router Security)**

The OSI model's Network layer defines routing, layer 3 switching, and IP addressing. This layer is used to communicate between network devices that are not connected to the same segment. The network layer serves as a guide and a traffic controller. It directs the flow of all data packets. When data enters the network layer, it is assigned an internet protocol (IP) address as a result the packet now becomes aware of its destination. The routers, on the other hand, are responsible for keeping track of and managing all of the traffic. The network layer performs routing on the network by utilizing many widely used protocols (Kumar et al., 2014).

Malicious actors can attack the network layer by overloading the network, impersonating the network, and sniffing the network traffic(Kaur & Singh, 2014). In relation to network overloading and congestion a denial of service (DoS) attack, such as a ping flood, can be used to do this by an attacker. When an attacker knows which IP addresses are affiliated with a target network, he or she will continually send an internet control message protocol (ICMP) ping or echo to overburden a portion of or the entire network, depending on the situation. This means that an attacker can target a single endpoint or a router in order to prevent all communication from taking place(Kumar et al., 2012).

IP spoofing is yet another method of attack to be aware of. An attacker will alter the source IP in the header of the message, which is commonly used for distributed denial of service (DDoS) attacks. IP spoofing has become almost normal practice for DDoS malware kits in recent years(Hastings & McLean, 1996).



With the use of IP and port sniffing, attackers can have an impact on the network layer. An attacker can employ IP and port sniffing to undertake reconnaissance and learn more about a user by analyzing the packets they send and receive. Malicious actors can steal vital information from a network connection that has not been secured(Anu & Vimala, 2017).

Routing attacks' most visible consequence is network outage, in which attackers drop packets and render destinations unavailable. This form of attack, in which traffic is "black-holed, (malicious node pretends like normal node and forward packets but selectively drops some packets) " is also referred to as a hijack attack. The attacker's objectives, on the other hand, may be more complex like Surveillance and Impersonation (Korba et al., 2013).

Government agencies may use routing attacks to perform surveillance and Intelligence agencies such as NSA have had their fair share of accusation for launching routing attacks to intercept and reroute traffic of interest for purposed of surveillance. This attack is typically referred to as an interception attack, as genuine destinations continue to receive traffic. Interception attacks are far more difficult to detect than hijack attacks since they do not disrupt communication, albeit performance may suffer as a result of the more convoluted methods. Additionally, authorities might use routing attacks to circumvent legal constraints by redirecting domestic traffic, such as emails between citizens and foreigners for monitoring purposes(Goldberg, 2017).

By intercepting packets via hijack or interception attacks and responding with forged responses, attackers can fool senders. These attacks have the potential to cause serious harm. In 2018, attackers impersonated Amazon's authoritative DNS service and responded to DNS queries for a bitcoin website with Russian IP addresses via routing attacks. The consumers were then sent to a bogus website that they mistook for their legitimate bitcoin provider. As a result, bitcoins was stolen.In order to send spam or other malicious traffic, attackers may spoof a huge number of IP addresses (Madory, 2018).

#### **2.5.3.4. Transport Layer Threats.**

Transmitting variable-length data sequences between source and host is defined by the Transport Layer. Because data comes in various sizes and is divided into packets, there are guidelines on how to handle it, the reliability of this layer can be achieved by ensuring the segmentation and de-segmentation mechanism and error control. Two protocols are synonymous with this layer: Transmission control protocol (TCP) and User Datagram

Protocol (UDP) .TCP favors data quality over speed while UDP favors speed above quality(Stewart & Metz, 2001).

Despite its status as a 'host layer,' the transport layer is vulnerable to the same risks that plagued the previous 'media layers.' Sniffing, especially relating to ports and protocols, may be found here as well (Kizza et al., 2013). The transport layer can be targeted by distributed denial of service (DDoS) attacks(Manavi, 2018).

At the transport layer, SYN floods and Smurf are two common attack types. By simply abusing the TCP three-way handshake, SYN flood, also known as Half Open Attack or TCP Sync Flood, happens when an attacker uses a fictitious IP address to initiate several connections to a server without waiting for the connection to finish (Parmar & Gosai, 2015). Smurf attacks to overburden network resources by causing a denial of service. The attacker sends out echoes of the Internet Control Message Protocol, causing an unending cycle of requests (Bhalekar & Shaikh, 2019).

Additionally, there is a problem to address at the transport layer level of abstraction: For a malevolent actor, it is an excellent site to carry out reconnaissance. Except if they aim to launch a distributed denial of service (DDoS) attack against you, they are not permitted to attack you directly at the transport layer. The actor, on the other hand, may learn a great deal about how to gain access to your environment, particularly the session layer, which is frequently targeted by hackers and other malicious actors(Elejla et al., 2017).

The importance of reliability at the transport layer cannot be overstated. Because all of the packets are moving around, there is a great deal going on in this layer. If this layer does not successfully segment and reassemble the packets, performance may be adversely affected (Hunt, 2002). As a result, the transport layer must be as error-free as is reasonably practicable. This is also the reason why it conducts error control functions. If there are any flaws in this section, interaction between hosts can get jumbled(Algaley & Yousif, 2022).

#### **2.5.3.5. Session Layer Threats.**

The Session Layer is a gatekeeper responsible for syncing everything up for action, it creates, manages, accepts, opens and closes sessions, inter-system communications and the interaction of local and distant applications. At times it is even responsible for sessions failing on

occasion, especially if your machine is managing a large number of them. As a result, not only is efficiency critical at the session layer, but so is security(Köksal & Tekinerdogan, 2019).

Managing the session layer is crucial. Session hijacking attacks occur in the session layer, they include Cross-site scripting, sidejacking, fixation, cookie stealing, Man in the Middle attack and brute force attacks. A session hijacking attack is one that compromises a token by estimating what a legitimate token session will be, allowing the attacker to gain unauthorized access to a server as a result of the breach to authenticate the session, the server places a temporary remote cookie in the client's browser. So, the remote server remembers the client's login status. A hacker requires the client's session token to perform session hijacking. This may be gained by fooling the user into visiting a malicious link with a preset session ID, for example. Using the stolen session token in their own browser session, the attacker can take over the targeted session. The server eventually thinks the attacker's connection is the same as the original user(Shi et al., 2021).

#### **2.5.3.6. Presentation Layer Threats.**

The presentation layer transforms machine-readable code into something that the end user may utilize in the application layer afterwards. It is where formatting, conversion and encryption happen. Threat actors hunt for attacks in presentation layer encryption weaknesses. SSL hijacking or sniffer is one of the most popular techniques. Malformed SSL requests are the most common type of attack seen at the presentation layer. Because attackers are aware that analyzing SSL encrypted packets consumes a lot of resources, they utilize SSL to tunnel HTTP attacks to the target site(Shi et al., 2021).

Man-in-the-middle (MitM) attack, as mentioned earlier, are a favorite tactic among threat actors. SSL hijacking may be harmful at the presentation layer when combined with malware. If an attacker has already placed malware on a system, the MitM will employ a proxy to act as an untrusted certificate authority. If this occurs, the browser will trust the incorrect certificate authority, allowing the attacker to access all communications(Keerthi, 2016).

#### **2.5.3.7. Application Layer Threats.**

The application layer, as the name implies, is intended to service the end user. Mail and file transfers, among other things, take place here. It includes online browsers, applications, and nearly anything else you see on your screen. To be clear, applications are not necessarily a

component of this layer, but the services they provide are. This layer allows for the most diverse cyber attacks and security breaches. It can result in the network being shut down, data being stolen, the program being crashed, information being manipulated as it travels from source to destination, and many other things. The list of threat can be exhaustive starting from the different types of malware, because all viruses, worms, key loggers Phishing, Backdoors, Program logic, flaws Bugs and Trojans do their damage to this part of the OSI model (Norman & Joseph, 2017).

As shown in Figure 2.8. network managers may learn that network security is more than simply OS hardening, authentication, and encryption with the aid of the OSI model and Network Security by Defense. Because security flaws exist at every level of the OSI model, precautionary security measures can be taken to protect networks (Groat et al., 2012). Three fundamental building components comprise the foundation for network security protection and attack is protecting Confidentiality, Integrity, and Availability (CIA).

The choice to use the Open Systems Interconnection (OSI) model in this research is justified by its comprehensive framework, layered approach to security, industry acceptance, interoperability, sensitivity to security concerns, educational value, relevance in modern networks, and facilitation of pattern development.

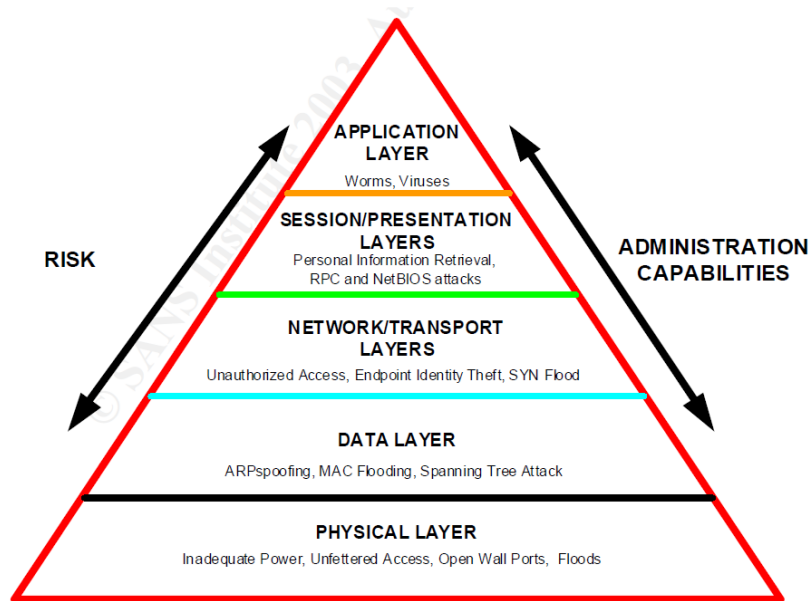


Figure 2. 8.OSI Addressing Security at Each Layer. (Holl, 2003)

The OSI model's structured layers provide a systematic approach to understanding network architecture and addressing security concerns at specific levels. Its global recognition as an industry standard ensures that research findings are applicable across diverse network environments. Additionally, the OSI model's adaptability to modern networks and alignment with network simulation enhances its relevance in the context of enhancing network security through defensive pattern models.

## **2.6. Network Security Architecture Role in Securing Networks.**

Strength of the organization's structure is a key component of its success. A well-thought-out company plan, well-trained workers, and seasoned business leaders are all essential. Consistency and commitment are essential to building a powerful team (Eskierka, 2011), which is why Network security architecture implementation is no exception. In order to keep an organization safe from outside attacks, it is critical that its network security architecture is impenetrable.

Network attacks and breaches happen in a variety of shapes and sizes, and they are always evolving. As a result, it is imperative for a business to be well-versed in measures and methods to counteract such dangers. An organization is subject to a variety of dangers if it lacks adequate security(Smys et al., 2020).

The organizational structure, standards, rules, and operational behavior of a computer network's security and networking components are all laid out in a framework known as a network security architecture (Smys et al., 2020).With the aid of a security architecture, you can better understand the relationship between your security controls and your entire systems structure. In order to keep your vital system's quality traits such as confidentiality integrity and availability intact, these controls are essential to their implementation(Schumacher et al., 2013b).

Three critical elements that are included in a security architecture are Network Elements which entails Network topologies, Network connections between nodes using specific protocols, Network communication protocols and Network nodes. Security Elements which entail strong encryption techniques, Secure network communication protocols, security software and security devices. Security Frameworks which entail Technology standards for

security software choices and security framework architecture standards. Lastly Standards, Security Procedures & Policies (Biskup, 2009).

The purpose of network security architecture is to provide assurance that a company's critical network infrastructure, including its most sensitive data and mission-critical applications, is secure from both known and unknown threats. To prevent, detect, and neutralize any and all network intrusions. Take care to maintain small, unnoticeable attack surfaces in your network, so that malicious actors may sneak up on their targets without raising suspicion. To ensure that all private and sensitive information is sent using strong encryption and end-to-end encryption methods, and that any attacks are actively recognized, mitigated, and destroyed using countermeasures like Moving-Target Defenses. In order to deliver a suitable and timely solution, you must have a thorough understanding of the many problems in your system (Checkpoint, 2021).

## **2.7. Network Security Design.**

Network security is built using a variety of components and technologies, and these components and technologies are represented in architectural design. In order to build the system, the designer will select the relevant technologies, including Blackbox firewall products, cryptographic protocols, proxy agents and packet filtering routers. As a result, a detailed architectural model will depict all of the system's parts and their interrelationships. The Architectural Model not only aids in a better knowledge of the system to be implemented, but it also allows for the establishment of important components and the effects on the overall system of errors in each of these components(Alabady, 2009).

It has been thoroughly explored and proven to be practical to use architectural design for software applications. The significance of architectural design is expanding as commercially available (COTS) components have grown more prevalent. The architecture of the system becomes crucially significant, much like in software development, because a network security system depends on a range of COTS that implement a variety of security technologies (Miller, 2013).

There are also survivability studies that focus on the same topic. Modeling a system's architecture and assessing the consequences of component failures on the overall aim may be accomplished using approaches like the easel language (Stojkovic & Steele, 2005).Although

the suggested security architectural design has some similarities to the survivability architecture approach, they also diverge dramatically. "Survivability is concerned primarily with system availability and goal fulfillment," according Stojkovic & Steele, (2005), whereas, as previously indicated, network security architecture is concerned primarily with authenticity and integrity, confidentiality, access control, and auditing needs. These and other research can provide some insight into the qualities of an effective network security architecture design model.

The designed model has a very desired attribute of formality. Studies over an extended period of time have shown the benefits of a formal model over an informal or semiformal one. Additionally, the model is assessed automatically rather than by a human, and it enables code to be immediately generated from the model (Rushby, 2001).

While creating a new network security architectural model, there are many lessons that may be learned from formal techniques. The ability of the model to represent lower-level subsystems that have not yet been built as a black box component in a high-level model is one characteristic that may be incorporated from formal modeling methodologies. Then, using a top-down development strategy, each of these subsystems might be developed separately. Starting with the stated lower-level black boxes, lower-level subsystems may be created and then used to construct a more complicated model of higher-level functionality. By using hierarchical breakdown, the model's scalability and understanding are both enhanced (Graft et al., 1990).

Network security experts have suggested that black-box packet filtering and proxy agent black-boxes (which may be further extended by using protocol-specific programs) be used to emulate firewalls. An advanced concept of a network security system could include a firewall black-box in addition to additional aspects like decentralized trust management and cryptographic associations management (Sena & Geus, 2002).

The layered technique, which divides the model into various tiers in accordance with certain requirements, as in the design of an operating system, for instance, is a desired characteristic that may be added to existing formal models. The standard in our case may be how sensitive the data is that has to be secured. This is a common approach to information security that is

present in many systems, including the Bell and LaPadula multi-level security model (Bell & LaPadula, 1975) , and it offers protection for crucial assets in the lower levels.

## **2.8. Frameworks Guiding Security of Networks.**

### **2.8.1. NIST Security Framework**

The National Institute of Standards and Technology (NIST) Security Framework is a collection of recommendations created by the NIST. This framework was developed by considering a range of security experts from diverse sectors and setting up a common set of rules and regulations that was then transformed into a framework. It is not a standard checklist that one would check off and mark as finished at the conclusion of each project stage. Instead, it focuses on assessing the current situation. How would you assess security? How to Approach Risk How do you handle threats to security? The framework is more interconnected and helps individuals make wise judgments. It also helps the team communicate about safety precautions, threats, and other topics. This framework focuses on managing cyber-securely, communicating internally and externally, upgrading and updating security rules, and other related things. The five fundamental elements that go into developing this framework are Identify, Protect, Detect, Respond, and Recover (Cockcroft, 2020).

### **2.8.2. COBIT Security Framework**

Control Objectives for Information and Related Technologies is referred to as COBIT. The management, governance, and security of information technology are all incorporated into this security framework together with strong business concepts. Information Systems Audit and Control Association (ISACA), created it. It is a worldwide organization of professionals that focus on information technology security governance. Businesses who want to improve the quality and security of their output might benefit from this approach. This framework is founded on two elements: the requirement to meet stakeholder expectations and the enterprise's end-to-end process control (Arora, 2010).

### **2.8.3. ISO/IEC Standards framework**

ISO/IEC Standards: This framework was created by the International Electrotechnical Commission (IEC) and the International Standards Organization (ISO) (IEC). British Standard BS 7799 served as the foundation for this framework's creation, however it has subsequently undergone several updates and modifications to become ISO/IEC 27001: 2013.It



is an industry-recognized set of best practices for high-level security management and execution. The framework urges companies to assess every aspect of the cybersecurity process, which includes the following: Environmental and physical security, access management and control, information technology security procedures, communication security, cryptography, incident response, and compliance. It contains suggestions for a vast array of security measures that might be used inside the businesses that will be dealt with utilizing this framework. In the course of risk assessment and management, all issues will be addressed (Disterer, 2013).

#### **2.8.4. HITRUST Cybersecurity Framework (CSF).**

To help healthcare organizations and their business partners comply with Health Insurance Portability and Accountability Act (HIPAA) regulations, HITRUST offers an integrated risk and compliance solution. To create a set of safeguards for the security and privacy of protected health information (PHI) and electronic PHI, leaders in the public and commercial sectors in the fields of privacy, information security, and risk management worked together (ePHI). Each of the 156 control requirements in the HITRUST CSF's 49 control objectives fits under one of the following 14 control categories: Access control, human resource security, risk management, and security policy are all included in the program for information security and management. Organization, compliance, and asset management for information security managing communications and operations, protecting the physical and natural environment, Information system acquisition, development, and maintenance Management of business continuity incidents, management of information security incidents, and confidentiality standards (Donaldson et al., 2015) (Donaldson et al., 2015)

#### **2.8.5. Internet of Things (IoT) Cybersecurity Alliance (IOTCA).**

The IoTCA's goal is to create a community of cybersecurity and IoT experts that can cooperate on practical IoT security issues and work toward developing an IoT posture that prioritizes security. Their technology uses a multi-layered approach to offer end-to-end security, covering all networked devices and the apps that run on them. The endpoint layer, which consists of gadgets and linked objects, and the short-range network layer make up the framework. Applications are handled by the data/application layer, while communications are handled by the network layer. Threats include limited resources, malware, device cloning, a

lack of monitoring, protocol manipulation, man-in-the-middle attacks, denial-of-service attacks, and unauthorized software and access are what they aim to stop (Walker, 2021).

### **2.8.6. MITRE ATT&CK.**

MITRE is a nonprofit government-funded research and development organization with a specialization in cyber security. After MITRE began documenting common cyber-attack TTPs against Windows commercial networks, ATT&CK became the de facto standard, providing a consistent lexicon for both offensive and defensive researchers. The Common Vulnerabilities and Exposures (CVE) list is created and trademarked by the MITRE Corporation. MITRE Enterprise has compiled a list of 14 techniques often used by cybercriminals to develop advanced persistent threats (APTs) in a corporate setting. Which include: reconnaissance; resource generation; initial access; execution; persistence; privilege escalation; defense evasion; credential access; discovery; lateral movement; collection; command and control; and exfiltration (Strom et al., 2018).

## **2.9. Threat Model and Threat Modeling.**

A threat model is a representation that describes the threats that affect the security of a given setting, they are developed to standardize and ease disaster planning and preparedness process. Threat models are derived from threat assessment processes (Marback et al., 2013a).

Threat modeling is a risk-based method to building secure systems. It is centered on detecting dangers in order to build mitigation methods. According to NIST, it is used to simulate both the offensive and defensive methods of a particular system in order to aid in the discovery of security solutions (Almubairik & Wills, 2016).

Threat modelling employ the use of abstraction in the quest of discovering security challenges. Abstractions assists in looking at security risk towards a system from a bigger point of view and finding the issues that other procedures and tools could not discover as a result of unique issues that are specific to design of systems. It also assists in discovering parallels and finding comparisons to problems encountered in other systems (Möckel & Abdallah, 2010).

Threat modelling has evolved from traditionally being used in software security to more complex systems such as to evaluation of security process in embedded systems, cloud computing infrastructure as well as control *systems* architecture. Threat modelling produces

threat scenarios which are attributed to threat sources, these scenarios can be characterized using tree structure, graphically or even verbally (Bodeau et al., 2018a)

From literature several diverse threat modeling frameworks, tools and methodologies do exist, they can be famed according to the goal of the threat modelling, the entity being modeled within a system setup or even a specific lifecycle phase of an entity. Some of them are broader than others, some are specific to a particular domain, while others have higher abstractions levels(Gonzalez, 2022).

Threat modeling methods can be incorporated with other methods to create enhanced toolsets for solving security problems for instance several research work has shown its cooperation with the risk process a to solve security problems (Mohanakrishnan, 2021).

A generic threat modeling method should be made up of the following basic phases, the first one being an in-depth understanding of the object to be modeled, second, what the object does, followed by how data flows and stored and finally who the users are(Tarandach & Coles, 2020).

Several threat modelling approaches have been proposed by different authors on how to approach threat modelling exercises. (Shosctack,2014) proposed a threat modeling approach by looking at the assets and impact of threats towards the assets, from this approach critical valuable Assets need be identified and considered one at a time and impact they face as a result of threat being actualized should also identified and prioritized. The supporting assets are also identified and highlighted since they can be exploited as a conduit to harm the critical assets.

(NIST, 2012) in their approach they look at threat modelling in relation to an attack(er) in relation to what they want to achieve and how they achieve it. Several constructs/characteristics of an attacker are modeled in this approach they include their intentions, capabilities, resources available to them and behaviors, all this makes it possible to understand and give insights of procedures, tactics and techniques of attackers. The authors propose the use of cyber kill model or cyber-attack life cycles to model attacker's behaviors into attack scenarios or threat scenarios. The treat scenarios can be described in details after the threat source have been identified and impact can be derived from the attacker's behavior, intentions and motivations.

(Souppaya & Scarfone 2016), their threat modeling approach looks at the software, system and data. The threat modelling at the software part focuses on the reduction of vulnerabilities within the software and it is performed during the design and development stage, the system part focusses on a working and operational system to improve its overall security and the data part concentrates on the protection of the data within the system. The general focus of this modelling approach is the type of the system, its function, what can go wrong and what can be used to cause harm. This modelling approach employs DFD`s to model the system, data and boundaries then threats relevant to each of them are determined.

Scarfone & Souppaya 2016 their threat modeling is data centric in approach i.e. the main focus is on data as opposed to other entities within a system. From this modelling the emphasis is on the identification and characterization of the data of interest within a system and its respective characteristics such as authorized inputs, processing, transmission, data flows, outputs, and access and security objectives of data in place within the system.

### **2.9.1. Threat Modeling Approaches.**

Using threat modelling techniques, a system abstraction, and profiles of prospective attackers, including their aims and tactics, and a library of possible threats are created. There are a wide variety of threat modelling techniques that have been utilized. Some emphasize on abstraction and foster granularity, while others are more people-centric. Some strategies focus on risk or privacy concerns. Combining threat modelling methodologies can provide a more comprehensive perspective of prospective threats (Konev et al., 2022).

To get the most out of threat modeling, it should be done early in the development cycle. This implies that possible problems may be identified and addressed early on, avoiding a far more expensive correction later on. Consideration of security needs through threat modeling can lead to proactive design decisions that allow threats to be mitigated from the start (Poston, n.d.).

When it comes to threat modelling, as with any other aspect of a project, there is no one "best" threat modelling approach; rather, the selection should be made in light of the project's unique objectives and concerns. The following Threat modelling approaches are drawn from many sources and focus on several steps (Marback et al., 2013b).

### 2.9.1.1. LINDDUN

A threat modeling technique called LINDDUN (Linkability, Identifiability, Non-Repudiation, Detectability, Disclosure of Information, Unawareness, and Non-Compliance) helps to systematically identify and address privacy vulnerabilities in systems (Wuyts, 2015). It was influenced by Microsoft's security development lifecycle's threat modelling approach for security (STRIDE) (Howard & Lipner, 2006), which was developed more than 20 years ago (Kohnfelder & Garg, 1999).

One of the most cutting-edge methods for simulating privacy issues is LINDDUN. There are many different security threat modeling approaches (UcedaVelez & Morana, 2015), however they all adhere to the same four high-level phases described by (Shostack, 2014a) as the following four inquiries: What possibly could go wrong? What are your strategies to deal with it, and how well did you do?.

The LINDDUN framework, shown in Figure 2.9, offers a methodical approach to privacy assessment. The process starts with a DFD of the system, which provides a broad description of the system's data flows, data storage, procedures, and external entities. By iterating over all model components and evaluating them from the standpoint of the threat categories, users of LINDDUN may construct threat trees and evaluate if a threat is applicable to the system.

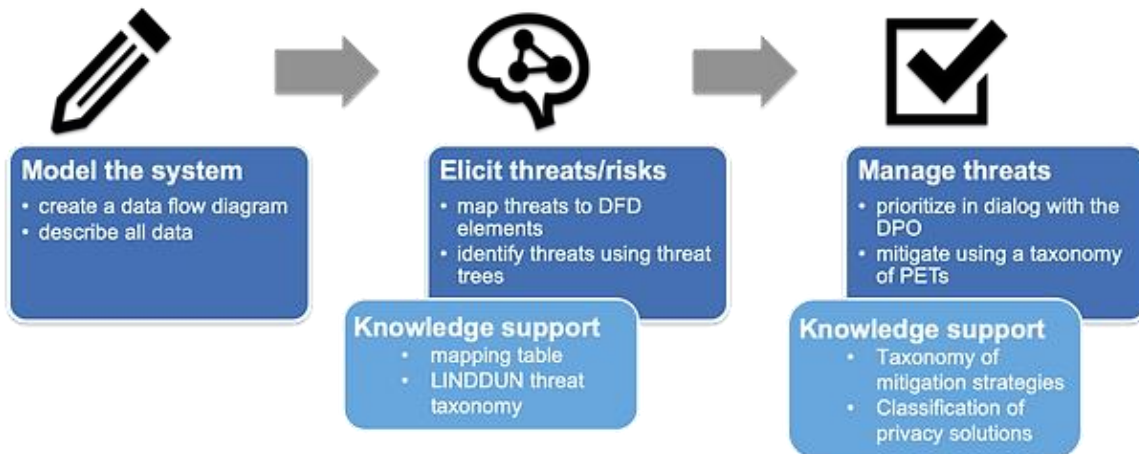


Figure 2. 9.Linddun Phases.  
Source (linddun, 2020).

Essentially, phases 2 and 3 are a series of questions designed to help the user begin the process of detecting potential security risks in the system. Phase 2 is all about mapping danger categories to the portions of the system in which they may arise. The rest of the process focuses on finding solutions and mitigating techniques (Wuyts et al., 2018).

LINDDUN has gained a lot of interest from both academics and the business world. (Y. S. Martin & Kung, 2018) have used LINDDUN in a number of academic projects, and it has been endorsed by several experts in the field of privacy engineering (Kostova et al., 2020)

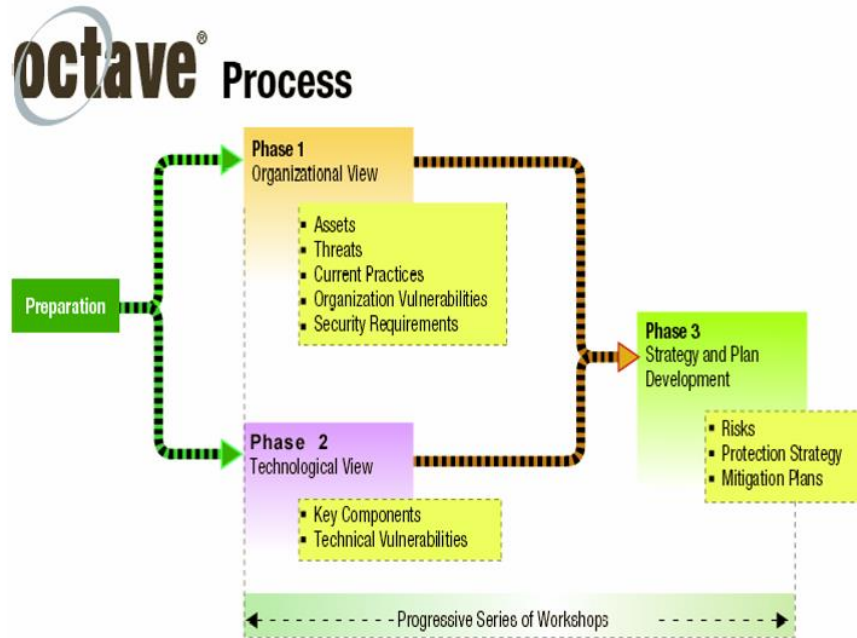
LINDDUN, like security threat modelling (Dhillon, 2011), might be regarded a time-consuming and sophisticated application. It is more common for LINDDUN to be used as a memory aid for a brainstorming-style activity than a technique to identify privacy risks.

The threat trees of the LINDDUN system are notoriously difficult to work with. In spite of the fact that they offer a useful overview, they may be deficient in semantics and contain just a few selection criteria to enable the evaluation of prospective hazards (Wuyts et al., 2020a).

LINDDUN threat trees' complexity necessitates a high level of privacy knowledge, according to industry feedback. This, along with the high cost of labor-intensive systematic elicitation and documentation, is preventing a thorough privacy threat modelling effort. LINDDUN's suggested and actual usages diverged so far that a more user-friendly version was developed to bridge the knowledge gap and ease the burden on users.

### **2.9.1.2. OCTAVE**

OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) It is a risk-based strategy and planning technique for IT security. The CERT Division of the SEI developed it in 2003, and it was enhanced in 2005. Organizational risks are the primary emphasis of OCTAVE, whereas technology risks are left out of the equation. Operational risk, security policies, and technology are its most three important parts. An organization's security-related activities and processes can be modelled using the OCTAVE approach. The threat to the organization's most vital assets is used to prioritize areas of improvement and to develop a security plan for the organization (Alberts, 2003). The OCTAVE process is depicted in Fig. 2.10.

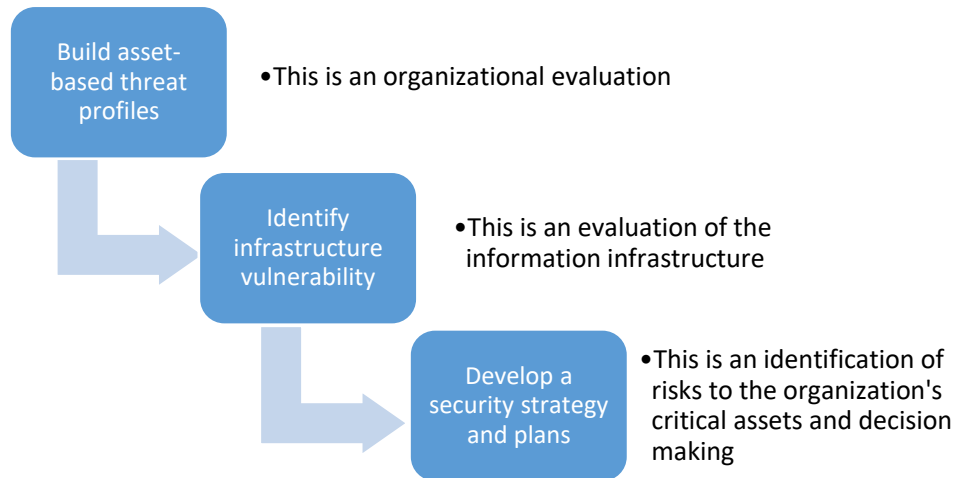


**Figure 2. 10.Octave Process.**

**Source (Cio-wiki, 2021.)**

The following evaluation process are included in the OCTAVE approach's assessment. Identify the organization's most valuable information assets. Determine which assets are most important to the organization and focus risk analysis efforts there. Look at the linkages between important assets, the danger to those assets, and the weaknesses both organizational and technical that might expose those assets to threats. Investigate how an organization's operations depend on certain assets and whether or not those assets are vulnerable to security threats. Lastly improve the safety of the organization's most important assets by developing a practice-based protection strategy and risk mitigation procedures.

In addition to the evaluation process Three-step technique to evaluating organizational, technological, and analytical factors is also used to establish asset-based threat profiles, identify infrastructure vulnerabilities, and design security strategy and plans see Figure 2.11.



**Figure 2. 11. Three-step technique to evaluating organizational threat profiles**

Source (author)

OCTAVE assesses activities rather than processes in their entirety and it is primarily suited for large businesses (Alberts et al., 2003). The approach is comprehensive, but it is also adaptable. The disadvantages of OCTAVE include a high time investment and extensive and ambiguous documentation (Stanganelli, 2016a) .

(Nweke & Wolthusen, 2020) claim that OCTAVE is quite complex. Learning requires a lot of work, and the procedures needed could take a while. Additionally, OCTAVE documentation can become long, which is likely to discourage policymakers from utilizing it as a threat modeling technique for their organization. The mechanism used for threat identification and categorization is another flaw in the OCTAVE threat modeling approach. When OCTAVE is used, it may become undesirable to record risks and threats using the threat tree because of the environment's complexity (Maghrabi et al., 2016). In the event of a very large computing environment, it can be difficult to determine which of the pathways accurately reflects the dangers being portrayed as the number of pathways grows.

### **2.9.1.3. STRIDE**

Designed by Loren Kohnfelder and Praerit Garg in 1999 and embraced by Microsoft in 2002 for producing secure systems, it is presently the most established threat modeling Methodology. STRIDE evaluates the system detail design by modeling the system in place and identifying the existing threats within the system (Khan et al., 2017a). The acronym



STRIDE, which stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege, is used to identify known threats to the system it is modeling (Karahasanovic et al., 2017). As demonstrated in Table 2.6 below, which lists the known threats and the properties they violate, (Selin, 2019) claims that STRIDE is more of a threat classification than a threat model or threat modelling framework. In order to have prepared reactions and mitigations to threats or attacks, one might use these groups of threats to develop a security strategy for a specific system.

**Table 2. 6.STRIDE Threat Categories definition and property violated.**

Adapted from (Prakash, 2020)

	<b>Threat</b>	<b>Definition</b>	<b>Property violated</b>
<b>S</b>	Spoofing	Impersonating something or someone else	<i>authenticity</i>
<b>T</b>	Tampering	Modifying data or code	<i>integrity</i>
<b>R</b>	Repudiation	Claiming to have not performed an action.	<i>non-repudiability</i>
<b>I</b>	Information disclosure	Providing information to someone not authorized to it.	<i>confidentiality</i>
<b>D</b>	Denial of service	Deny or degrade service to users.	<i>availability</i>
<b>E</b>	Elevation of privilege	Gain capabilities without proper authorization.	<i>authorization</i>

STRIDE employs the use of DFD`s in modelling the system which assists in accurately Identifying the boundaries of the system, data stores and objects, interfaces, processes functions ,techniques, entities and events which is a critical step in determine the success of the threat modelling (Shevchenko, Chick, et al., 2018a). From this identification one can be able to determine the trust boundary interactions that poses vulnerabilities towards the system and which becomes instrumental in determining the required mitigation techniques(Bodeau & McCollum, 2018).

(Scandariato et al., 2015) in their study of evaluating the productivity and performance of STRIDE they concluded that it is a relative easy to learn and execute methodology with low rates of false positives in terms of threats, (Khan et al., 2017b) in their work “threat modeling framework for cyber-physical systems using STRIDE” considers it as a light-weight and effective threat modeling methodology that simplifies the identification of vulnerabilities within a system.

### 2.9.1.4. DREAD

DREAD stands for Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability. Developed by Microsoft, it is a threat modelling technique. DREAD employs the standard qualitative risk assessment of HIGH, MEDIUM, LOW, with a qualitative risk rating of 3,2,1 given to each. For the most part, DREAD's threat modelling technique employs a score system to evaluate the likelihood of occurrence for each of the defined regions of the asset being threatened. Threat modelling using the DREAD technique is able to forecast the likelihood of occurrence of each threat discovered throughout the modelling process by integrating the derived risk rating values (UcedaVelez & Morana, 2015). The Figure 2.12 show a summarized DREAD approach and each term is discussed sequentially.



Figure 2. 12.Summary of DREAD threat model

Source (Wildcard, 2021)

It is important to understand how much damage an attack may bring to consumers and the organizations as a whole. Financial responsibility or damage to an organization's reputation are both examples of tangible damage. It also relies on the type of the attacks and the assets attacked.

Reproducibility evaluates the ease with which the attack can be repeated. If an attacker were to attempt an attack, they should be rewarded based on how much work they would have to put into it. There will be a higher rating for attacks that are easy to imitate than attacks that are more difficult to reproduce in the scoring system

Attackers can take advantage of a vulnerability if it is exploitable. Numerous attacks exist, some of which are simple enough for anybody to carry out, while others need a certain level of expertise to pull off. This knowledge led to the rating of threats with a high degree of exploitability as high risk and those with a low level of exploitability as low risk.

When a danger is realized, the number of people who will be affected is known as the number of affected users. A threat with a higher probability of affecting a large number of users than one with a lower probability of doing so will have a higher risk factor rating.

Discoverability is how easily a vulnerability may be found. There are threats that are extremely difficult to learn and threats that are quite simple to understand. A danger that is more difficult to understand would be given a lower score than one that has been made available to the general public for review and discussion(Wildcard, 2021).

Despite the fact that DREAD is an asset-centric threat modelling technique, it has been used in conjunction with the STRIDE model in the literature (Abomhara et al., 2015), (Amini et al., 2015) .The DREAD scoring method is employed in this technique to determine the likelihood of an attack exploiting a certain danger.

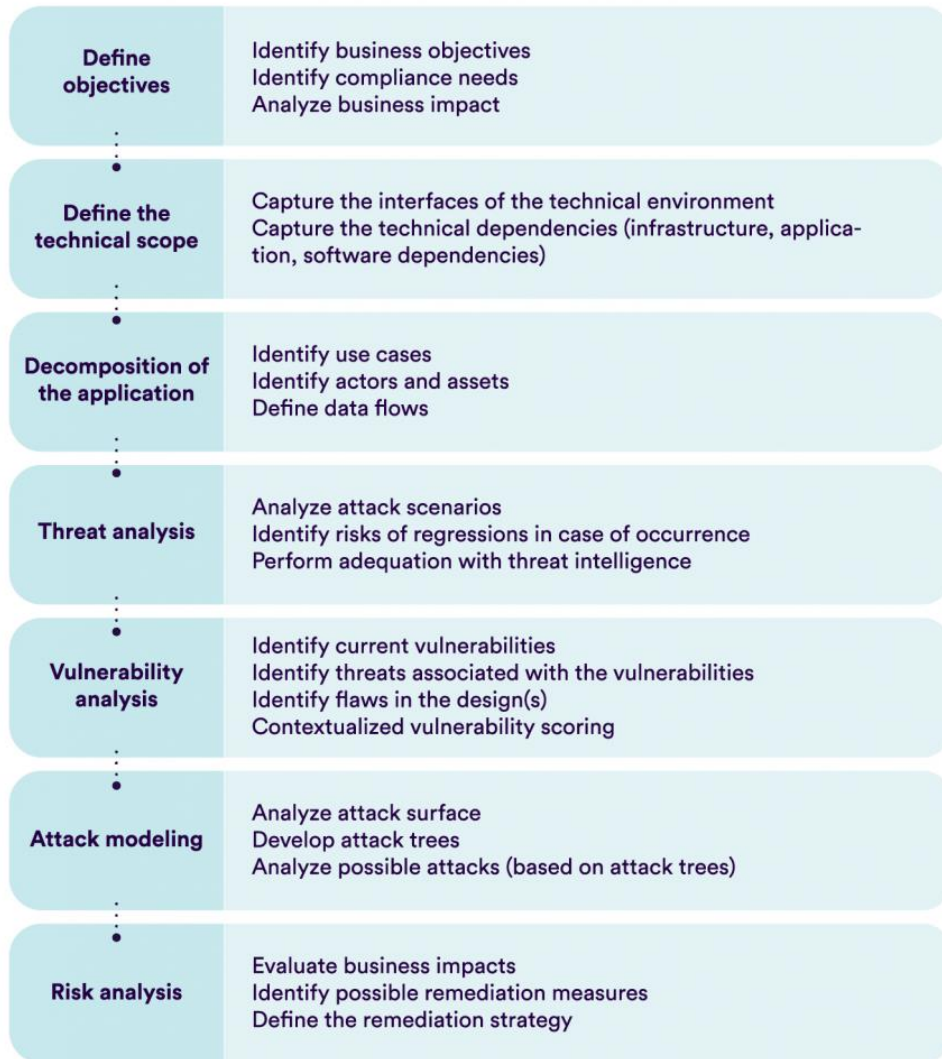
DREAD: has been demonstrated to be rather subjective and to provide inconsistent outcomes. In fact, Microsoft stopped using DREAD for their software development life-cycle in 2010 (Bodeau et al., 2018b)This highlights DREAD's limitations as a threat modelling technique. DREAD, on the other hand, is still frequently used and recommended for threat and risk modelling efforts. As a result, valuable ideas on how to enhance the scoring scheme's repeatability have been offered in (LeBlanc, 2007).

#### **2.9.1.5. PASTA.**

Tony UcedaVélez (UcedaVelez, 2012) created the Process for Attack Simulation and Threat Analysis (P.A.S.T.A.) in 2012 as a risk-centric threat modelling methodology. According to (UcedaVelez & Morana, 2015), the PASTA technique may be used in practically every context, with the exception of those in which executive sponsorship of the process and the created artefacts is not accessible. This is because the PASTA approach's deliverables are also meant to be acquainted with the organization's leaders.

PASTA aims to balance business objectives with technology requirements (Shevchenko, Chick, et al., 2018b). At different stages, it uses a variety of design and elicitation methodologies. For instance, high-level architectural diagrams are used. DFDs are used in stage three's second phase of establishing the technical scope. Stage six involves the construction of attack trees and use and abuse instances for analysis and attack modeling (Shull, 2016). The threat modelling process is elevated to a strategic level with the help of this approach, which demands security input from operations, governance, architecture, and development as well as key decision makers (Simeonova, 2016) PASTA adopts an attacker-centric stance and is sometimes referred to as a risk-centric paradigm. Last but not least, the process produces asset-centric output in the form of threat evaluation and enumeration (UcedaVelez, 2012)

The PASTA threat modelling technique consists of the following phases for the actual execution. The first level of threat modeling is determining objectives, which entails clearly establishing the system's business goals. In the second stage, the technological scope is established by listing all the system's resources. The system is then broken down in order to better understand how it functions. Threat analysis is done in the fourth stage to identify system hazards. The next stage is weakness and vulnerability analysis, which identifies vulnerable areas within the system and links them to the attack tree presented in the threat analysis step. The goal is to investigate the likelihood that the discovered vulnerabilities may be exploited using attack modeling and simulation. Finally, residual risk analysis and management are carried out to lessen threats that pose significant systemic concerns. Fig. 2.13 shows a representation of each level. PASTA is designed for businesses that seek to match threat modeling with their strategic objectives. This is because PASTA extends security responsibility to the entire firm by incorporating business impact assessments as a critical component of the PASTA process. This approach could be a drawback of using PASTA since it might need several hours of training and teaching for important stakeholders.

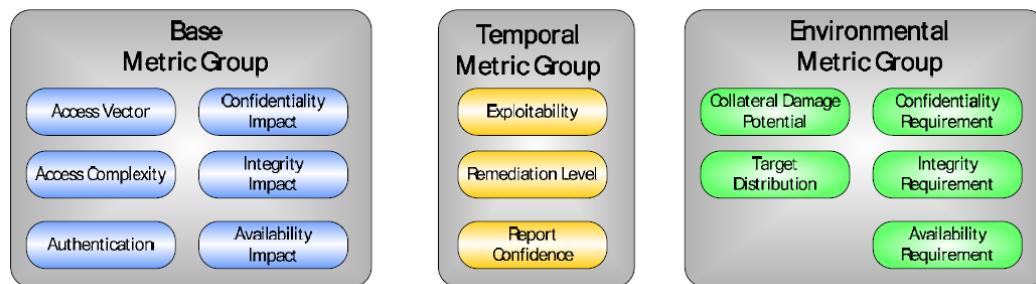


**Figure 2. 13.Stages of PASTA**  
Source (Dolbeau, 2022)

### 2.9.1.6. CVSS

For systems vulnerabilities, CVSS offers an open framework for describing their features and consequences. It captures the key characteristics of a vulnerability, and produces a numerical score representing its severity. CVSS was created and is maintained by NIST (Booth et al., 2013)with contributions from a CVSS Special Interest Group (Baker et al., 2013)supported by FIRST (Abraham et al., 2015) It can be applied on a wide range of technical systems because of its common and standardized scoring methodology. A CVSS score may be calculated using an online calculator (Booth et al., 2013).

As illustrated by Figure 2.14 the three categories of CVSS are: Base, Temporal and Environmental. The characteristics of a vulnerability are embodied in the members of the Base subset of the population. In the Temporal subcategory, we see the features of a vulnerability as they shift and change through time. Every person has their own set of environmental vulnerabilities, which are represented by the Environmental category. By using CVSS, IT administrators, bulletin providers, security vendors, application vendors, and researchers may all profit.



**Figure 2. 14.Categories of CVSS.**

Source adapted from (CVSS, 2019)

Analysts assign values to each indicator in the CVSS score, which is then calculated. It is important to note that there is usually a lack of clarity in the equations utilized in this procedure, although the documentation goes into great detail to describe every measure.

This approach is frequently utilized, despite some reservations about its opaque score calculation and the possibility of discrepancies generated by various judgment "experts" (Mell et al., 2007). Like some other threat models, CVSS is frequently used in conjunction with other threat modelling techniques.

### **2.9.1.7. TRIKE.**

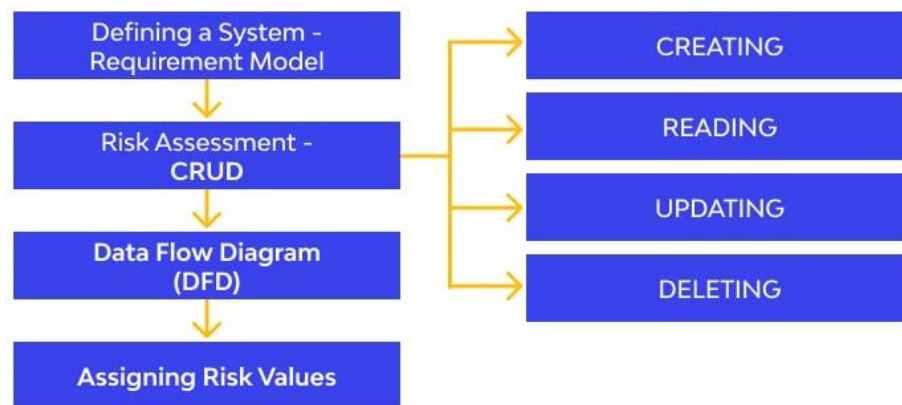
TRIKE is a security audit framework that leverages threat modelling as a strategy. This approach to threat modelling takes a risk management and defensive approach (Mead et al., 2018a). Like many other techniques as shown in Figure 2.16, TRIKE begins with a system definition. The analyst must first recognize and analyze the assets, scheduled activities, regulations, and system actors before building a requirement model. This procedure may result in the creation of an actor-asset-action matrix, where the rows correspond to actors and the columns to assets.

There should be four portions in each matrix cell, one for each CRUD action (creating, reading, updating, and deleting). The analyst should choose one of three values permitted activity, prohibited action, or action with rules in these cells. There should be a rule tree for each cell (Saitta et al., 2005).

Upon establishing the criteria, a DFD is constructed. Each piece is assigned to a certain actor or set. The analyst detects risks that fall under one of two categories: elevated privilege or denials of service (Stanganelli, 2016b) Whenever a danger is found, it becomes the root node of an attack tree.

Trike uses a five-point rating system for each activity based on the likelihood of an attack that might cause damage to assets through CRUD. A lower number denotes a larger risk, and actors are rated on a five-point scale according to the perceived risk they provide to the asset. Additionally, actors are evaluated on a three-dimensional scale based on the possible actions they may do on each object (Shevchenko, 2018).

The Trike scale system does not seem to be a formal technique. Despite the fact that the website for Trike 2.0 is live, it is not being maintained and no documentation has been provided for users. Figure 2.15 bellows shows the TRIKE methodology.



**Figure 2. 15.TRIKE methodology**  
Source adapted from (wallarm, 2020)

### **2.9.1.8. Attack Trees.**

According to (Ellison & Woody, 2010), security requirements engineering should take into consideration the expected threats and risks towards a system. A risk analysis process should be used to pinpoint the attacks towards the system, while the security requirements engineering should suggest mitigating solutions to attacks.

Attack trees are one of the mature practice that has been widely used for threat modelling and describing attacks within a system (Shevchenko, Frye, et al., 2018a). Attack trees are conceptual diagrams that provides a systematic description of the security of systems by showing how an asset or a target could be subjected to an attacked and this in turn assists in analyzing the threats within a system. Founded on the prior works by (Leveson, 1995) Bruce Schneier developed attack trees (Schneier, 1999)

A tree structure is used to depict an attack on a system, with the root node representing the objective and the leaf nodes representing possible approaches to achieve that goal. Each node becomes a sub goal, and the offspring of that node represent the paths to that sub goal's completion. Alternatives are represented by OR nodes, whereas various stages toward the same objective are represented by AND nodes (Wideli et al., 2019).

After the tree is constructed, multiple values can be assigned to the leaf nodes, which are then used to determine the goal's security. The security expert and system engineer have complete control over these parameters. The procedure of assigning values is done by hand. Other properties, such as the time it takes to complete a step, the cost of operations, the competence necessary to launch an attack, and so on, can be added to attack trees. Nowadays attack trees are blended with other techniques and within frameworks such as PASTA, CVSS and STRIDE. Table 2.7 provides a summary of threat modelling method features.



**Table 2. 7.Summary Threat Modeling Methods Features.**

DREAD	facilitates the evaluation of the risk posed by a threat exploit, has the ability to predict the possibility that a threat exploit will materialize, contributes to risk management, has a built-in ranking of threat mitigation measures, offers flexibility, and may be applied in any situation. (Hussain et al., 2014).
Trike	helps identify appropriate mitigation solutions, directly contributes to risk management, prioritizes threat mitigation, fosters stakeholder cooperation, has automated components, and has vague and insufficient documentation (Shevchenko, Frye, et al., 2018b).
OCTAVE	This method takes a lot of effort and has murky documentation, yet it helps find useful mitigation strategies and directly supports risk management. Additionally, it prioritizes danger reduction and promotes cooperation among stakeholders. Repeating it consistently yields the same results. It is purposefully made to be scalable.(Bodeau et al., 2018c).
PASTA	Contributes directly to risk management by assisting in the identification of suitable mitigation approaches. • Promotes collaboration among stakeholders by including built-in prioritizing of threat mitigation. • Requires a lot of time and effort but produces a lot of information.(Mead et al., 2018b).
Attack Trees	Facilitates the identification of applicable mitigation measures; produces consistent findings when repeated; and is simple to apply provided the user already has a solid grasp of the system in question.(Xiong & Lagerström, 2019).
CVSS	Has a built-in threat mitigating prioritization system produces predictable results when repeated Has automated elements • Uses obscure scoring calculations (Potteiger et al., 2016).
LINDDUN	It assists in the discovery of relevant mitigation solutions, prioritizes threat mitigation, and can take a long time to accomplish (Wuyts et al., 2020b).
STRIDE	assists in identifying applicable mitigation strategies • is the most mature • is simple to use, but requires a significant amount of time(Maheshwari & Prasanna, 2016).

## 2.10. Security Risk Assessment and Network Security.

Security risk assessment (SRA) protects an organization from hackers and cyber thieves and identifies and mitigates security threats to an organization it also examines an organization's security posture and compliance with industry standards and regulations (Progoulakis et al., 2021).

Security risk assessment procedures monitor open ports, anti-virus updates, password rules, patch management, and encryption strength. With this information, an organization's network security specialists may assess control effectiveness, identify risks, develop specific plans and solutions, identify vulnerabilities, and provide remedies (Lamarca, 2020).

In addition to identifying vulnerabilities, security risk assessment provides various advantages. Security risk assessment helps in the identification of weak security measures, vulnerable systems, and security risks to an organization so as to fix these flaws and improve

an organization's network security (Stoneburner et al., 2002). It lets you examine security measures in place. It also examines security measures' efficiency and improve them using security risk assessment tools. It can also assist in taking preemptive efforts to improve security controls' efficacy. It checks if an organization is compliant with industry standards (Abrar et al., 2018).

By using this risk assessment technique, businesses and managers may better allocate time and funds to protect and defend assets that need it the most. Thus, risk assessment may be seen as a tool for productivity that helps the company save time, money, and reputation (Lund et al., 2010). Although the risk assessment process generally uses some similar basic ideas, such as  $\text{risk} = \text{impact} * \text{likelihood}$ , there are usually many different and well-known models used to carry out the actual risk assessment. Some methods emphasize system breakdown to help identify risks (Biswas et al., 1989). These models offer a qualitative modeling approach that will facilitate the design of a risk assessment system and enhance the risk assessment process. When determining an overall risk assessment, it is important to take into account the failure rate of information systems as well as the unpredictable nature of human behavior. This technique helps to overcome these uncertainties.

Other methods need the use of a knowledge-based system together with qualitative issue resolution, which may lead to the creation of a universal and transportable risk assessment tool (Marhavilas et al., 2011). The usage of such knowledge-based systems frequently includes the addition of fault and event trees. Event and fault tree analysis includes accurately identifying each potentially damaged system as a branch on the tree after distinguishing unique possible failures as independent "tree-roots or trunks." This method allows for the precise creation of a list of all potentially affected systems for a particular failure (Rezayat, 2000). The ability to simulate "what-if" scenarios using fault tree analysis is one of the main advantages of using it to develop knowledge-based systems. By looking into potential system failures, organizations and managers may get a complete and accurate view of potential risk. Another popular method for assessing risk is the idea of annualized loss expectation. A financial definition of risk that businesses use to calculate the estimated value or cost of an event that might result in a certain risk is called annualized loss expectation (Varga et al., 2021). Using this process, an organization calculates risk by multiplying a certain financial

sum by the likelihood of the risk "occurrence." The cost is calculated by summing the direct and indirect financial sums related to the risk's occurrence over the course of a year.

Different methods have been used to approach the risk assessment process. One model divides risk assessment into six distinct phases (Yang et al., 2006). This method's initial step is to create a cost factor grading scheme. When the grading system is created, risks are found. The next step is to determine the risk probability. The assessment of risk severity that follows standardizes an overall risk on a scale of 1-100. The following categories can then be applied to the 1-100 scale. Systems having an overall risk rating of 0 to 5 are deemed "low risk," 5 to 15 "moderate risk," 15 to 50 "high risk," and 50 to 100 "very high risk." The third step is to offer suggestions for ways to reduce the hazard that has been described (Yang et al., 2006).

Not all risk-adjustment strategies take into account both effect and likelihood. Some risk assessments focus solely on the possibility that the hazard may materialize (Aubert et al., 2005). When a given risk's influence or incidence leads to an irreversible condition, this kind of risk assessment is especially helpful. This kind of risk evaluation is used often in the medical industry. Since death is the end result, medical risk evaluations often solely consider the possibility of a certain condition. In these cases, the impact is no longer considered since it is irrevocable (Aubert et al., 2005).

The ability of a company to protect its information technology assets depends on the completion of a proper risk assessment, which is both beneficial and necessary (Maclean, 2017). The company and management team will be able to make precise and knowledgeable judgments on resource management, hiring, and budgeting when the risk assessment process is finished. A thorough grasp of the dangers related to each individual system as well as the overall level of risk associated with the deployed technology results from a well-defined risk assessment (Shameli-Sendi et al., 2016b).

Every company in every sector of the economy must play a crucial role in the accurate, thorough, and efficient risk assessment of a network system. As standards develop, practices change, and new types of risk assessment are introduced, organizations must find a way to make sense of all of this. A robust risk assessment process gives organizations more power by ensuring that risks have been identified and accurate, pertinent controls are in place (Peltier, 2005).

## **CHAPTER 3 METHODOLOGY**

### **3.1. Research Design**

The research design unfolds in two main phases (1) model development (2) simulation, each contributing distinct perspectives to achieve the study objectives.

### **3.2. Phase 1: Model Development**

#### **3.2.1. Literature Review**

The model development phase commences with an extensive review of existing literature on network security including threats and attack techniques, models, frameworks and artifacts to guide in development of secure networks and related methodologies. This review informs the identification of foundational concepts, best practices, and gaps in current approaches.

#### **3.2.2. Conceptualization of holistic security pattern-based model**

Building upon the insights gained from the literature, defensive pattern model is conceptually formulated. This model is designed to encapsulate a comprehensive and structured approach to address network security challenges.

### **3.3 Phase 2: Simulation**

#### **3.3.1 Selection of Simulation Tools**

The simulation phase involves the application of selected simulation tool to assess the practical efficacy of the pattern model. Simulation tool is chosen based on their relevance to network security scenarios, considering factors such as scalability, accuracy, and real-world applicability.

#### **3.3.2 Scenario Design**

Realistic network security scenario is designed to simulate various threat vectors, vulnerabilities, and attack scenarios. These scenarios aim to emulate the complexity and diversity of challenges faced by modern network architectures.

### **3.3.3 Implementation of the Model**

The developed pattern model is implemented within the simulated environments. This phase involves integrating the model into the network architecture and evaluating their performance in response to simulated security threats.

### **3.3.4 Metrics Definition**

Quantitative metrics are defined to measure the effectiveness of the security pattern-based model. These metrics include but are not limited to response time, incident detection accuracy, and overall network resilience. The criteria for success are established based on industry standards and best practices.

## **3.4 Ethical Considerations**

The research adhered to ethical principles, ensuring the confidentiality of sensitive information and permission to use the data has been granted by the owners of the data by being provided free for download and use by anyone as long as acknowledgement is done. Simulations are conducted in a controlled environment preventing unintended consequences.

## **3.5. Dataset**

This study employs the use of a network based secondary dataset to test the model. When choosing the dataset for this study it leverages on detailed overview analysis of network-based intrusion detection data sets with respect to the data set properties by (Ring et al., 2019). Their study looked 34 datasets ranging from the year 1998 to 2017 (10 years) and evaluated them based on 15 properties which included the year of creation of dataset, public availability, normal user behavior within the traffic, attack/malicious traffic, meta data, format of the traffic (packet based, flow-based or other type), anonymity, data count, duration of collection, kind of traffic, type of network environment of the dataset, complete network meaning it is a traffic from a network with all the typical network resources, predefined splits of training and test sets, balanced traffic which means it containing both normal and malicious traffic, lastly whether the data set has explicit labels.

**Table 3. 1.Dataset properties and their value ranges (Ring et al., 2019)**

Data set properties and their value ranges		
General Information	Year of Traffic Creation	year (1998 – 2017)
	Public Availability	no, no information found (n.i.f.), on request (o.r), yes
	Normal Traffic	no, yes
	Attack Traffic	no, not specified (n.s.), yes
Nature of the Data	Metadata	no, some, yes
	Format	bidirectional (bi.) flow , logs, other, packet, unidirectional (uni.) flow,
	Anonymity	none, not specified (n.s.), yes, yes (specific attributes)
Data Volume	Count	size of data in Gigabyte (GB) or number of flows/packets/points
	Duration	recording time of the data set
Recording Environment	Kind of Traffic	emulated, real, syntethic
	Type of Network	diverse networks, enterprise network, honeypot(s), internet service provider (ISP), not specified (n.s.), small network, production network, university network
	Complete Network	no, not specified (n.s.), yes
Evaluation	Predefined Splits	no, not specified (n.s.), yes
	Balanced	no, not specified (n.s.), yes
	Labeled	indirect, no, yes, yes (IDS), yes with background (BG.)

Since it is recommended when searching for a good dataset in use should be current, well labeled and the required format, the study picked 12 datasets from their analysis that were falling in their latest three years (2015 to 2017) as shown in table 3.2 then analyzed the other properties and settled for UNSW-NB15 which had the desired features for the study. From the table 3.2 the highlighted cells depict the reasons as to why a dataset was dropped.

**Table 3. 2.Study dataset selection criteria**

Data Set	General Information				Nature of the Data			Data Volume		Recording Environment			Evaluation		
	Year of Traffic creation	Public Available	Normal Traffic	Attack Traffic	Meta Data	Format	Anonymity	Count	Duration	Kind of Traffic	Type of Network	Complete network	Prefed Splits	Balanced	Labeled
AWID	2015	o.r.	yes	yes	yes	other	none	37M packets	1 hour	emulated	small network	yes	yes	no	yes
CICIDS 2017	2017	yes	yes	yes	yes	packet, bi. flow	none	3.1M flows	5 days	emulated	small network	yes	no	no	yes
CIDDs-001	2017	yes	yes	yes	yes	uni. flow	yes (IPs)	32M flows	28 days	emulated	small network	yes	no	no	yes
CIDDs-002	2017	yes	yes	yes	yes	uni. flow	yes (IPs)	15M flows	14 days	emulated	small network	yes	no	no	yes
DDoS 2016	2016	yes	yes	yes	no	packet	yes (IPs)	2.1M packets	n.s.	synthetic	n.s.	n.s.	no	no	yes
IRSC	2015	no	yes	yes	no	packet, flow	n.s.	n.s.	n.s.	real	production network	yes	n.s.	n.s.	yes
Kent 2016	2016	yes	yes	n.s.	no	uni. flow, logs	yes (IPs, ports dates)	130M flows	58 days	real	enterprise network	yes	no	no	no
NDSec-1	2016	o.r.	no	yes	no	packet, logs	none	3.5M packets	n.s.	emulated	small network	yes	no	no	yes
NGIDS-DS	2016	yes	yes	yes	no	packet, logs	none	1M packets	5 days	emulated	small network	yes	no	no	yes
UGR'16	2016	yes	yes	yes	some	uni. flows	yes (IPs)	16900M flows	4 months	real	ISP	yes	yes	no	yes with BG.
Unified Host and Network	2017	yes	yes	n.s.	no	bi. flows, logs	yes (IPs, ports dates)	150GB flows	90 days	real	enterprise network	yes	no	no	no
UNSW-NB15	2015	yes	yes	yes	yes	packet, other	none	2M points	31 hours	Real/emulated	small network	yes	yes	no	yes

## **3.6. The Study Test Environment**

### **3.6.1. Kaggle Test Bed**

Kaggle proves to be an excellent choice for conducting a study that involves classifying attacks and clustering them based on three layers of network security: the User layer, Host layer, and Media layer. The platform has amassed a diverse collection of real-world datasets, offering researchers the opportunity to access data representative of various attack scenarios. This diversity is pivotal in training robust and applicable machine learning models tailored to the intricacies of cybersecurity. Kaggle ensures a secure and controlled environment for working with sensitive data, addressing privacy and security considerations. Researchers can confidently navigate their studies while benefiting from the collaborative and competitive features that Kaggle uniquely offers in the realm of machine learning and data science.

### **3.6.2. Python**

The case for using Python in conjunction with Kaggle for a study on classifying attacks and clustering them based on network security layers is strong due to Python's readability, simplicity, and widespread adoption. The language's extensive ecosystem, including popular libraries like NumPy and Scikit-learn, empowers researchers to efficiently manipulate and analyze datasets within Kaggle's environment. Python's dominance in the machine learning community, supported by its integration with deep learning frameworks like TensorFlow and PyTorch, ensures that researchers can implement sophisticated models for classification tasks. Another reason is it can carry out predictive analytics to various datasets due to the ability of predefined algorithms such as logistic regression. It can also assist in statistical functionalities to understand the data and extract information that can be used in the decision-making process among others. The language's prevalence in Kaggle aligns with the platform's educational resources, tutorials, and community discussions, providing a cohesive and well-supported environment for impactful research in the cybersecurity field.

## **3.7. Performance Metrics**

There exist a number of metrics to evaluate ML based IDS systems; however, this research aims to maximize the correct predictions of instances in the test dataset. The main measure to look at is the *Accuracy (AC)* defined as follows:

$$Accuracy = \frac{TN + TP}{FP + FN + TP + TN} \quad \dots eq \dots (3.1)$$

whereby the *TP* stand for True Positive and is the rate of examples correctly identified as attacks. *TN*, True Negative, is the rate of legitimate traffic classified as legitimate. *FP*, False Positive, sometimes referred to as Type I error, is the rate of legitimate traffic classified as attacks. *FN*, sometimes referred to as Type II error, is the rate of legitimate traffic classified as intrusions. Additional metrics considered in this paper are the Recall, the Precision and *F1*score defined as follows

$$Precision = \frac{TP}{TP + FP} \quad \dots eq \dots (3.2)$$

$$Recall = \frac{TP}{TP + FN} \quad \dots eq \dots (3.3)$$

$$F1_{score} = 2 \frac{Precision \cdot Recall}{Precision + Recall} \quad \dots eq \dots (3.4)$$



## **CHAPTER FOUR**

### **MODEL FORMULATION**

This chapter is titled “Model Formulation”. The chapter provides an (1) *Introduction*. Thereafter follows a discussion on a proposed extension to the OSI Model in (2) *Extending the OSI Model* to include a User layer hence mapping the extended OSI model into the three-layer network security domain (TLNSD). In (3) *Model Development* section, the Network Architecture Security Pattern Model is presented, composed of the Attack Context Checking, Attack Surface Identification, and Risk Assessment Components. In addition, this section also presents the CAPEC Repository which provides the dataset that will be used to identify the attacks.

#### **4.1. Introduction.**

The network engineer of today must be security-conscious, and the security engineer must be aware of the network he is responsible for protecting (Cole et al., 2003) They could overlook the fact that security measures like firewalls and authentication controls only make up a portion of the solution. Such network-level defenses can only prevent a very specific range and type of threats. A multi-tiered approach to security is required if one actually wishes to defend the company, and here is where the OSI model proves to be quite helpful (Eric, 2016)

The Open Systems Interconnection, or OSI Model, is a security architecture that defines application security guidelines in seven levels. (Solomon, 2016) physical, data link and network layers (media layer), and transport, session, presentation and application layer(host layer), all of which must be secured for networks to be considered safe.

Several writers have proved the OSI Model's relevance and use in the area of network security. Reed (Reed, 2003) gives an overall viewpoint in which typical information security challenges map clearly to the logical structures offered in the OSI Seven Layer Network Model, demonstrating the Seven Layer Model's use in assessing information security problems and solutions. They assess typical information security risks and controls on each layer and demonstrate that the Seven Layer Model's scheme for layer interaction provides insight into some of the issues encountered by concentrated, "single-layer" security solutions. They offer a holistic multi-layer strategy based on network model layers to tackle the problem rather than discrete solutions and logical or physical hardware layers.

Pace, (2014) illustrates a rational, thorough, and practical way to safeguarding an organization's information resources by utilizing the OSI Model's seven levels. They conclude that no one layer of the model, when fully implemented, provides even a smidgeon of safety. A comprehensive security solution takes into account all levels of the OSI model.

Martinović et al., (2014) proposed several techniques of control and protection for the OSI's multiple layers. It can be seen from this that granularity is achieved in terms of network security by moving from general to more specific security measures. All of this is done to improve security by combining multiple layers of security, also known as "defense-in-depth," which states that "even if one measure fails, another one will take its place."

## **4.2. Extending the OSI model**

The seven-layer model is more than sufficient for network applications, but when it comes to network security, there are notions that require organization that do not fit into the traditional network model. Crutchley, (2002) mentions two more factors—people and policy—that are crucial to the assessment of a network security posture in a brief online essay. With people engaging with applications at layer eight and policies (theoretically) governing people's behaviors at layer nine, Crutchley suggests adding these two components to the model as two extra levels.

(Greg, 2019) proposes an eighth layer which he refers to as the human layer, which is the layer at which technology interfaces with people. This layer deals with people and policies. The rational being that apart from vulnerable software's and hardware as enablers of attacks, human/user are also a source that can be exploited especially for those who are not security conscious. He proposes that the two critical issues should be addressed in this layer and this include security training of users so that they can make informed decisions when faced with security challenges and secondly establishing security policies, guidelines and procedures to secure organization against an attacks all this they inform play a critical role in setting the overall tone and define how security is perceived within an organization and lack of it is one of the biggest vulnerabilities many organizations have.

We can implement the best security solutions known at the various layers of the OSI model and still be vulnerable through people and employees hence the eight layer “human layer” is

an important consideration on the OSI Model(Gregg et al., 2006) for a holistic defense of networks .

The seven-layer OSI Reference Model was given a human factors expansion by Bauer & Patrick, (2014).This extension gives a common conceptual vocabulary to promote meaningful talks between the HCI (Human-Computer Interaction) disciplines and those in charge of network and application design, and it is compatible with the design principles of the OSI model. As a function of network and device capabilities, this new conceptual common ground may be utilized to relate applications to human requirements and to establish whose responsibility a problem is. The model Figure 4.1 aids in determining which discipline a problem belongs to by providing a standard vocabulary to assist bridge diverse fields of study. Additionally, it demonstrates how a full end-to-end perspective necessitates understanding how user experience is impacted by overall network and application performance. Finally, it offers a plan for ensuring that applications can function satisfactorily within network constraints while still meeting Layer 10 requirements.

**Figure 4. 1.OSI and User Interaction**  
**Source (Bauer & Patrick, 2014)**

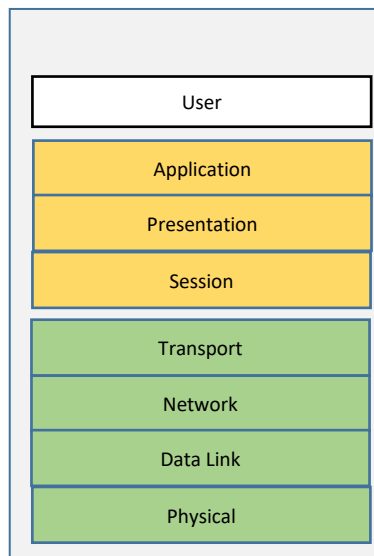
	10.	Human Needs (communication, education, acquisition, security, entertainment...)
HCI	9.	Human Performance (perception, cognition, memory, motor control, social...)
	8.	Display (keyboard, GUI/CLI, vocal, bpp, ppi, ppm...)
OSI	7.	Application (http, ftp, nfs, pop...)
	6.	Presentation (ps, lz, iso-pp...)
	5.	Session (dns, rpc, pap...)
	4.	Transport (tcp, udp, rtp...)
	3.	Network (ip, dhcp, icmp, aep...)
	2.	Data Link (arp, ppp...)
	1.	Physical (10bt, xDSL, V.42...)

Greg, (2019) refers to Layer 8 as "user" or "political". Cyberoam associates Layer 8 with identification of user, controlling of their activities in a network and setting policies (Cyberoam, nd). (Curry, 2013) points out that Layer 8 represents the individual person, Layer 9 represents the organization, and Layer 10 is concerned with government or legal compliance).Because the OSI layer numbers are used to characterize the network, a user-caused problem can be stated as a layer 8 problem (Apposite, 2017).Mosco (1996) cites the 8th layer as crucial to understanding the OSI Model because it drives political policies like as spectrum management, network neutrality, and digital inclusion, all of which help shape the technology utilized inside the OSI's seven layers.

According to (Kaspersky, 2020) user can become attack vectors in many forms putting organization networks at greater risks and acting now, to prevent employee-related threats, has never been more important. There is need for training so that users can become more aware of the impact of their actions to help to mitigate the risk of users becoming an attack vector.

According to (Sapronov, 2015) users have the ability to swing the balance one way or the other especially now that there is standoff between cyber criminals and security professionals, as a result of the continuous invention of new tricks in order to evade the security software and hardware currently being used.

Based on the discussion the study concludes that a user in a network setup should cease being something that system administrators and the top management do not know what to do with, instead becoming an important aspect that can be leveraged in the protection of networks, something that is resistant and reliable, and also demands the vision of network security professionals. Hence this study adopts an enhanced OSI model with the user aspect as part of one of its constructs to assist in evaluating network security problems and solutions, see Figure 4.2.



**Figure 4. 2.Extended OSI Model.**

## **4.3. Model Development**

### **4.3.1. Attack Context Checking**

#### **4.3.1.1. Three Layer Network Security Domain (TNLSD) Component**

Security specialists advocate the notion of security defense in depth. According to this theory, network security should be tiered, with several measures utilized to secure the network. No security system can be guaranteed to survive every possible attack. As a result, each mechanism should have a backup mechanism (Richardson, 2022).

When building and implementing network security, following a structured modular set of procedures will assist handle the many problems that play a role in security design.

Many security plans have been established haphazardly and have failed to safeguard assets and satisfy the core security goals (Corgi, 2020).

Because of modularity, you can keep each design aspect basic and easy to grasp. Simplicity reduces the need for substantial training for network operations workers and speeds up design implementation. Because each layer has distinct functions, testing a network architecture is simplified. Fault isolation is increased because network personnel may readily identify network transition points, allowing them to isolate potential failure locations.

With its three-layer hierarchical concept, Cisco advocates a modular approach. This approach splits networks into core, distribution, and access layers to aid in security design and implementation. Based on the needs of the company, they propose that networks be developed in hierarchical, modular, redundant, and secure network designs (Upravnik, 2016). Hierarchy and modularity let you to build a network with numerous interconnected components in a layered and organized manner. A hierarchical model can help you optimize network performance, decrease the time it takes to install and debug a design, cut expenses, and increase security (Tiso, 2011).

Discovering possible attacks holistically is an important step in designing system security since the detected attacks will identify fundamental security needs and offer insight on what and why security measures are necessary. (Backers 2015) proposed a comprehensive attack analysis approach that investigated different attacks from the attacker's perspective. The framework accepts a three-layer requirements model with social, software, and physical layers

as input, capturing a holistic system context that is utilized as the domain model throughout the attack analysis. The methodology produces a collection of potential multistage attacks, which are then fed back into the three-layer requirements model to identify essential security requirements.

Given the above the study modularize the extended OSI model into three layers which are the organization layer, Host Layer and Media Layer and christen them as the three-layer network security domain as shown in Fig 4.3. The Media layer combine's the physical, data link and network layer which are in summary concerned with controlling the physical delivery of data over the network. The host layer which combines transport session presentation and application layers is concerned accurate delivery of data between computers. The organization layer is concerned with the users and how they interact with the network.

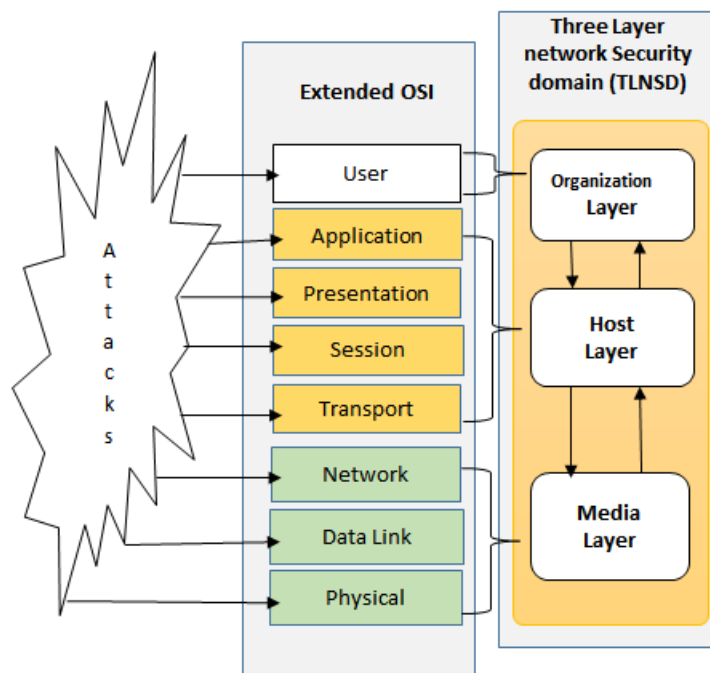


Figure 4. 3.Three-layer network Security domain (TLNSD).

#### 4.3.1.2. Anti-Goal Identification.

(Lamsweerde, 2004a) was the first to employ anti-goals to represent an attacker's malevolent intentions for system assets. An anti-goal model depicts how an attacker's abstract anti-goals

are narrowed to terminal anti-goals that attackers may achieve, so capturing the attacker's techniques. Analysts may successfully detect system dangers and utilize this information to develop safe systems by using anti-goal models.

Several articles have used anti-goals to capture the reasoning behind attacker behaviors. (Lamsweerde, 2004b) describes a method for modeling, specifying, and analyzing application-specific security needs. The technique is built on a goal-oriented framework for generating and resolving roadblocks to goal achievement. The enhanced framework covers harmful hurdles (referred to as anti-goals) erected by attackers to jeopardize security goals. Threat trees are created progressively through anti-goal refinement until leaf nodes are obtained that are either software vulnerabilities that the attacker may see or anti-requirements that the attacker can implement. By applying threat resolution operators to the specification of the anti-requirements and vulnerabilities discovered by the analysis, new security requirements are obtained as countermeasures. In addition, the study presents formal epistemic specification structures and patterns that may be utilized to aid in the formal derivation and analysis process. The strategy is shown using a web-based banking system, where subtle attacks have lately been identified.

(T. Li, Horkoff, Beckers, et al., 2015) describes an ongoing study to build a comprehensive attack analysis approach. The method uses goal modeling to capture attacker harmful intent as anti-goals, which are then methodically improved and operationalized into real attack activities that target various assets (e.g., human, software, and hardware). A comprehensive attack pattern repository (CAPEC) is seamlessly incorporated into the methodology to give analysts with practical security expertise and aid in the identification of potential attacks in certain scenarios. Finally, a set of security rules for mitigating discovered threats is presented.

According to (T. Li, Paja, et al., 2015) , the increasing complexity of systems makes their protection increasingly difficult, as a single vulnerability or disclosure of any component of the system might result in major security breaches. This is exacerbated by the fact that the system development community has not kept up with advances in attack knowledge. The paper proposes a holistic attack analysis approach to identifying and combating both atomic and multistage attacks, taking into account not only software attacks but also attacks against people and hardware. To bridge the knowledge gap between attackers and defenders, the

malevolent intents (i.e., anti-goals) of attackers are collected, and a comprehensive attack pattern repository (CAPEC) is used to operationalize attacker objectives into real attack activities. Based on the findings of the attack analysis, suitable security controls may be chosen to combat possible threats.

(T. Li, Horkoff, Paja, et al., 2015) have used the anti-goal method in their work to identify system dangers and develop relevant countermeasures. They explore genuine attack situations to analyze anti-goal enhancements from an attacker's perspective. For developing anti-goal models, they used three ideas from a goal model method developed by (Jureta et al., 2010) : Goal, Task, and Domain Assumption. A goal records attacker intents (i.e., anti-goals); a task gives precise attack activities undertaken by attackers; and a domain assumption specifies a system-relevant suggestive attribute.

(Horkoff & Yu, 2016), their solution uses a three-layer, goal-oriented requirements model to account for security vulnerabilities at various abstraction layers. We may build an attack strategy that suggests a set of attack scenarios from which associated security measures can finally be generated by iteratively refining root anti-goals into operationalizable anti-goals. Such techniques capture not just the universe of conceivable attacks, but also an attacker's strategy, which may include alternate plans and the combination of many stages to achieve a harmful aim.

### **4.3.2. Attack Surface Identification.**

Analyzing the attack surface is an efficient and methodical method of finding all potential attack scenarios, which are required for doing security analysis from the perspective of an attacker. It is vital for focused security analysis to determine which attack surfaces are likely to be operationalized. Each attack surface, which is made up of one or more anti-goals that explain the malevolent intentions of attackers, gives information on what and when attackers may aim to attack.

Analysts must realistically determine how attackers may attack a system in order to design mitigation techniques.

To establish a comprehensive attack strategy, (Beckers ,2016) has created a framework based on real evidence to assist methodical study of attack tactics, resulting in more complete plans and a more thorough security analysis. They conduct grounded research on three genuine



attack situations to analyze how attackers elaborate their malevolent intentions in reality, and as a result, they find five anti-goal refinement patterns.

(T. Li & Horkoff, 2014) suggested an anti-goal refinement methodology that systematically refines anti-goals by exploiting anti-goal refinement patterns and finally discloses attack scenarios. The framework was assessed by applying it to a credit card theft scenario, with the outcome demonstrating that the framework is capable of generating a comprehensive attack plan that not only covered the previously known attack scenarios, but also identified new attack possibilities.

(Mylopous, 2016), investigated three real-world attack scenarios to learn how attackers modify their malevolent intents, and derived five refinement patterns. Based on such refinement patterns, they present an anti-goal refinement framework for methodically generating attack tactics from the perspective of an attacker. Finally, they assess their performance using a credit card fraud scenario.

(Tong, 2015), describe attack tactics that attackers may use to destroy systems by modeling an attacker's high-level malevolent intentions as structured anti-goals, which can be developed and operationalized like conventional goals but from an attacker's perspective. They contend that the attacker has specialized techniques of elaborating their high-level anti-goals till they reach clear and precise anti-goals, and that the elaboration of anti-goals equates to the production of their attack strategies. They suggest characterizing an anti-goal as a quadruple: asset, threat target, and interval, in order to methodically analyze the evolution of hostile purpose.

According to Figure 4.4, the study identifies the attack surface and perform attack context checking on the Three Layer Network Security Domain (TLNSD) by subjecting and analyzing the attacker's malicious intent through the anti-goal identification process, and the study adopts and characterize the attackers' anti-goals as a triple construct consisting of Asset, Threat, and Target.

The attacker's intents against the network are captured as anti-goals, which are then methodically evaluated to uncover the attacker's attack plans. When attacking a network, an attacker might employ a variety of attack tactics in order to achieve his primary anti-goal. An attack plan clarifies which system components to attack and when to attack them. The study

propose that once root antigoads are discovered, they be systematically refined in order to investigate potential attacks across three levels of the network security domain. To that aim, study want to explore a variety of attack scenarios in order to better understand how attackers devise attack techniques in order to carry out their malevolent objective. The study can then identify the attack targeting the various surfaces based on the study.

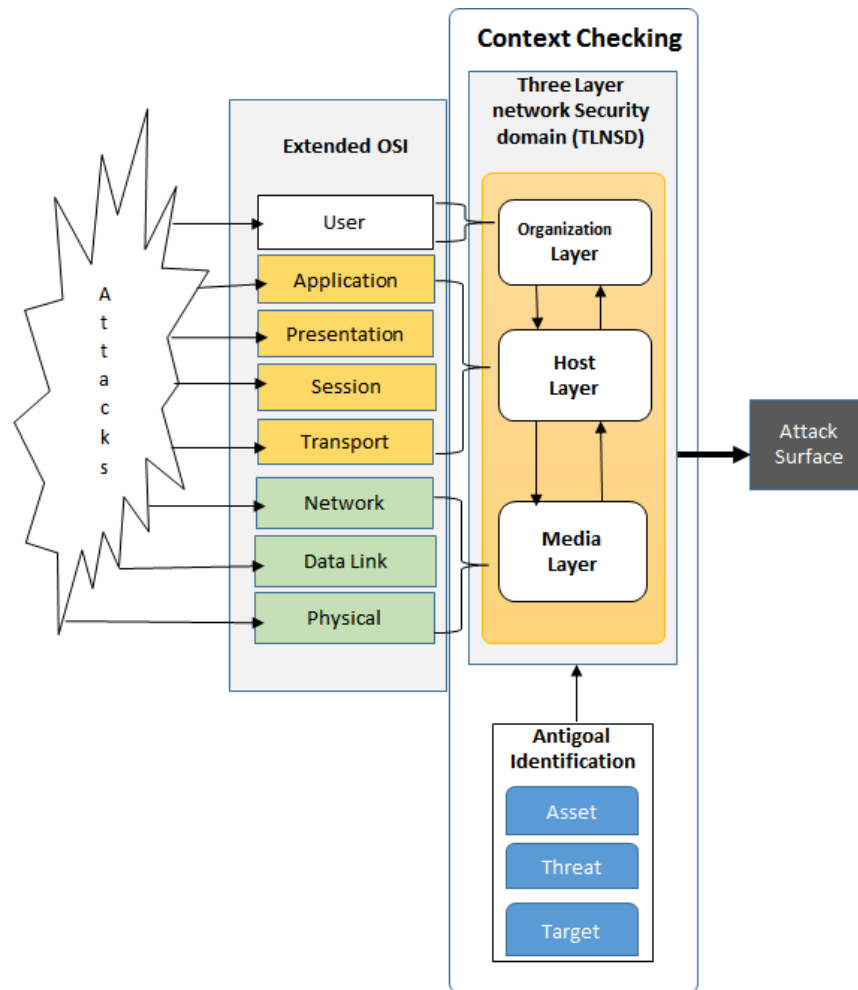


Figure 4. 4.Attack surface Identification.

### 4.3.3. CAPEC Repository

The CAPEC (Common Attack Pattern Enumeration Classification) Repository is a formal illustration of a Network attacker’s tools, methodologies, and perspective. it contains a collection of known patterns of attack employed by adversaries to exploit known weaknesses in a network set up (capec.mitre.org, 2021).

CAPEC defines each attack using descriptive textual fields known as elements which explains in details each exploit identified. The current CAPEC has 527 specific security attacks with their elements(CAPEC, 2021). MITRE developed CAPEC as a result of lack of a standard and consistent documentation of attacks that would support the attack-specific security research(Hoglund & McGraw, 2005). CAPEC is necessary standard for effective mitigating of attacks, a security analyst who is fretful and interested in formulating a defense mechanism against an attack or reducing exposure to the attack, should be able to review an attack pattern within CAPEC.

A systematic representation and official standard for identifying individual attack patterns are provided by the Common Attack Pattern Enumeration and Classification (CAPEC) list (A. Martin, 2006).Deepening one's understanding of attack patterns may increase awareness of existing exploits, vulnerabilities, and weaknesses as well as the injection of security across the board in a system (Gegick & Williams, 2005a).By reducing exposure to documented vulnerabilities and known flaws, integrating and expanding attack pattern information can increase security (Pauli & Engebretson, 2008).A security checklist that incorporates attack patterns might result in a greater degree of protection (Engebretson et al., 2008)..

(Kaiya et al., 2014) in the process of eliciting security requirements propose a method leveraging on CAPEC which they concluded enables security experts to save time. (Kanakogi et al., 2021) suggested a technique employing Natural language Processing Technique to track associated CAPEC-ID from CVE-ID using Common Weakness Enumeration in order to address the question of how to effectively respond to security vulnerabilities (CWE)..The traditional tracing technique makes advantage of the connections between each repository. Manual tracing is necessary, but accuracy could also be a problem. The Doc2Vec and TF-IDF measures are used in the tracing approach to determine similarity between the CVE-ID and CAPEC-ID. The findings indicate that the Doc2Vec model may be improved, despite the fact that TF-IDF had a greater accuracy.

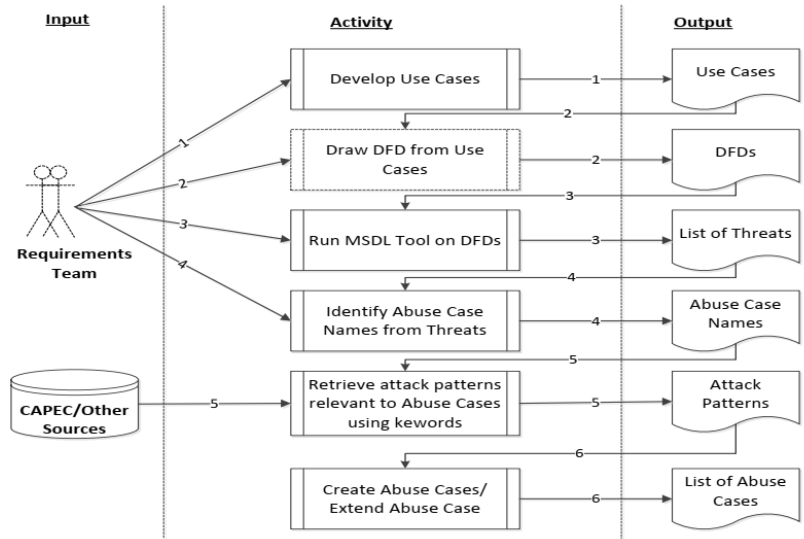
#### **4.3.3.1. Attack Patterns**

Attack patterns are derived from the notion of design patterns and explain typical approaches for attacking software. Attack patterns capture and explain the attacker's point of view, which can assist software engineers in thinking like an attacker. (McGraw & Hoglund, 2004)

identified 49 attack patterns. The Common Attack Pattern Enumeration and Categorization (CAPEC) repository contains 463 publicly available attack patterns, as well as a thorough schema and taxonomy for classification (Y. Zhu, 2015).

(Barnum & Sethi, 2007) shown that attack patterns may be deployed at any stage of the safe software development life cycle. (Gegick & Williams, 2005b) employed regular expressions to create attack patterns based on existing vulnerability databases, which they then used to discover security weaknesses during program design. Pauli & Engebretson (2009) developed a software tool for retrieving associated CAPEC attack patterns based on system prerequisites, such that mitigation methods for the recovered attack patterns might be used during system design and implementation.

As illustrated in Fig.4.5, (Yuan et al., 2015) suggest a way for constructing abuse scenarios based on Microsoft's threat modeling and attack patterns.



**Figure 4. 5.**The method for developing abuse cases based on Microsoft threat modeling and attack patterns. (Yuan et al., 2015)

Potential risks are examined using their technique by following Microsoft's threat modeling procedure. Initial abuse cases are prepared based on the detected dangers. A library of attack patterns, such as CAPEC, is searched, and attack patterns related to the abuse instances are obtained. The information obtained from the attack patterns is utilized to extend the first misuse scenarios and offer mitigation methods at this level. Such an approach has the potential to aid software developers without extensive knowledge of computer security in developing

relevant and valuable abuse scenarios, hence reducing security risks in the software systems they design.

(Horkoff, 2015) stated that CAPEC prioritizes the pragmatic development of security patterns with a thorough schema and categorization taxonomy in a comparison of numerous attack pattern repositories. It is gaining traction in academia and business because it delivers a large amount of practical security expertise. As a result, we picked CAPEC as the realistic attack knowledge source for our strategy.

Since our model is about control patterns, as shown in Fig 4.6 after identifying the attack strategy through the Antigoal process, we leverage on CAPEC repository to assist in identifying the existence of the attack pattern related to the attack strategy in order to assist in identification of mitigation strategies.

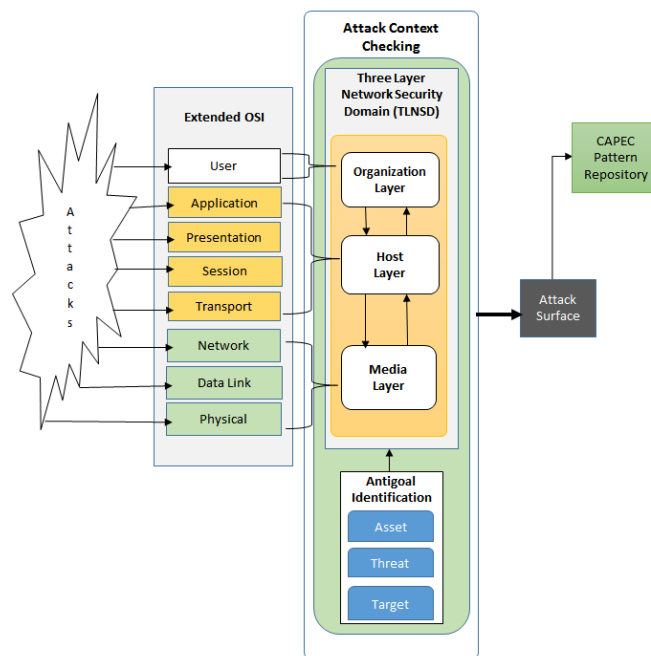


Figure 4. 6. Identifying existing attack pattern.

#### 4.3.3.2. Retrieving Relevant Attack Patterns Component.

The Common Attack Pattern Enumeration and Classification (CAPEC) hosts over 504 attack patterns (still growing), along with a comprehensive schema and classification taxonomy. CAPEC however, is not easy to use, since users have to go through the whole list to get the attack pattern they are looking for. This does not make using CAPEC for software

development attractive to developers, (Shostack, 2014b) also reports “the impressive size and scope of CAPEC may make it intimidating for people to jump in”.

In order to automatically associate attack patterns with requirements specifications and further derive security requirements, several researchers have attempted to address the scalability issue by defining term-maps, which link terms in requirements specifications to specific security terms used in CAPEC (Kaiya et al., 2014). In order to make it easier to navigate the many different attack patterns, (Engebretson & Pauli, 2009) augment the CAPEC attack patterns with the ideas of parent threat and parent mitigation. (Yuan et al., 2014a) provide a tool to make it easier to get CAPEC patterns by mapping CAPEC patterns to the STRIDE threat categories.

In order to help developers, find security flaws in their code, Gegick and Williams (2005) developed a set of 53 attack patterns. The four vulnerability databases served as the basis for the creation of these attack patterns. Regular expressions were employed to represent these attack patterns and contain the actions that can be taken to attack the program. These attack patterns were then utilized to find security holes by looking for a similar sequence of components in the system design. To counter this, we've developed misuse scenarios by mapping attack patterns from CAPEC to STRIDE.

Pauli and Engebretson (2008b) developed an approach for teaching attack patterns based on a hierarchy to present information logically. This hierarchy includes the following levels of abstraction from highest to lowest: vulnerability, attack pattern, exploit, bug and flaw, activation zone, injection vector, payload, and reward. Students were asked to map CAPEC Release 1 to the abstraction levels of this hierarchy. The objective of this work is to assist students to learn and retain information on attack patterns through the mapping process. They mapped attack patterns to abstraction level for teaching, we map attack patterns to STRIDE to retrieve relevant attack patterns from CAPEC.

Wiesauer and Sametinger (2009) developed a taxonomy for security design patterns using attack patterns. In their taxonomy, they described a criterion for selecting attack patterns based on security requirements. The purpose of the taxonomy was to help users see relevant security design patterns when selecting attack patterns. Their work assigned security design patterns to CAPEC attack patterns and employs the STRIDE model to group attacks into

different categories to classify security patterns, our work maps CAPEC attack patterns to STRIDE to identify Relevant attack patterns

McGraw (2006) mentioned in his book that attack patterns can be used for developing abuse cases, however, he did not discuss an approach to select and use relevant attack patterns for developing abuse cases. Our work introduces an approach for selecting and utilizing CAPEC attack patterns by mapping it to STRIDE.

For a mature business, the ability to rapidly implement a Threat Modeling strategy enables them to Identify, Protect, Detect, Respond, and Recover (Tatam et al., 2021). In their investigation, they determined that STRIDE is organized and well-documented, and that it stands out since so many resources were committed in its development and documentation.

(Yuan et al., 2014b) made a tool to find relevant CAPEC attack patterns for software development. This was done so that patterns could be found in a systematic way that fit the context. The tool can find the attack patterns that are best for a certain type of STRIDE and for the software that is being made. It can be used with the Microsoft SDL tool for threat modeling. It also lets developers use keywords to look for CAPEC attack patterns. At the moment, there is no clear mapping between CVE and CAPEC. (Kanakogi et al., 2021) suggests using TFIDF and Doc2Vec to automatically find the related CAPEC-IDs from the CVE-ID so that vulnerabilities can be fixed more quickly.

(Seehusen, 2015) developed a web-based tool (TrAP) for categorizing attack patterns and mapping them to STRIDE categories. The tool calculates a metric of for each attack pattern in terms of textual values of properties such as Severity, Completeness, Attacker Skills, and Likelihood of Exploit and uses it to rank each attack pattern according to each particular STRIDE category. The ranking puts the attack patterns most relevant to a particular STRIDE category at the top of the list of retrieved patterns.

From Figure 4.7 since an Attack strategy can be linked to several attack patterns within the CAPEC repository which can become too many and laborious to analyze we propose to restrict or rather pick only those that are mapped to STRIDE since is a technique widely use in retrieval of suitable patterns for threat modelling base on several studies highlighted earlier. With that in mind it would then ideally assist in identifying the applicable attack pattern per the attack domain which is the Three Layer Network Security Domain (TLNSD).

If from the attack strategy you are unable to identify the applicable patterns from the CAPEC repository the model creates an option of deriving alternative attack patterns. In this study we adopt the POSA template for deriving the alternative patterns. Once the patterns are derived it can be updated to attack pattern library for our case the CAPEC generating new knowledge. The same pattern can then be subjected to the CPAEC-STRIDE mapping process.

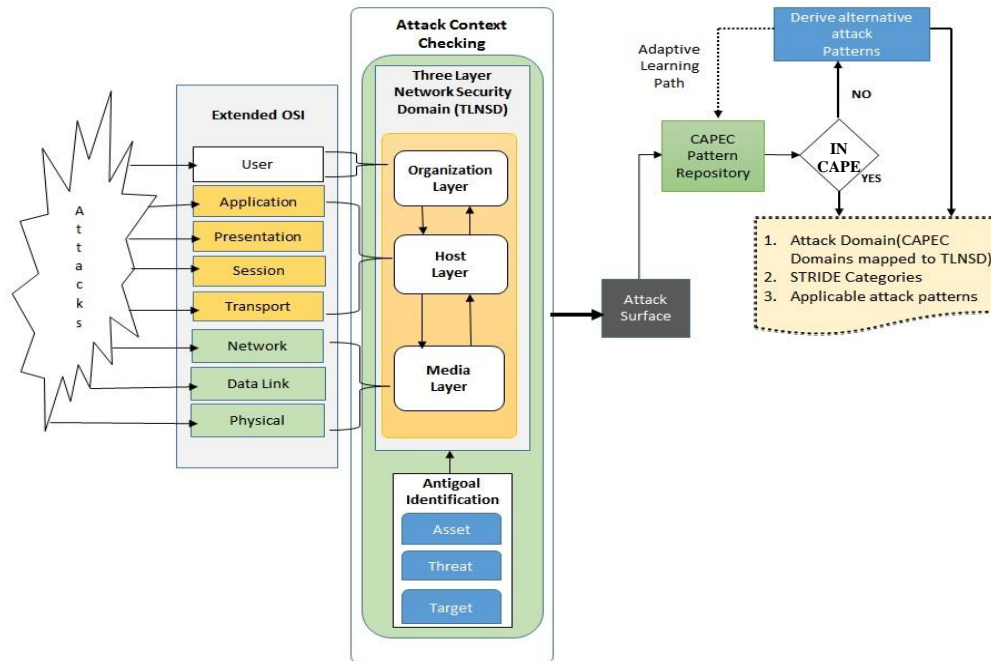


Figure 4. 7.Retrieving Relevant Attack Pattern and Deriving Alternative Attack Pattern.

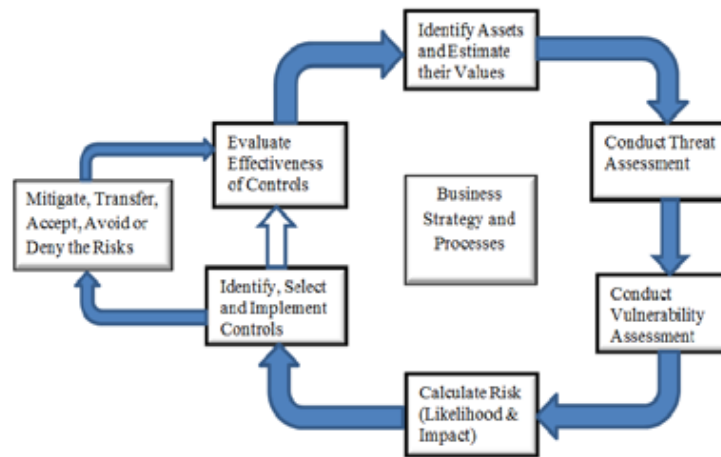
#### 4.3.4. Risk Assessment Component.

(Abuonji & Rodrigues, 2018) points out that effective security is built on good risk management. This is because organizations need to evaluate their risks and put in place the right security controls to help them deal with the risks they face. (Hewitt, 2020) says that risk management tries to reduce risk by recognizing the risks that are already there, figuring out how bad they are, and planning how to deal with them. A good risk management plan will help an organization set up ways to avoid possible threats, lessen their effects if they do happen, and deal with the results. (Hewitt, 2020) point out that there are three steps you can take to manage risk in your network security. First make a map of your network by listing the assets that cybercriminals might want to steal. second figure out what the network's risks are



and how they could affect your business. Third Plan for an attack by thinking about what you'll do if someone breaks into your network. How soon will you be able to figure out who is attacking?

According to (B. Cole, 2020), the risk management process consists of the following steps: setting the context of the Risk, identifying the Risk, analyzing the Risk, assessing and evaluating the Risk, mitigating the Risk, monitoring the Risk, communicating and consulting with stakeholders. This should help with questions like "what might go wrong?" " how would it effect the organization?" " what could be done if anything happened?" and "how would the organization pay for it?". The following seven phases make up the iterative process of risk management (Abuonji & Rodrigues, 2018). according to Figure 4.8.

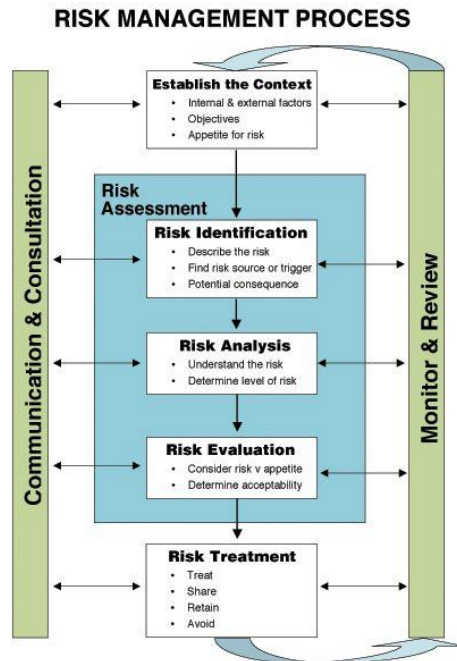


**Figure 4. 8.Risk Management Process**  
**Source:(Abuonji & Rodrigues, 2018)**

(1) Recognizing assets and calculating their values (2) Conducting threat assessments within the organization and its environment (3) Conducting assessments of vulnerabilities within the organization (4) Executing threat-vulnerability (T-V) pairing (5) Calculating risk in terms of likelihood of occurrence and impact on the information assets (6) Choosing and implementing the most appropriate security controls (7) Evaluating the efficacy of those controls that have been put in place.

The ISO 27005:2011 (ISO/IEC 27005:2011, 2018) is a standard that was developed by ISO to provide a guiding principle for security risk management. This standard is a general approach and does not specify or advocate for a particular methodology, but rather acts as a

base by which several security risk management methodologies have conformed or adopted. In this study we also adopt this general approach and not a specific methodology for risk management. The process model of this risk management approach is displayed in Figure 4.9.



**Figure 4. 9.**Risk Management process model  
(ISO/IEC 27005:2011(En), 2018)

Various studies have come up with conceptual models for risk management they include CORAS (Lund et al., 2010), EBIOS(Expression des Besoinset Identi\_cationdes Objectifs de Sécurité) (Iguer et al., 2013), MEHARI (MEthode Harmoniséed' Analyse du Risque Informatique) (Mihailescu, 2012), OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)(Jufri et al., 2017), CRAMM (CCTA Risk Analysis and Management Method) (Yazar, 2002), ISSRM (Altuhhova et al., 2012) among others, within this study adopted ISSRM to guide in conducting risk assessment since it contains the asset, risk and risk treatment concepts and it is in compliance with ISO/IEC 27005 standards , Figure 4.10 shows ISSRM process, while Figure 4.11 the meta model of ISSRM

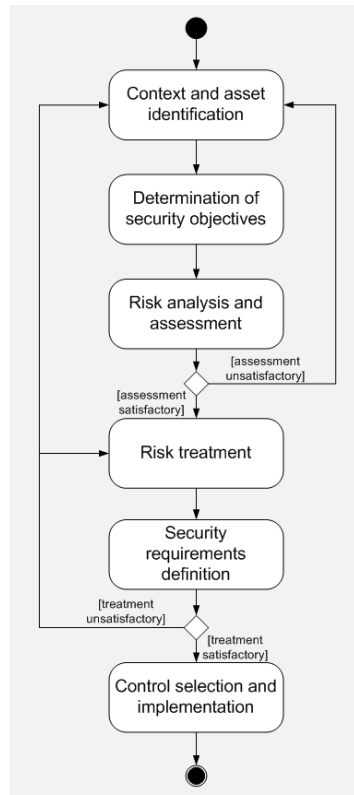


Figure 4. 10.ISSRM process (Mayer, 2009)

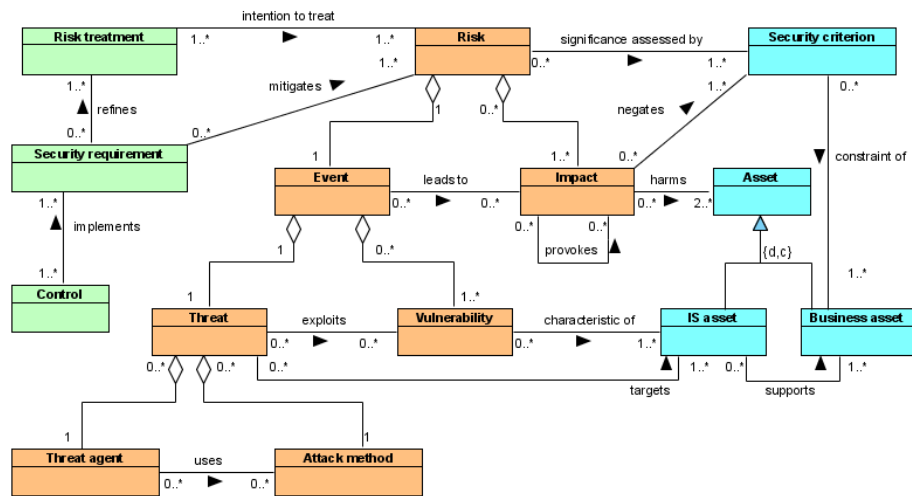


Figure 4. 11.Information System Security Risk Management (ISSRM) metamodel (Mayer et al., 2018)

### 4.3.5. Network Architecture Security Pattern Based Model

The final step in building is that once the relevant attack has been identified it is subjected to the risk assessment process at the risk management component this give rise to Figure 4.12 the final model structure, once the relevant attack has been. Information System Security Risk Management Risk management as previously seen identifies estimates and evaluates risks and delivers security requirements which in turns leads to control mechanisms, in our model once the requirements are identified it guides in the generation of the defensive control patterns in this process, we adopt the POSA approach to guide us in the process. Finally, the patterns generated are evaluated to see their contribution towards the security assurance of the network architecture.

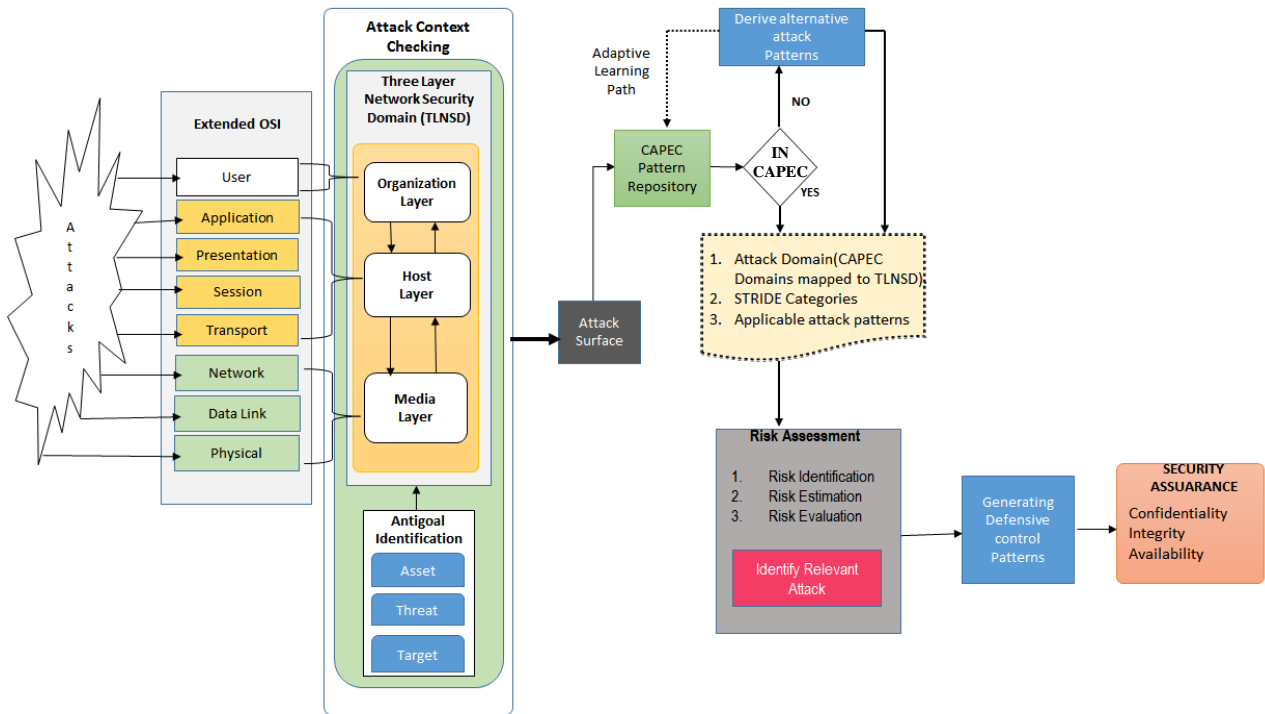


Figure 4.12. Network Architecture Security Pattern Based Model.

## **CHAPTER 5**

### **MODEL TESTING AND RESULTS**

#### **5.1. Model Testing**

The term "attack surface" is used to describe the entirety of an organization's or system's potential entry points for malicious hackers. Intruders can inflict damage to your network by manipulating or downloading data if they are able to get access to it, which is why it is important to secure all potential entry points. The easier it is to maintain security; the smaller your organization's attack surface may be. A surface analysis is a good first step in reducing or protecting your attack surface, and then implementing a strategic defensive strategy reduces the likelihood of an expensive attack or extortion attempt.(Atighetchi et al., 2014).

##### **5.1.1. First Stage of Model.**

The overall output of the model was to generate defensive control patterns of attacks within a network. To achieve this, we took the approach of splitting the model into three parts each with an input, process and output steps, with the output of a subsequent part acting as an input of latter part. Figure 5.1 shows the first part of the split whose output was the discovery of attacks on the network surface, the input in this instance was the network traffic that can be generated from a typical network from various devices such as intrusion detection systems, firewalls, and protocol analyzers among others. The traffic was subjected to attack context checking process in order to discover the attacks on the network surface.

In this model the attack surface consisted of three-layer domains that is the Media, Host and Organization/User which is christened (TLNSD). To achieve the stated output, the process part of the model subjected the network traffic through a machine learning process running on a python test bed.

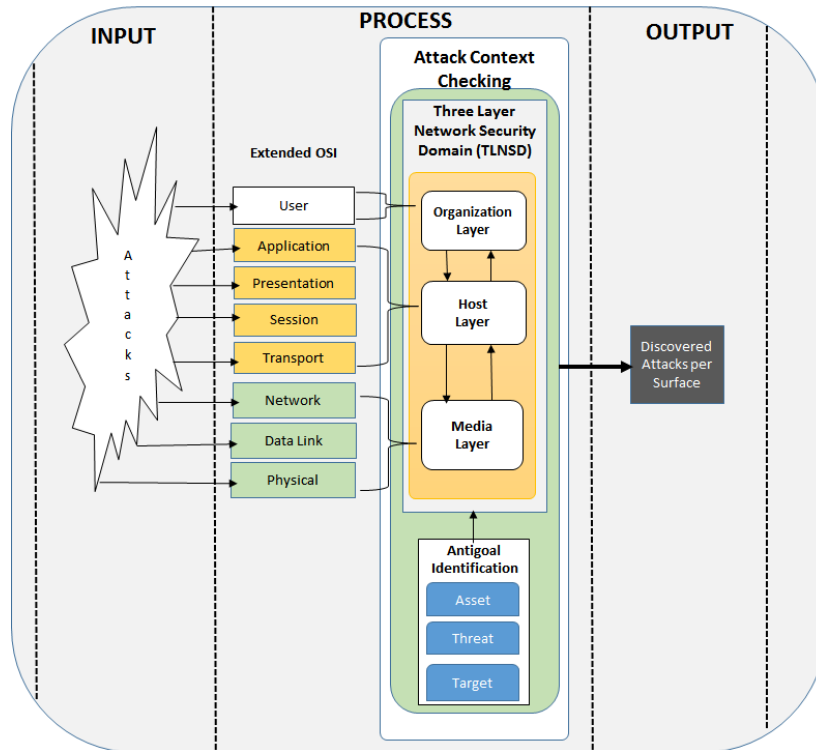


Figure 5. 1.First stage of the model

### 5.1.2. The Second Stage of the Model.

The second stage of the model as shown in Figure 5.2 also had the input, process and output components, the output was the identification of relevant attacks. The input part was the discovered attacks on the TLNSD which was an output of the first part of the split model. In the Process step the CAPEC repository is run through the Machine Learning process and a comparative analysis is done between it and the discovered surface attacks to check the existence of the discovered attacks within it, if discovered they are mapped according to STRIDE model categories and their applicable attack patterns, if the attacks are not identified within the repository then attack patterns are derived and updated within the repository which is also subsequently mapped according to STRIDE and applicable attack patterns.

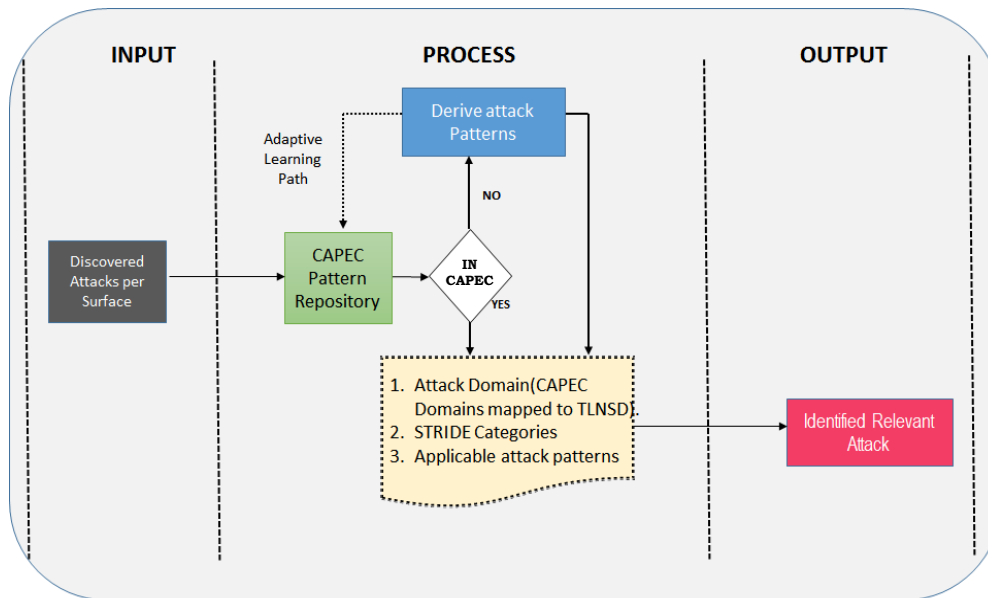


Figure 5. 2.Second Stage of the Model.

### 5.1.3. Third Stage of the Model.

The third stage of the model is depicted in Figure 5.3 has the overall output of the model. For it to be achieved the identified relevant attacks were subjected to risk analysis process. Risk analysis plays a critical role in identifying and recommending how to protect networks systems.

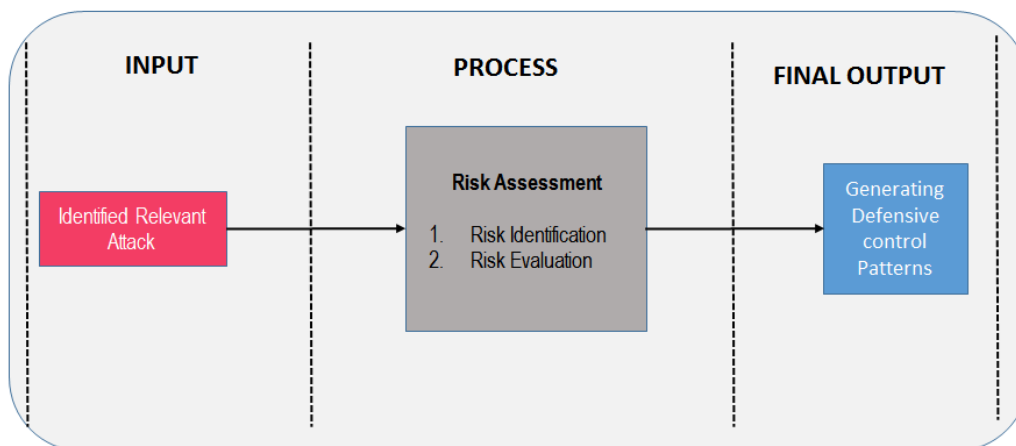


Figure 5. 3 Third Stage of the Model.

## 5.2. Study Data Set -UNSW-NB\_15

### 5.2.1. Dataset Description

The study used the UNSW-NB15 data set to test the model. The dataset is a hybrid of the real modern normal behaviors and the synthetically attack activities which was created in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) citation. Three tools were used to generate different aspects of this dataset they include tcpdump tool (tcpdump tool, 2014) used for capturing raw network traffic, IXIA Perfect Storm tool (IXIA Perfect Storm One Tool, 2014) used to generate, simulate a hybrid of normal and abnormal network traffic, the tool was also instrumental in extracting nine categories of attacks from the dataset as seen in Table 5.4, a simulation process of 36 hours produced 100GB of network traffic data which is split into sets of 1Giga Bits by the tcpdump tool, the Argus (Argus tool, 2014), BroIDS (BroIDS Tool, 2014) and twelve algorithms were used to extract 49 features from the dataset and also to analyze in detail the flows of connection packets. These techniques were configured in a parallel processing to extract 49 features with the class label. After finishing the implementation of the configured techniques, the total number of records, were 2,540,044 which were stored in four CSV files.

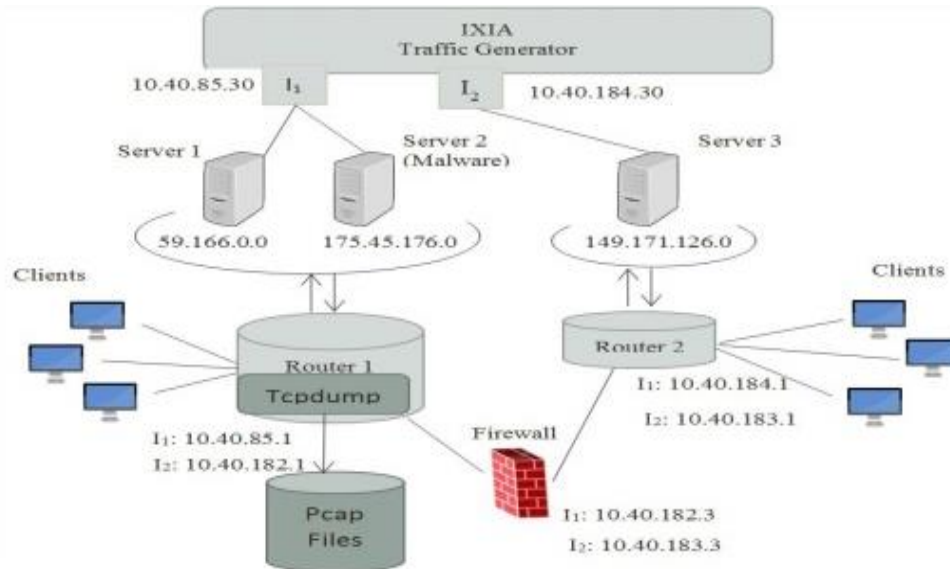
#	Name	Description	#	Name	Description	#	Name	Description	#	Name	Description
1	srrip	Source IP address	17	Spkts	Source to destination packet count	33	tcpttt	TCP connection setup round-trip time: the sum of 'synack' and 'ackdat'.	42	ct_srv_dst	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
2	sport	Source port number	18	Dpkts	Destination to source packet count	34	synack	TCP connection setup time, the time between the SYN and the SYN_ACK packets.	43	ct_dst_ltm	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
3	dstip	Destination IP address	19	swin	Source TCP window advertisement value	35	ackdat	TCP connection setup time, the time between the SYN_ACK and the ACK packets.	44	ct_src_ltm	No. of connections of the same source address (1) in 100 connections according to the last time (26).
4	dspport	Destination port number	20	dwin	Destination TCP window advertisement value	36	is_sm_ips_ports	If source (1) and destination (3)IP addresses equal and port numbers (2)/(4) equal then, this variable takes value 1 else 0	45	ct_src_sport_ltm	No. of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
5	proto	Transaction protocol	21	stcpb	Source TCP base sequence number	37	ct_state_ttl	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).	46	ct_dst_sport_ltm	No. of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
6	state	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)	22	dtcpb	Destination TCP base sequence number	38	ct_flow_http_mthd	No. of flows that has methods such as Get and Post in http service.	47	ct_dst_src_ltm	No. of connections of the same source (1) and the destination (3) address in 100 connections according to the last time (26).
7	dur	Record total duration	23	smeansz	Mean of the ?how packet size transmitted by the src.	39	is_ftp_login	If the ftp session is accessed by user and password then 1 else 0.	48	attack_cat	The name of each attack category. In this data set, nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
8	sbytes	Source to destination transaction bytes	24	dmeansz	Mean of the ?how packet size transmitted by the dst.	40	ct_ftp_cmd	No. of flows that has a command in ftp session.	49	Label	0 for normal and 1 for attack records
9	dbytes	Destination to source transaction bytes	25	trans_depth	Represents the pipelined depth into the connection of http request/response transaction	41	ct_srv_src	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).			
10	sttl	Source to destination time to live value	26	res_bdy_len	Actual uncompressed content size of the data transferred from the server's http service.						
11	dttl	Destination to source time to live value	27	Sjitt	Source jitter (mSec)						
12	sloss	Source packets retransmitted or dropped	28	Djitt	Destination jitter (mSec)						
13	dloss	Destination packets retransmitted or dropped	29	Stime	record start time						
14	service	http, ftp, smtp, ssh, dns, ftp-data, irc and (-) if not much used service	30	Ltime	record last time						
15	Sload	Source bits per second	31	Sintpkt	Source interpacket arrival time (mSec)						
16	Dload	Destination bits per second	32	Dintpkt	Destination interpacket arrival time (mSec)						

Figure 5. 4.UNSW-NB15 Dataset Description.

The testbed configuration utilized to produce the UNSWNB15 dataset is shown in Figure 5.5. This testbed's primary goal was to record any regular or anomalous traffic that was distributed among network nodes and started from the IXIA tool (e.g., servers and clients). It is significant to note that the IXIA tool was used to generate attack traffic in addition to regular traffic; the



attack behavior was sourced from the CVE website for the goal of accurately simulating a contemporary threat environment.



**Figure 5.5.** The Testbed Visualization for UNSW-NB15.  
(Moustafa & Slay, 2015)

On the IXIA traffic generator, three virtual computers were set up. Servers 1 and 3 were configured to handle normal traffic, but server 2 was designed to handle potentially harmful traffic. Two virtual interfaces with the IP addresses 10.40.85.30 and 10.40.184.30 connected servers and collected communications from both public and private networks. The servers and hosts were connected by two routers. IP addresses 10.40.85.1 and 10.40.182.1 were set up on Router 1, whereas 10.40.184.1 and 10.40.183.1 were set up on Router 2. These routers were connected to a firewall that was configured to let all traffic—normal and irregular—pass through. On router 1, the tcpdump program was installed in order to gather the Pcap files of the simulated uptime. The data set characteristics, including the simulation duration, flow numbers, total source bytes, total destination bytes, number of source packets, number of destination packets, protocol types, number of normal and abnormal traffic, and number of unique source and destination IP addresses, are described in Table 5.1.

Table 5. 1.Data Set Statistics

Statistical features		Simulation Period (31 hrs)
No._of_flows		1964509
Src_bytes		10800692594
Des_bytes		89046756452
Src_Pkts		82298235
Dst_pkts		105988377
Protocol types	TCP	1492153
	UDP	990144
	ICMP	524
	Others	524
Label	Normal	2218761
	Attack	321283
Unique	Src_ip	81
	Dst_ip	89

The whole architecture used to create the final shape of the dataset from Pcap data to CSV files with 49 features is depicted in Fig. 5.6. On the testbed depicted in Fig. 5.5, the simulation was running while Pcap files were being created using the tcpdump program. The Argus and Bro-IDS tools were used to obtain the properties of the data set, and a group of twelve algorithms were used to provide additional features for the dataset.

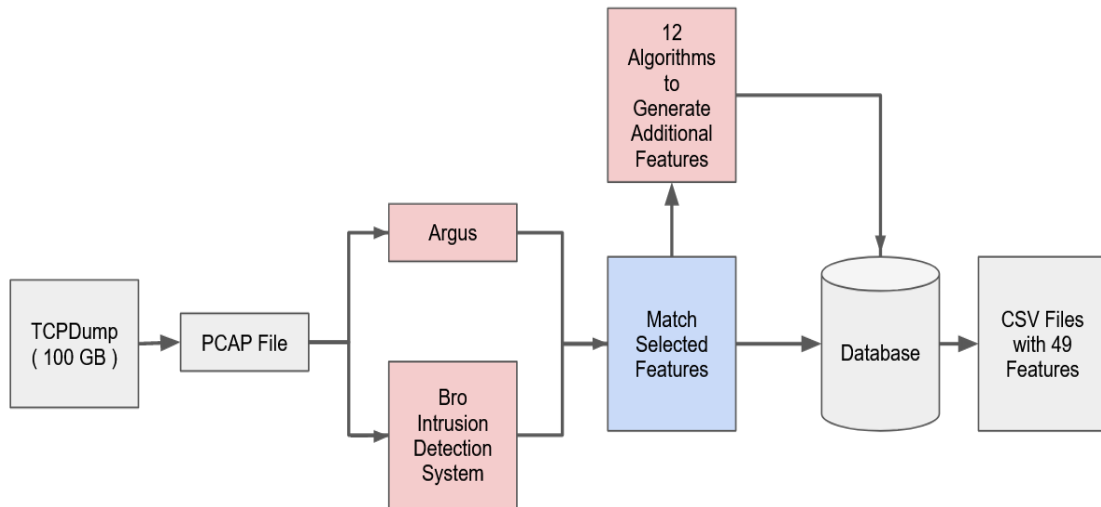


Figure 5. 6.Architecture for Generating UNSW-NB15 Data Set.

### 5.2.2. UNSW-NB15 Features Categories.

Features within the UNSW-NB15 dataset are categorized into seven types they include: Flow, Basic, Content, Time, General Purpose, Connection and Label. Table 5.2 gives an elaboration on the feature type their attributes within the dataset and its description.

**Table 5. 2.Feature Attributes.**

Feature Type	feature No.	Name of attributes	Description
Flow	(1-5)	Script, Sport, Dstip, Dsport, Proto	It contains the identifier attributes between hosts such as client-to-serve or server to-client
Basic	(6-18)	State, Dur, Sbytes, Dbytes, Sttl, Dttl, Sloss, Dloss, Service, Sload, Dload, Spkts, Dpkts	are attributes that characterize protocol connections
Content	(19-26)	Swin, Dwin, Stepb, Dtcpb, Smeansz, Dmeansz, trans_depth, res_bdy_len	These are TCP/IP attributes and http services attributes.
Time	(27-35)	Sjit, Djit, Stime, Ltime, Sintpkt, Dintpkt, Tcprtt, Synack, Ackdat	Has time related attributes such as arrival time, start/end time and round trips of packets?
General Purpose	(36-40)	is_sm_ips_ports, ct_state_ttl, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd	Here each features has its own purpose according to service protocol.
Connection	(41-47)	ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm	Built based on the chronological order of the last time feature
Label	(48-49)	attack_cat, Label	It represents the label of the record. Attack_cat represents the nine categories of the attack and the normal, and label is 0 for normal and otherwise 1.

### 5.2.3. UNSW-NB15 Data Types

Table 5.3 the elaborates on the counts of feature attributes and their respective data types

**Table 5. 3.Data Types.**

Data Type	Name	Count
Binary	is_sm_ips_ports, is_ftp_login, Label	3
Float	Dur, Sload, Dload, Sjit, Djit, Sintpkt, Dintpkt, tcprtt, synack, ackdat	10
integer	Sport, dsport, sbytes, dbytes, sttl, dttl, sloss, dloss, Spkts, Dpkts, swin, dwin, stcpb, dtcpb, smeanz, dmeanz, trans_depth, res_bdy_len, ct_state_ttl, ct_flw_http_mthd, ct_ftp_cmd, ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm	28
nominal	Srcip, dstip, proto, state, service, attack_cat	6
Timestamp	Stime, Ltime	2

### 5.2.4. UNSW-NB15 Attack Type Categories

Table 5.4 gives a glimpse of the types of attacks within the dataset and their description.

There are total of nine types of attack which include Analysis, DoS, Fuzzers, Exploit, Generic, Reconnaissance, Shellcode and Worm.

**Table 5. 4.Attack type categories**

<b>Attack Categories</b>	<b>Description</b>
Analysis	A method through which an attacker can get unauthorized remote access to a system by circumventing standard authentication and gain access to in information stored in databases and file servers.
DoS	An exploit under which the attacker is able to disrupt system resources by keeping them very busy in order to prevent authorized access to services and some extent causing it to fail or halt.
Fuzzers	An attack where the hacker attempts to find vulnerabilities in the system software, software applications, or network and suspend these resources for a set length of time.it can be automated to randomly feed multiple iterations of data into a program execution until one of those variants uncovers a vulnerability.
Exploit	Incursions or codes that take advantage of software security weaknesses, errors, or bugs they are usually included into malware, allowing for very simple and quick dissemination.
Generic	An attack that attempts to crack and gain access the secret keys within a cryptographic system
Reconnaissance	It is a probing attack, which collects information of the target computer network in order to circumvent its security controls.
Shellcode	A malware that infiltrates a small piece of code beginning with a shell to gain control of the infected system it usually manipulates program functions and registers.
Worm	Malware that replicates itself and moves to other systems through the network, based on the security weaknesses in the targeted system that it wishes to access. It consumes an excessive amount of system memory and network traffic which Reduces system availability

### **5.3. Machine Learning Testbed.**

As stated earlier to test the model three subparts of the original Model were run in Kaggle platform with python using the UNSW-NB15 network traffic dataset. Kaggle Machine learning platform was chosen in comparison to other platforms because, it has the necessary Python programming language libraries preinstalled, It has a an adequate amount of resources such as memory, CPU and storage space, It is a free and open source platform, It is a secure platform for your program can be made public to all people, private to the account holder or even the specific people he may authenticate to access the program(Przybyla, 2020) (Chugh, 2020).The system resources for the testbed included Intel Xeon CPU 2.0 CPU with a core count of 4vCPU, the RAM was 30GB and Hard disk of 73GB. See appendix 1 for more details

#### **5.3.1. Importation of Machine Learning Libraries and Modules**

Libraries are collections of commands and operations written in a particular language. A strong collection of libraries may help programmers complete complicated jobs faster and with fewer lines of code changes. They make "doing machine learning" easier for those who lack development experience. They lessen the processing time, tedium, and inefficiencies

caused by manually coding algorithms, mathematical formulas, and statistical equations. Appendix 1; Code Box1 shows the libraries and modules imported in machine learning platform.

### **5.3.2. Data Preprocessing**

Data preprocessing is a crucial step in preparing data for machine understanding, particularly in the context of real-world datasets prone to issues like missing, inconsistent, and noisy data. Such problems can significantly impact the performance of data mining techniques, leading to subpar results and distorted statistics. Duplicate or missing numbers may distort overall data statistics, while outliers and inconsistent data points can result in unreliable predictions. To enhance data quality, preprocessing involves understanding the data, visualizing it through statistical methods or libraries, and summarizing key aspects such as class distribution, duplicates, missing values, and outliers. Eliminating irrelevant fields and performing dimensionality reduction, including feature engineering to identify influential characteristics for model training, are integral components of the data preprocessing process.

### **5.3.3. Importation of Dataset**

This study involved the use of the UNSW\_NB15 dataset which was split into four files:

1. UNSW-NB15\_1.csv
2. UNSW-NB15\_2.csv
3. UNSW-NB15\_3.csv
4. UNSW-NB15\_4.csv

The importation was done using the *pd.read\_csv()* function which is usually used open various types of files such as excel files, database files and many other files to be analyzed. It also takes different kinds of parameters depending on the file(s) to be opened. In this instance, the parameter *low\_memory* was set to False to silence the error that comes up after opening very big files. The file is therefore opened at once without any error popping up provided that the file had the correct structure. As seen in Appendix 1; Code Box 2 four CSV files of the UNSW\_NB15 dataset that is UNSW-NB15\_1.csv was assigned variable un\_1, UNSW-NB15\_2.csv was assigned variable un2, UNSW-NB15\_3.csv was assigned variable un3 and UNSW-NB15\_4.csv was assigned variable un4. A quick run on the structure of the dataset using the *head()* function it was observed as seen in Table 5.5 that the columns do not have

the column header names. Due to the need for an organized structure of the dataset for analysis, feature selection and feature engineering process, the column headers were renamed to appropriately and renaming was guided by the dataset description document.

**Table 5. 5.Output of .head() on the dataset showing column headers with no name.**

	59.166.0.0	6055	149.171.126.5	54145	tcp	FIN	0.072974	4238	60788	31	...	0.6	13	13.1	6	7.1	1	1.1	2	Unnamed: 47	0.7
0	59.166.0.0	7832	149.171.126.3	5607	tcp	FIN	0.144951	5174	91072	31	...	0	13	13	6	7	1	1	2	NaN	0
1	59.166.0.8	11397	149.171.126.6	21	tcp	FIN	0.116107	2934	3742	31	...	1	1	2	7	5	1	1	4	NaN	0
2	59.166.0.0	3804	149.171.126.3	53	udp	CON	0.000986	146	178	31	...	0	13	13	6	7	1	1	2	NaN	0
3	59.166.0.8	14339	149.171.126.6	14724	tcp	FIN	0.038480	8928	320	31	...	0	8	20	7	5	1	1	4	NaN	0
4	59.166.0.8	39094	149.171.126.3	53	udp	CON	0.001026	130	162	31	...	0	8	13	6	5	1	1	1	NaN	0

5 rows × 49 columns

### 5.3.4. Renaming of Columns Headers.

The concept of list in python makes a good foundation that can lead to the renaming of columns that will be used in a dataset. To rename the column headers of the data frame the appropriate name for the columns were sought from the data description file. The main reason was to assist the researcher to merge the four datasets together and also to be able to manipulate the data to desired output and results based on column headers. Appendix 1; Code Box 3 shows the python code listing of *col* variable holding the column headers, Appendix 1; Code Box 4 shows the column headers assigned to the four individual dataFrames, Table 5.6 shows the resultant output(renaming) of with the assigned column header names.

**Table 5. 6.Dataset with New Column Headers.**

	59.166.0.0	1390	149.171.126.6	53	udp	CON	0.001055	132	164	31	...	0.17	3	7	1	3.1	1.1	1.2	1.3	Unnamed: 47	0.18
0	59.166.0.0	33661	149.171.126.9	1024	udp	CON	0.036133	528	304	31	...	0	2	4	2	3	1	1	2	NaN	0
1	59.166.0.6	1464	149.171.126.7	53	udp	CON	0.001119	146	178	31	...	0	12	8	1	2	2	1	1	NaN	0
2	59.166.0.5	3593	149.171.126.5	53	udp	CON	0.001209	132	164	31	...	0	6	9	1	1	1	1	1	NaN	0
3	59.166.0.3	49664	149.171.126.0	53	udp	CON	0.001169	146	178	31	...	0	7	9	1	1	1	1	1	NaN	0
4	59.166.0.0	32119	149.171.126.9	111	udp	CON	0.078339	568	312	31	...	0	2	4	2	3	1	1	2	NaN	0

5 rows × 49 columns

### 5.3.5. Boolean Filtering of Malicious Logs

In this study, the portion of the dataset that was of interest was the malicious part. From the description document of the UNSW\_NB15 dataset, the malicious logs were indicated by the value *1* in the *Label* column. To filter the malicious logs from the dataset, the malicious logs with the label of *1* were filtered from the four datasets. The filtering process was done resulting

in the DataFrames *mal\_1*, *mals2*, *mals3* and *mals4* representing the malicious parts of dataset *un\_1*, *un2*, *un3* and *un4* respectively. Appendix 1: Code Box 5, Code Box 6, Code Box 7, Code Box 8 shows the filtering process on the malicious dataFrames. The *un1* dataset had malicious logs extracted from it. Table 5.7 shows sample malicious log.

**Table 5.7. Sample of malicious logs from un\_1**

	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ltm
87137	175.45.176.1	0	149.171.126.12	0	unas	INT	0.000004	200	0	254	...	0	11	11	2	6
180941	175.45.176.0	64610	149.171.126.18	8080	tcp	FIN	0.418539	1514	1554	254	...	0	2	3	1	1
123245	175.45.176.1	1043	149.171.126.18	53	udp	INT	0.000005	114	0	254	...	0	12	12	5	5
142562	175.45.176.0	54070	149.171.126.13	110	tcp	FIN	0.760984	1612	436	254	...	0	1	1	1	1
89691	175.45.176.1	1043	149.171.126.18	53	udp	INT	0.000003	114	0	254	...	0	7	7	3	3

5 rows x 49 columns

The *un2* dataset had malicious logs extracted from it. Table 5.8 shows sample malicious log.

**Table 5.8. Sample of malicious logs from un\_2**

	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ltm
680587	175.45.176.3	47439	149.171.126.18	53	udp	INT	0.000003	114	0	254	...		18	18	18	18
681869	175.45.176.3	1043	149.171.126.18	53	udp	INT	0.000005	114	0	254	...		43	43	43	43
652833	175.45.176.3	0	149.171.126.19	0	trunk-2	INT	0.000003	200	0	254	...		5	5	2	2
661887	175.45.176.3	0	149.171.126.19	0	unas	INT	0.000009	200	0	254	...		5	5	1	1
472843	175.45.176.3	53330	149.171.126.19	111	udp	INT	0.000008	168	0	254	...		3	1	1	2

5 rows x 49 columns

The *un3* dataset had malicious logs extracted from it. Table 5.9 shows sample malicious log.

**Table 5.9. Sample of malicious logs from un\_3**

	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ltm
222485	175.45.176.1	1043	149.171.126.18	53	udp	INT	0.000009	114	0	254	...		41	41	21	22
237865	175.45.176.1	1043	149.171.126.18	53	udp	INT	0.000005	114	0	254	...		53	53	35	35
302540	175.45.176.1	47439	149.171.126.18	53	udp	INT	0.000002	114	0	254	...		38	38	7	7
218469	175.45.176.1	0	149.171.126.10	0	prm	INT	0.000009	200	0	254	...		13	13	4	6
159834	175.45.176.3	26210	149.171.126.14	179	tcp	FIN	0.498492	810	268	254	...		14	7	4	5

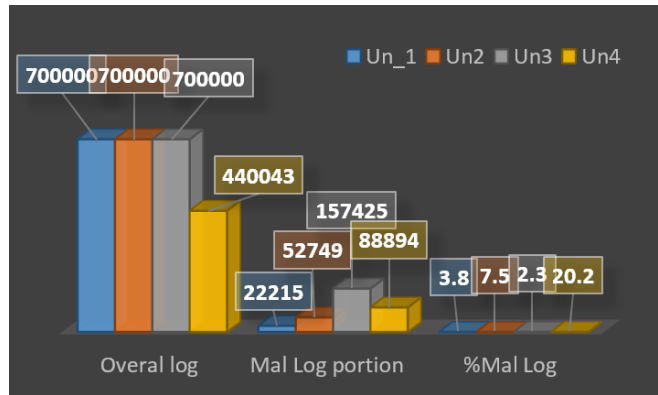
5 rows x 49 columns

The *un4* dataset had malicious logs extracted from it. Table 5.10 shows sample malicious log.

**Table 5. 10. Sample of malicious logs from un\_4**

	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ltm
205661	175.45.176.1	1043	149.171.126.14	53	udp	INT	0.000008	114	0	254	...		33	33	16	16
425457	175.45.176.0	1043	149.171.126.10	53	udp	INT	0.000009	114	0	254	...		17	17	19	18
250085	175.45.176.0	33452	149.171.126.10	65535	udp	INT	0.000003	656	0	254	...		6	2	2	3
60743	175.45.176.1	47439	149.171.126.14	53	udp	INT	0.000003	114	0	254	...		25	25	18	18
412614	175.45.176.1	5137	149.171.126.17	25	tcp	FIN	2.836784	54963	2002	62	...		1	1	1	2

5 rows x 49 columns



**Figure 5. 7. Percentage of malicious log in Un\_1 Un2 Un3 and Un4.**

Figure 5.7 shows the percentage of the malicious logs within the four dataset *un\_1* at 3.8%, *un2* at 7.5%, *un3* at 2.3% and *un4* at 20.2% respectively.

### 5.3.6. Merging the Dataset.

Since the study adopted a full census approach it was required to utilize all the four datasets, this necessitated the merging of the datasets. The malicious logs from the four datasets were also merged creating the *un1* dataset which stored the cumulative malicious log as seen in Appendix 1; Code Box 9., a quick run on the shape of the *un1* dataset gives a total of total of 321,283 rows and 49 columns as shown in Appendix 1; Code Box 10.

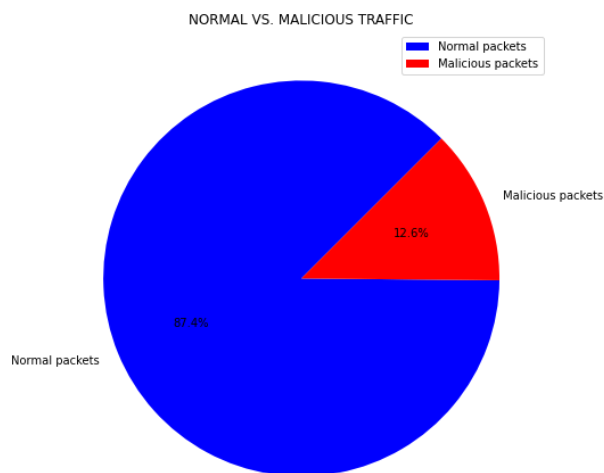
## 5.4. Analysis of the Malicious Combined Dataset

### 5.4.1. Malicious Count Within the Dataset

In the over 2.5 million rows in the combined dataset, 2.2 million rows were non-malicious while 321,283 rows were malicious. The differentiating factor between the malicious and the clean logs was that in the *Labels* column, the malicious rows had a value of one and the clean



datasets had a value of zero. The malicious rows contributed to about 12.6% of the dataset which is illustrated in the Figure 5.8.



**Figure 5.8. Distribution of Normal vs. Malicious Traffic.**

#### **5.4.2. Identification of Missing Values.**

Handling missing values in real-world data is crucial due to various factors such as data capture issues and corruption. Many machine learning algorithms do not handle missing values well, making proper data preparation essential. Several methods exist for addressing missing values, including deleting rows or columns with null values, imputing missing values for categorical variables, and using algorithms like k-NN, Naive Bayes, and random forest that support missing values. Predicting missing values based on other features in the dataset is another approach, as is using deep learning libraries like datawig for imputation through Deep Neural Networks.

Each method has its own advantages and drawbacks, and there's no one-size-fits-all solution. The choice of approach depends on achieving a robust model with optimal performance, considering the nature of the data and collection methods. A background understanding of the dataset is essential for effective preprocessing and handling missing values. Ultimately, the strategy adopted should align with the characteristics of the data, and different methodologies may be applied accordingly Appendix 1: Code Box 11 was used to check for missing values. From the Table 5.11 it was established that only two out of the 49 columns had missing values.

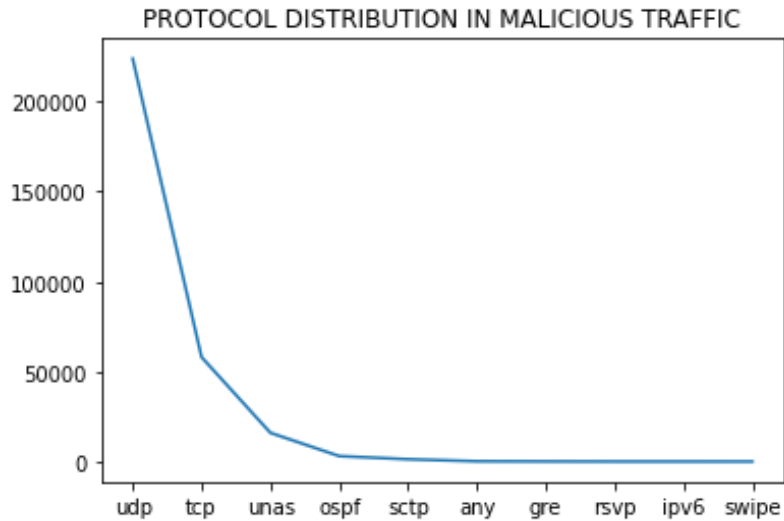
The two columns were *ct\_flw\_http\_mthd* and *is\_ftp\_login* with 282939 and 297183 missing values respectively.

**Table 5. 11. Missing Values in The Malicious Dataset.**  
no of missing values

	no of missing values
<i>ct_flw_http_mthd</i>	282939
<i>is_ftp_login</i>	297183

### 5.4.3. Protocol Distribution in the Malicious Traffic

We set out to look at the protocols in the malicious traffic as seen in Appendix 1; Code Box 12 a total of 129 protocols were found to have been used to perpetuate malicious activities within the network. Figure 5.9 shows the top ten significant counts of the various malicious protocols the lead was taken by UDP with 223750 instances of attacks and the tenth was swipe at 262 attacks.



**Figure 5. 9. Top Ten Malicious Protocol Count.**

#### 5.4.3.1. Top Three Malicious Protocols

From Figure 5.9 three protocols *udp*, *tcp*, and *unas* show a highly significant instances of malicious activities ranging from 22370, 58184 and 16202 respectively the other protocols a low number of instances ranging from 3000 to 272. Each of the top malicious protocols were classified into its own unique dataframe, as seen in Appendix 1: Code Box 13 all the malicious

TCP protocols were placed in the *mal\_tcp* dataset the UDP in *mal\_udp* dataset and unas in *mal\_unas* dataset. A filtering of the three produced the following samples of TCP UDP and UNAS malicious traffic in Tables 5.12, 5.13, and 5.14

**Table 5. 12.Sample Filtering malicious TCP traffic.**

	srcip	sport	dstip	dport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ltm	ct_src_dport_ltm	ct_dst_sport_ltm
20	175.45.176.2	23357	149.171.126.16	80	tcp	FIN	0.240139	918	25552	62	...	0	3	2	2	1	1	1
21	175.45.176.0	13284	149.171.126.16	80	tcp	FIN	2.390390	1362	268	254	...	0	5	2	2	1	1	1
38	175.45.176.2	13792	149.171.126.16	5555	tcp	FIN	0.175190	8168	268	254	...	0	1	1	1	1	1	1
39	175.45.176.2	26939	149.171.126.10	80	tcp	FIN	0.190600	844	268	254	...	0	3	1	1	1	1	1
56	175.45.176.0	39500	149.171.126.15	80	tcp	FIN	0.177449	1214	268	254	...	0	5	2	1	1	1	1

**Table 5. 13.Sample Filtering malicious UDP traffic.**

	srcip	sport	dstip	dport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ltm	ct_src_dport_ltm
19	175.45.176.3	21223	149.171.126.18	32780	udp	INT	0.000021	728	0	254	...	0	1	1	1	1	1
109	175.45.176.3	22359	149.171.126.11	111	udp	INT	0.000017	168	0	254	...	0	2	1	1	1	1
241	175.45.176.3	55249	149.171.126.10	5060	udp	INT	0.000019	1388	0	254	...	0	1	1	1	1	1
245	175.45.176.0	36159	149.171.126.14	111	udp	INT	0.000012	168	0	254	...	0	1	1	1	1	1
599	175.45.176.2	29442	149.171.126.19	111	udp	INT	0.000017	168	0	254	...	0	1	1	1	1	1

**Table 5. 14.Sample Filtering malicious Unas traffic.**

	srcip	sport	dstip	dport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ltm	ct_src_dport_ltm
84913	175.45.176.1	0	149.171.126.12	0	unas	INT	0.000035	200	0	254	...	0	12	12	6	6	6
84914	175.45.176.1	0	149.171.126.12	0	unas	INT	0.000035	200	0	254	...	0	12	12	6	6	6
84915	175.45.176.1	0	149.171.126.12	0	unas	INT	0.000035	200	0	254	...	0	12	12	6	6	6
84916	175.45.176.1	0	149.171.126.12	0	unas	INT	0.000035	200	0	254	...	0	12	12	6	6	6
84917	175.45.176.1	0	149.171.126.12	0	unas	INT	0.000035	200	0	254	...	0	12	12	6	6	6

For ease of analysis of attacks the high significant instances protocols and low significant were merged and grouped differently. Appendix 1: Code Box 14 shows the merging process of the three top malicious protocols were stored into the *malpt* dataframe. Table 5.15 shows the sample output of the merging process.

**Table 5. 15. Output of the top 3 significant malicious protocols.**

	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd
20	175.45.176.2	23357	149.171.126.16	80	tcb	FIN	0.240139	918	25552	62	...	0
21	175.45.176.0	13284	149.171.126.16	80	tcb	FIN	2.390390	1362	268	254	...	0
38	175.45.176.2	13792	149.171.126.16	5555	tcb	FIN	0.175190	8168	268	254	...	0
39	175.45.176.2	26939	149.171.126.10	80	tcb	FIN	0.190600	844	268	254	...	0
56	175.45.176.0	39500	149.171.126.15	80	tcb	FIN	0.177449	1214	268	254	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
303248	175.45.176.2	0	149.171.126.17	0	unas	INT	0.000003	200	0	254	...	
303249	175.45.176.2	0	149.171.126.17	0	unas	INT	0.000003	200	0	254	...	
303250	175.45.176.2	0	149.171.126.17	0	unas	INT	0.000003	200	0	254	...	
303255	175.45.176.2	0	149.171.126.17	0	unas	INT	0.000003	200	0	254	...	
303256	175.45.176.2	0	149.171.126.17	0	unas	INT	0.000003	200	0	254	...	

298136 rows × 49 columns

Then a Boolean filter was done to extract the logs that were in the malicious DataFrames *unil* but not in the *tmalpt* dataframe this was done so as to get the low significant malicious protocols which the output was stored in *nunas* frame as seen in Appendix 1: Code Box 15. The Table 5.16 bellow shows the sample output of 126 low significant malicious protocols extracted from the Boolean process.

**Table 5. 16. Low significant malicious protocols.**

	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src	ct_srv_dst
173319	175.45.176.3	0	149.171.126.17	0	sun-nd	INT	0.000015	60	0	0	...		4	4
468970	175.45.176.1	0	149.171.126.17	0	iso-tp4	INT	0.000005	200	0	254	...		6	6
471091	175.45.176.1	0	149.171.126.17	0	aes-sp3-d	INT	0.000009	200	0	254	...		9	9
42893	175.45.176.0	0	149.171.126.17	0	pim	INT	0.000009	200	0	254	...		4	4
654327	175.45.176.3	0	149.171.126.19	0	any	INT	0.000006	200	0	254	...		3	3
445646	175.45.176.0	0	149.171.126.13	0	scps	INT	0.000003	200	0	254	...		5	5
222193	175.45.176.1	0	149.171.126.10	0	vrrp	INT	0.000003	200	0	254	...		4	4
651923	175.45.176.3	0	149.171.126.19	0	cbt	INT	1.051516	360	0	254	...		10	10
489412	175.45.176.2	0	149.171.126.11	0	ospf	INT	5.052801	2688	0	254	...		1	1
442392	175.45.176.0	0	149.171.126.13	0	prm	INT	0.000008	200	0	254	...		6	6

In order to have a cumulated analysis on the remaining protocols, they were all renamed to *others* as shown in Appendix 1: Code Box 16. Table 5.17 shows the sample output obtained

**Table 5. 17.Other Protocols.**

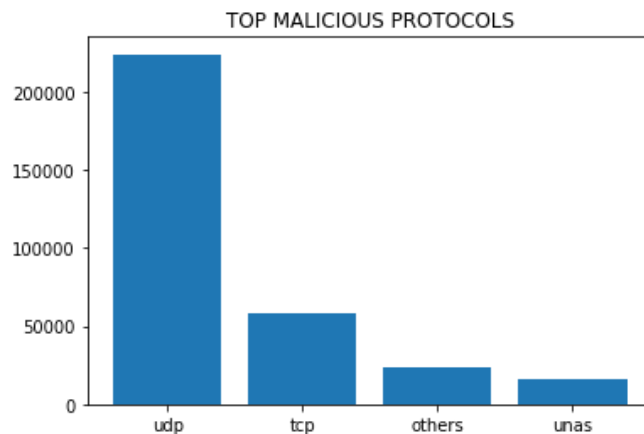
	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	...	ct_ftp_cmd	ct_srv_src
468848	175.45.176.1	0	149.171.126.17	0	others	INT	0.000008	200	0	254	...		6
2446	175.45.176.3	0	149.171.126.18	0	others	INT	44.300476	12832	0	254	...		1
504150	175.45.176.0	0	149.171.126.15	0	others	INT	0.000010	152	0	254	...		1
657464	175.45.176.3	0	149.171.126.19	0	others	INT	0.000009	200	0	254	...		2
472171	175.45.176.1	0	149.171.126.17	0	others	INT	0.000007	200	0	254	...		14

A summary of value count within the *tmalpt* and *nunas* frames was done as illustrated in Appendix 1: Code Box 17 and Table 5.18 shows a new shape and the protocol distribution

**Table 5. 18. Protocol Distribution of Tmalpt and Nunas.**

proto	
udp	223750
tcp	58184
others	23147
unas	16202

The Figure 5.10. shows the distribution of the malicious protocols.



**Figure 5. 10. Distribution of Malicious Protocol.**

The pie chart in Figure 5.11 shows the percentage distribution of the malicious protocols.

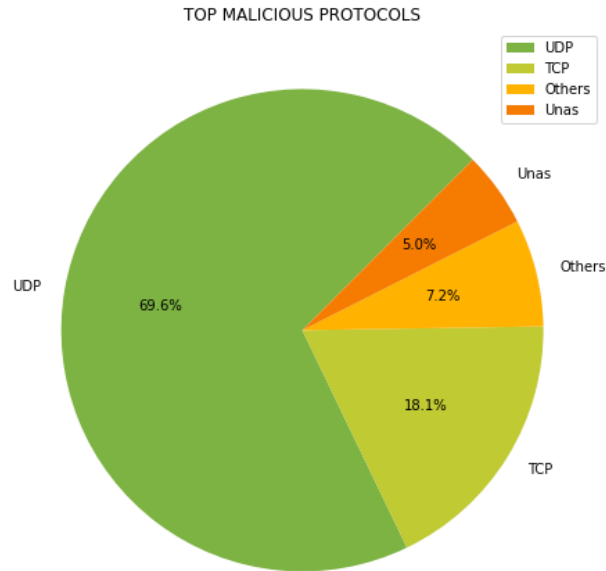


Figure 5. 11. % distribution of malicious protocol

#### 5.4.4. Attack Distribution.

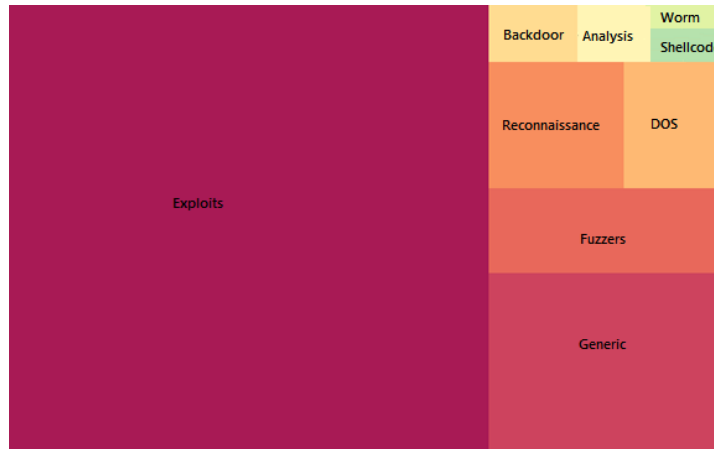
In attack distribution analysis we went out to look for the following aspects within the dataset, in terms of categories of attacks what percentage does each attack constitute within the traffic! What is the distribution of attack categories in relation to the protocols? A treemap visualization of the malicious dataset with the aid of squarify library was used as shown in Appendix 1: Code Box 18.

Table 5. 19. Percentage of Attack Distribution within the Malicious Traffic.

attack_cat	attack count%
Exploits	67.068908
Generic	13.858499
Fuzzers	5.974484
DoS	5.089905
Reconnaissance	3.805990
Fuzzers	1.572134
Analysis	0.833222
Backdoor	0.558697
Reconnaissance	0.547492
Shellcode	0.400893
Backdoors	0.166209
Shellcode	0.069409
Worms	0.054158

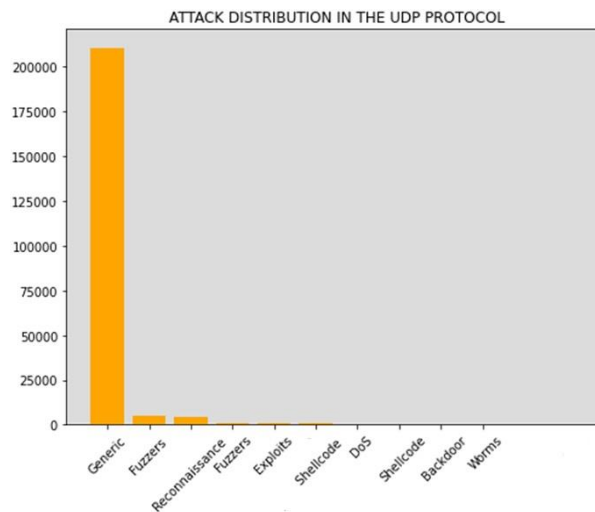
Table 5.19 the percentage attack counts the top attack in the list of attacks was Exploits 67.069%, Generic 13.858%, Fuzzers 7.547%, Reconnaissance 4.353%, DOS 5.089%, Backdoors 2.40%, Analysis 0.724%, Shellcode 0.470 %, Worms 0.054%

Figure 5.12 shows the tree map visualization of the the attack distribution this was generated using using the python code as shown in Appendix 1: Code Box 19.



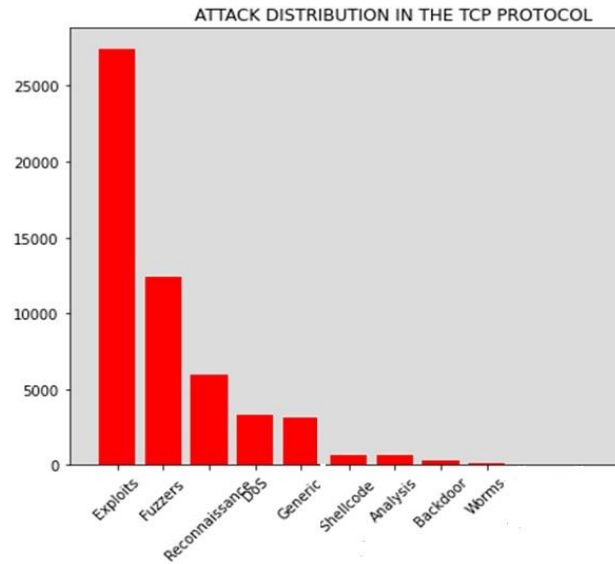
**Figure 5.12. Treemap Visualization of Attack Distribution.**

Within the UDP protocol as seen in Fig 5.13. the following was the distribution of attacks, the Generic attacks took the lions share with 210600 (94%), followed by Fuzzers at 6043(2.701%), Reconnaissance at 4890(2.185%), Exploits 874(0.391%), Shellcode 761 (0.340%), DoS 527(0.235%), Backdoors 34(0.015%) and Worms were the least at 21 (0.009%).



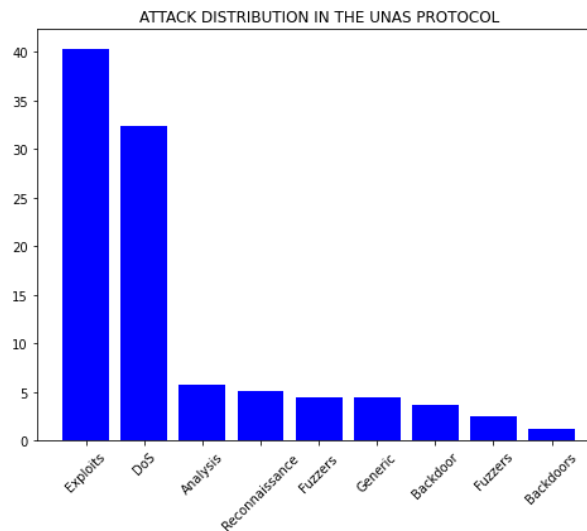
**Figure 5.13. Attack Distribution UDP Protocol.**

In relation to the TCP protocol the distribution of attacks as seen in Figure 5.14 exploits attacks were leading at 27443(47.196%) instances, Fuzzers at 15474 (26.612%), Reconnaissance at 6965 (11.978%), DoS at 3336 (5.737%), Generic at 3118 (5.362%), Shellcode was 750(1.289%), Analysis 622(1.069%), Backdoor 323(0.492%) Worms153 (0.263%).



**Figure 5. 14. Attack distribution TCP protocol.**

In relation to the UNAS protocol the Distribution of attacks as shown in the Figure 5.15 were observed as follows, the exploits were leading with 40.32%, DoS 32.38%, Analysis 5.715%, Fuzzers 6.987%, Reconnaissance 5.185%, Backdoor 4.975%, Generic 4.443%, DoS 2.542%, and Backdoor at 1.271%.



**Figure 5. 15. Attack Distribution UNAS Protocol**



When looking at other protocols the distribution of attacks was as follows in Figure 5.16 exploits 9676 (42.872%), Dos 7244(32.097), reconnaissance 1292(5.724%), Backdoor 1166(5.166%), analysis 1129(5.002%), Generic 1043(4.621%) and Fuzzers 1019 (4.515%). There were no attacks recorded for Shellcode and worms categories with other protocols.

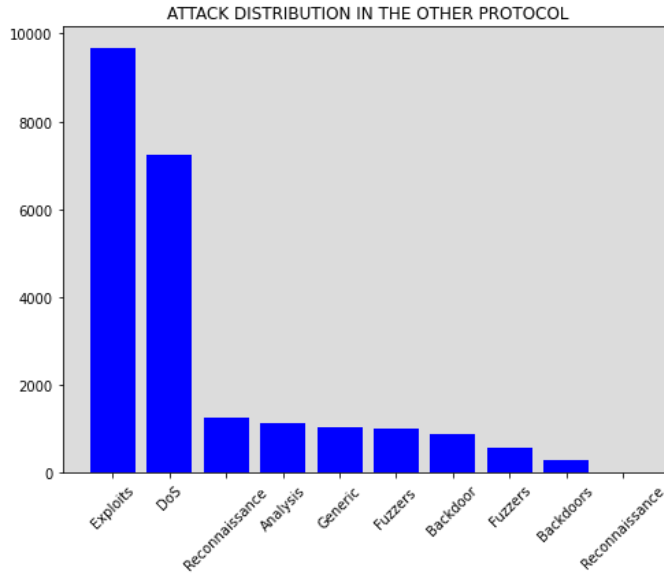


Figure 5. 16. Attack Distribution Other Protocol.

#### 5.4.5. Targeted Ports Per Attack

From the malicious dataset a total of 1405 different port numbers were a target of attacks, and the cumulative frequency of attacks on the same was 27443 Table 5.20 below shows a sample of the ports with at least 100 and above attacks. Port 80 the http service had 11751 (42.820%) attacks, port 25 the smtp service 4427 (16.132%) attacks, port 21 for ftp service 2206 (8.03848%), port 110 for POP3 1558 5.677222 attacks, port 143 for IMAP services had 1547 (5.637139%) attacks, port 3306 for MySQL services had 434(1.58146%) attacks, port 8080 for http services had 336(1.224356%) attacks, port 139 for NetBIOS services had 257 (0.936487%) attacks, port 445 for SMB Active Directory services 173 (0.630398%) attacks, port 23 for telnet services had 135 (0.491929%) attacks. Appendix 1: Code Box 20 shows the code used for this purpose.

**Table 5. 20. Sample of Ports with 100 Attacks and Above.**

	port	attack_counts
0	80	11751
1	25	4427
2	21	2206
3	110	1558
4	143	1547
5	3306	434
6	8080	336
7	139	257
8	445	173
9	23	135

General statistics of the malicious logs per protocol is given in Table 5.21

**Table 5. 21. Statistics of the Malicious Logs per Protocol**

	TCP log timings	UDP log timings	UNAS log timings	Overall log timings
Count	58184.00	223750.0	16202.00	321283.00
Mean	1.946001	0.103501	0.000007	0.730478
Standard deviation	5.584161	2.286526	0.000003	4.866499
Minimum value	0.000000	0.000000	0.000000	0.000000
25 <sup>th</sup> percentile	0.527969	0.000004	0.000004	0.000005
50 <sup>th</sup> percentile	0.857681	0.000008	0.000008	0.000009
75 <sup>th</sup> percentile	1.505891	0.000009	0.000009	0.000010
Maximum value	59.999527	59.99602	0.000035	59.999527

#### 5.4.6. Main Dependency Protocols Timings Per State

In the analysis below Table 5.22 establishes the time used per attack state in the top protocols in the UNSW-NB\_15 dataset.

**Table 5. 22. Main dependency protocols timings per state**

TCP protocol		UDP protocol		UNAS protocol		Other Protocols	
State	Duration	State	Duration	State	Duration	State	Duration
Fin	104221	Req	12200.2	Int	0.1123	Int	53164.9
Con	8934.07	Int	9100.86			Req	40582.7
Rst	69.8937	Con	1857.36			Con	4557.99
Acc	0.972						
Clo	0.6167						

### 5.4.7. Mean Of Time Distribution in Malicious Packets

Table 5.23 shows the mean of Time distribution within the TCP, UDP, UNAS and OTHER protocols.

**Table 5. 23. Mean of Time Distribution in Malicious Packets.**

TCP protocol		UDP protocol		UNAS protocol		Other Protocols	
State	Duration	State	Duration	State	Duration	State	Duration
Con	36.4656	Req	38.0068	Int	0.000007	Req	44.6946
Rst	5.3764	Con	3.1968			Con	6.8131
Fin	1.7992	Int	0.0408			Int	2.4648
Clo	0.6167						
Acc	0.485						

### 5.4.8. Attack Category Time Duration in The Malicious Dataset

Appendix 1: Code Box 21 was used to extract duration of attack of each attack category within the malicious dataset was extracted. Table 5.24 shows the total time duration taken by each category of attack.

**Table 5. 24. Attack Category Time Duration in the Malicious Dataset.**

Attack Category	Duration of Attacks
Exploits	94522.20123
Fuzzers	64832.28926
DoS	40045.14186
Reconnaissance	14057.57736
Generic	11478.62264
Backdoor	5180.99554
Analysis	3793.153263
Shellcode	548.274819
Worms	77.858737

### 5.4.9. Attack Category Time Duration For TCP, UDP, UNAS And Others.

Appendix 1: Code Box 22 shows the attack category time duration of the three significant protocols TCP, UDP, UNAS and Other protocols attacked within the malicious dataset. The table 5.25 shows the time duration of attacks of category in relation to the three

**Table 5. 25. Attack category time duration for TCP UDP UNAS.**

TCP protocol		UDP protocol		UNAS protocol		Other Protocols	
Attack Cat	Time Duration of Attacks	Attack Cat	Time Duration of Attacks	Attack Cat	Time Duration of Attacks	Attack Cat	Time Duration of Attacks
Exploits	48697.2621	Fuzzers	21032.7386	Exploits	0.04507	Fuzzers	6065.89847
Fuzzers	37733.3631	DoS	1175.86599	DoS	0.03631	Reconnaissance	7101.10226
DoS	11429.0327	Exploits	718.41175	Analysis	0.00645	Analysis	2136.27677
Reconnaissance	6956.18749	Generic	219.626311	Reconnaissance	0.00583	Backdoors	4854.04504
Generic	6598.62073	Backdoors	11.184046	Generic	0.00496	DoS	28381.4429
Analysis	1656.87004	Worms	0.000156	Fuzzers	0.00397	Exploits	45106.4823
Shellcode	548.26883	Reconnaissance	0.281778	Backdoor	0.00286	Generic	4660.37064
Backdoor	315.760745	Shellcode	0.005989				
Worms	231.994907						

### 5.4.10. Mean Duration of Attacks Per Protocol

Appendix 1: Code Box 23 shows the extraction of mean duration of attacks per protocol.

Table 5.26 shows the respective mean duration attacks per protocol.

**Table 5. 26. Mean Duration of Attacks per Protocol**

TCP protocol		UDP protocol		UNAS protocol		Other Protocols	
Attack Cat	Mean Duration of Attacks	Attack Cat	Mean Duration of Attacks	Attack Cat	Mean Duration of Attacks	Attack Cat	Mean Duration of Attacks
DoS	3.143824	Fuzzers	2.679468	Fuzzers	0.000007	Fuzzers	3.5272625
Fuzzers	2.572246	DoS	2.231245	DoS	0.000007	Reconnaissance	10.092908
Analysis	2.663778	Exploits	0.821981	Exploits	0.000007	Analysis	1.892185
Generic	2.116299	Backdoors	0.2372	Backdoors	0.000007	Backdoors	3.2779805
Exploits	1.774488	Generic	0.001043	Generic	0.000007	DoS	3.917924
Worms	1.516307	Reconnaissance	0.000375	Reconnaissance	0.000007	Exploits	4.661687
Backdoors	1.0855995	Shellcode	0.000008	Shellcode	0.000007	Generic	4.468236
Reconnaissance	0.996701	Worms	0.000007	Worms	0.000007		
Shellcode	0.7187945						

## 5.5. Data Cleaning

The primary goal of Data Cleaning as explained earlier is to build a valid dataset by detecting and eliminating mistakes that would arise from a set of duplicate data or missing values. This improves the training data quality for analytics and enables more accurate decision making.

### 5.5.1. Identification And Filtering of Missing Values.

In big data analytics, missing values is one of the main causes of inaccuracy when it comes to the subjection of various algorithms. Due to this, the missing values have to be handled correctly. The `unl.isnull().sum()` was used to identify the missing values within the dataset. It works by detecting missing values in the given dataframe. It returns a Boolean same-sized object indicating if the values are NA. In this instance the `sum` function was invoked to return the total missing values within the sets of features in the dataset `unl`. The value 0 depicts no values are missing. Out of all the 49 columns as shown in Table 5.27, only two were found to have missing values. The two columns were **`ct_flw_http_mthd`** with and **`is_ftp_login`**.

Table 5. 27. Missing values in malicious dataset.

Features	Missing Values	Features	Missing Values	Features	Missing Values
srcip	0	dwin	0	is_ftp_login	297183
sport	0	stcpb	0	ct_ftp_cmd	0
dstip	0	dtcpb	0	ct_srv_src	0
dsport	0	smeansz	0	ct_srv_dst	0
proto	0	dmeansz	0	ct_dst_ltm	0
state	0	trans_depth	0	ct_src_ltm	0
dur	0	res_bdy_len	0	ct_src_dport_ltm	0
sbytes	0	Sjit	0	ct_dst_sport_ltm	0
dbytes	0	Djit	0	ct_dst_src_ltm	0
sttl	0	Stime	0	attack_cat	0
dttl	0	Ltime	0	Label	0
sloss	0	Sintpkt	0		
dloss	0	Dintpkt	0		
service	0	tcprrt	0		
Sload	0	synack	0		
Dload	0	ackdat	0		
Spkts	0	is_sm_ips_ports	0		
Dpkts	0	ct_state_ttl	0		
swin	0	ct_flw_http_mthd	282939		

For the purpose of analyzing the columns with missing values, it was first extracted from the dataset as shown in Appendix 1: Code Box 24. The data was then stored temporarily in the **mvs** dataset. A run of shape **mvs** gave 321283 rows and 2 columns. The cells with values on both columns were 22215(6, 9%) and the one missing 299086(93.1%). A brief analysis is done for the two columns using the code in Appendix 1: Code Box 25 and the output is shown on the Table 5.28.

**Table 5. 28. Missing values statistics.**

Statistics	ct_flw_http_mthd	ls_ftp_login
Count	22215	22215
Mean	0.053342	0.002161
Standard deviation	0.225520	0.046434
Minimum value	0.000000	0.000000
25 <sup>th</sup> percentile	0.000000	0.000000
50 <sup>th</sup> percentile	0.000000	0.000000
75 <sup>th</sup> percentile	0.000000	0.000000
Maximum value	2.000000	1.000000

### 5.5.2. Filling of the Missing Values

In this process, the missing values on the two columns, were to be filled with the mean. To do so, a duplicate of the cells with the missing values was created then filled with their respective mean. The **mvs** dataset was duplicated. From there, the column **ct\_flw\_http\_mthd** was filled with the mean. After this, the same process was repeated for the second column **is\_ftp\_login**. From there, the column names were then renamed in order to distinguish them from other columns after they have been combined with the complete dataset. Finally, a confirmatory evaluation is done to find out whether there are any missing values which returns the values 0 for the detecting they have been filled successfully this process is shown in Appendix 1: Code Box 26.

### 5.5.3. Filling Missing Values with The Median

Median refers to the middle value in a certain group of data. The **mvs** dataset was duplicated then column **ct\_flw\_http\_mthd** and **is\_ftp\_login** was filled with their respective median

values this process is shown in Appendix 1: Code Box 27. A look at the sample of five in Table 5.29 the output confirms the columns are filled.

**Table 5. 29. Sample Missing values filled with Median.**

	median_ct_flw_http_mthd	median_is_ftp_login
321128	0	0
262107	0	0
458085	0	0
521486	0	0
344062	0	0

#### **5.5.4. Forward Filling of Missing Values**

In this procedure, consider a column with the values 1 to 5 and a missing value  $x$ . The data can have the structure of the set  $s = [1, 2, x, 4, 5]$ . In the procedure of forward filling, the missing value  $x$  is replaced with the value that comes before it. In the instance of the set  $s$ , the previous value is 2 thus that is what will replace  $x$  in its position. the **mvs** dataset is then duplicated. From there, the column `ct_flw_http_mthd` is renamed to `ffill_ct_flw_http_mthd` and is filled with the median of that column. Finally, a confirmatory evaluation was done to find out whether there are any missing values. If there are none, the code has been executed successfully.

#### **5.5.5. Backward Filling of Missing Values.**

Consider a set  $t$  with values 1 to 5 and a missing value  $x$ . The data will have the structure  $t = (1, 2, x, 4, 5)$ . In backward filling, the missing value is replaced with the value that comes after it. The set will have a new structure  $t = (1, 2, 4, 4, 5)$ . The same concept applies to the dataset that will be backward filled. To accomplish the process, a duplicate of the dataset that had the two columns was created and the back filling of the missing values done this process is shown in Appendix 1: Code Box 28.

#### **5.5.6. Filling Of Missing Values in The Mal Dataset with Created Values**

Initially the two columns with missing values were dropped so that they can be replaced with the newly created datasets as shown in Appendix 1: Code Box 29. A confirmation was done by calculating the difference between the numbers of columns in the original dataset to the columns in the current dataset this process is shown in Appendix 1: Code Box 30. Afterwards

the two modified columns were appended to the original dataset of malicious logs which created a dataframe that would be used for the feature selection process. In order to add the two columns whose missing data was replaced by the mean, a duplicate of the data with 47 columns is created as shown in Appendix 1: Code Box 31. After that, the new columns are added. A confirmation was done shows the two columns *mean\_ct\_flw\_http\_mthd* and *mean\_is\_fip\_login* was added as shown in Appendix 1: Code Box 32. The same processes were repeated for the forward filling, backward filling and median as shown in Appendix 1: Code Box 33.

## **5.6. Machine Learning algorithms**

The following algorithm were selected for the purpose of classification and clustering networks attacks

### **5.6.1. Feature Selection -SelectKBest Algorithm**

Feature selection is a critical step in model training, as it helps to focus on the most relevant features and can improve model performance and reduce overfitting. The SelectKBest algorithm is chosen to perform feature selection. It selects the top K features based on statistical tests, such as F-test or mutual information. By choosing the most informative features, the algorithm aims to enhance the performance of the subsequent models.

### **5.6.2. Base Learners -KNeighbors, Random Forest, GaussianNB**

Base learners are individual models that form the ensemble. They are trained on subsets of the data or with different features to capture diverse patterns. KNN is a simple and intuitive algorithm that classifies a data point based on the majority class of its k-nearest neighbors. KNN is chosen as a base learner because it captures local patterns in the feature space. It can be effective when instances of the same class are clustered together. By including KNN as a base learner, the ensemble benefits from its ability to capture local decision boundaries.

Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions. Random Forest is a popular choice for base learners in ensembles because it provides a good balance between bias and variance. It reduces overfitting and increases the model's generalization ability. Each tree in the forest is trained on a random subset of features and data, introducing diversity among the base learners.



Gaussian Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming that features are conditionally independent given the class. GaussianNB is chosen as a base learner for its simplicity and speed, especially when dealing with continuous data. It assumes that features are normally distributed within each class, making it suitable for certain types of datasets. Including GaussianNB as a base learner provides diversity in the modeling approach, particularly when the data's underlying distribution aligns with the assumptions of the algorithm.

### **5.6.3. Meta-Learner (Logistic Regression)**

The meta-learner combines the predictions of the base learners to make a final prediction. Logistic Regression is often used as a meta-learner due to its simplicity and interpretability. Logistic Regression is chosen as the meta-learner (TrainMetaLearner) because it is well-suited for binary classification problems and provides probabilities as outputs. Its simplicity makes it computationally efficient and less prone to overfitting, especially when the base learners are diverse.

### **5.6.4. K-means Clustering**

Clustering is used to group similar data points together, which can reveal underlying structures in the data. K-means clustering is applied to the normalized predictions of the meta-learner (KMeansAlgorithm). The clustering process helps identify patterns or groups of similar instances in the data. This can be valuable for tasks such as customer segmentation or anomaly detection.

### **5.6.5. Feature Scaling**

Feature scaling ensures that all features have similar scales, which is important for algorithms sensitive to the scale of input features, such as K-means clustering. Feature scaling (e.g., Min-Max scaling) is applied to the predictions of the meta-learner (FeatureScaling). This step ensures that the features used in K-means clustering have consistent scales, improving the effectiveness of the clustering algorithm.

In summary, the chosen algorithms are designed to work together in a stacking ensemble framework. Feature selection, diverse base learners, a logistic regression meta-learner, and K-means clustering are employed to enhance model performance, interpretability, and uncover potential patterns in the data.

## 5.7. Ensemble Technique Approach

Ensemble is a machine learning technique employs several learners to address a single classification or regression issue. Ensemble techniques build a group of models (learners) and combine them as opposed to traditional machine learning approaches, which aim to build a model from training data. After two groundbreaking studies in the 1990s, ensemble techniques became a significant area of research. According to (Hansen & Salamon,1990), when a group of classifiers is used in conjunction to make a prediction, the results are frequently more accurate than those of a single classifier. (Schapire,1990) demonstrated weak models (learners) can be boosted by strong models(learns). three common ensemble techniques exist include Bagging, Boosting and stacking.

Bagging method combines the averaging method for regression and the voting method for classification. This methodology employs the voting method for classification and the averaging method for regression. The ensemble approach is used to aggregate data from base learners and then vote on the labels during the voting phase. The one with the most votes is chosen as the system's Prediction(Zhou, 2012). Random forest algorithm combining random decision tree trains by drawing random subset of the training set is one of the algorithm used this method(Altman & Krzywinski, 2017).

Boosting improves weak classifiers by increasing their execution. It implements a successive learning technique by enabling variations to work on a given batch of data which is repeated until the desired outcome is obtained(Syarif et al., 2012). By leveraging the misclassified training cases that earlier models, boosting progressively creates an ensemble model. The adaboost algorithm is one of the illustration of this approach(Mayr et al., 2014).

Stacking, combines the predictions of other models using an algorithm. It works and learns from data by utilizing different algorithms at the base layer and one at the Meta layer. The base layer algorithm' outputs are combined to provide an input for the Meta layer algorithm. The Meta layer algorithm receives input, and then generates an output that will be the final model.

### 5.7.1. Stacking Ensemble

This study adopted Stacking, since its powerful ensemble learning technique, combines predictions from diverse machine learning models to enhance predictive accuracy. The method leverages the strengths of different models, reducing overfitting and increasing robustness to outliers. Stacking's flexibility allows for the use of varied base models, adapting to dataset characteristics. It excels in handling complex relationships and heterogeneous data. The final stacked model may also offer improved interpretability. However, successful implementation requires careful tuning, validation, and the selection of complementary base models to avoid overfitting. Figure 5.17 shows the adopted stack, it contains the select Kbest Feature Selection Algorithm for its ability to identify the appropriate features, three classifiers base-learners KNeighbors for its ability classifies network traffic data based on the similarity, RandomForest for its ability to detect anomalies and GaussianNB for its ability to classify normal and abnormal distribution in the network traffic and one clustering classifier meta-learner KMeans.

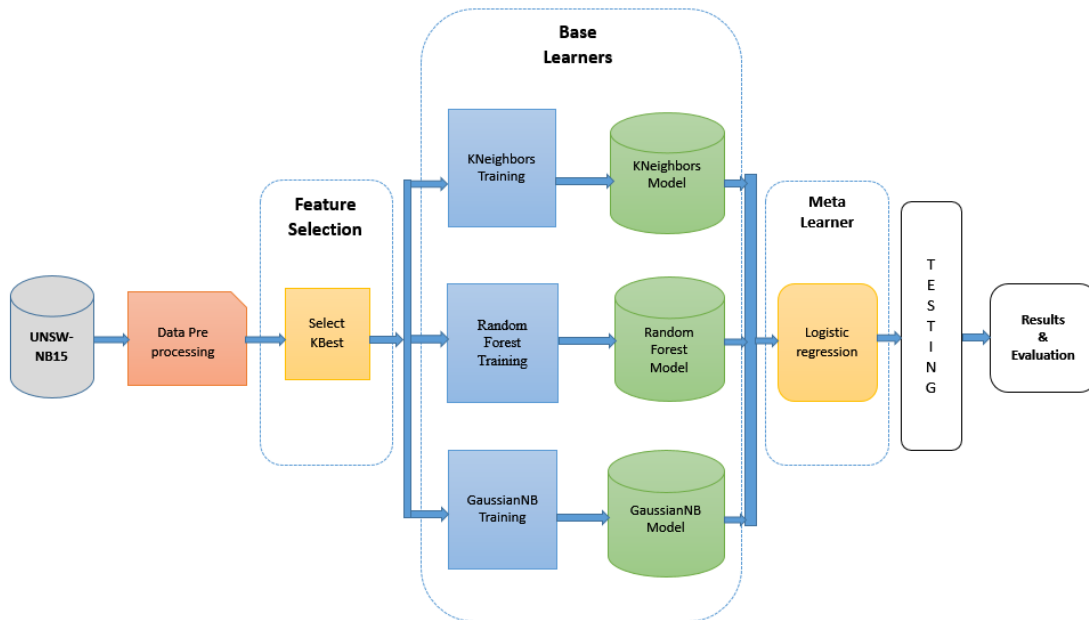


Figure 5. 17. Stacking Ensemble Approach

### 5.7.2. Feature Selection

To use machine learning algorithms efficiently, it is becoming increasingly important to perform feature selection (V. Kumar & Minz, 2014). Features election, also referred to as attribute selection or variable selection, is a process of selecting more relevant attributes, and

removing irrelevant or less relevant attributes or noisy data or features that do not add additional value to a machine learning algorithm. Using only relevant features for machine learning algorithms allows for faster processing and more accurate predictability (Yang & Zhu, 2011). A lot of work has been done on feature selection using traditional techniques like Information Gain, the Gini Index, uncertainty, and correlation coefficients (Forman, 2003). This study used the SelectKBest algorithm for feature selection. The SelectKBest method chooses the top  $k$  best features from the input dataset based on a scoring measure. The feature selection equation is represented as:

$$fs(x) = x_{selected} \quad \dots\dots eq... (5.1)$$

where  $fs$  is the feature selection,  $x$  is the input feature matrix and  $X_{selected}$  comprises the  $k$  best features that were chosen. The parameter  $k$  for SelectKBest defines the number of characteristics to be chosen.

### 5.7.3. Base Learners

The KNearest Neighbors (KNN) method classifies or predicts the target variable using the feature space's  $k$  nearest neighbors. The KNearest Neighbors training equation is represented as follows

$$b1(x_{train}, y_{train}) = model_{b1} \quad \dots\dots eq... (5.2)$$

where  $b1$  is the first base learner KNN,  $X_{train}$  is the training feature matrix and  $y_{train}$  is the associated labels. The argument for KNearest Neighbors is  $k$ , which specifies the number of neighbors to take into account.

The Random Forest (RF) method constructs a decision tree ensemble and generates predictions based on the majority vote or average of the individual trees. The Random Forest training equation can be represented as

$$b2(x_{train}, y_{train}) = model_{b2} \quad \dots\dots eq... (5.3)$$

where  $b1$  is the second base learner RF,  $X_{train}$  is the training feature matrix and  $y_{train}$  is the associated labels. Random Forest settings include the number of trees, the maximum depth, and other tree-specific characteristics.

To classify or forecast the target variable, the Gaussian Naive Bayes (GNB) method assumes that features are independent and uses a Gaussian distribution. The Gaussian NB training equation is represented as:

$$b3(x_{train}, y_{train}) = model_{b3} \quad \dots\dots eq \dots (5.4)$$

where  $b3$  is the third base learner GNB,  $X_{train}$  is the training feature matrix and  $y_{train}$  is the associated labels. There are no settings to adjust for Gaussian NB.

#### 5.7.4. Meta Learner

Logistic Regression ( $LR$ ) employs the logistic function to represent the association between the chosen characteristics and the intended variable. A representation of the Logistic Regression training equation is:

$$l(x_{meta}, y_{meta}) = model_l \quad \dots\dots eq \dots (5.5)$$

where  $l$  is the  $LR$ , where  $X_{meta}$  is the meta-features (predictions from  $b1$ ,  $b2$ , and  $b3$ ) and  $y_{meta}$  is the associated labels. The regularization term, solver technique, and convergence criteria are some of the variables in logistic regression.

Stacking Ensemble's prediction ( $e$ ) combines the predictions of each model ( $B1$ ,  $B2$ ,  $B3$ , and  $L$ ) after they have all been trained to provide the final prediction. The test feature matrix  $X_{test}$  is used in the equation for the stacking ensemble prediction, which is written as

$$e(x_{test}) = l(b1(fs(x_{test})), b2(fs(x_{test})), b3(fs(x_{test}))), \quad \dots\dots eq \dots (5.6)$$

where  $X_{test}$  is the test feature matrix. To apply the equations and parameters for training and prediction, it would typically follow the following steps:

- |   |
|---|
| <p>BEGIN:</p> <ol style="list-style-type: none"> <li>1. Apply feature selection (<math>fs</math>) on the training dataset (<math>x_{train}</math>) to select the top k features.</li> <li>2. Train the base learners (<math>b1</math>, <math>b2</math>, <math>b3</math>) on the selected features from step 1 using their respective algorithms and parameters.</li> <li>3. Generate predictions from each base learner on the meta-features (<math>x_{meta}</math>) of the training dataset.</li> <li>4. Train the meta learner (<math>m</math>) using the meta-features and the corresponding labels (<math>y_{meta}</math>).</li> <li>5. For prediction, apply feature selection (<math>fs</math>) on the test dataset (<math>x_{test}</math>) to select the same k features as in step 1.</li> <li>6. Generate predictions from each base learner on the selected features from step 5.</li> <li>7. Combine the base learners' predictions as meta-features and input them into the trained meta learner (<math>m</math>) to obtain the final prediction.</li> </ol> <p>END</p> |
|---|

Figure 5. 18. Meta learner prediction flows

### 5.7.5. Clustering Algorithm Based on KMeans

The primary goal of the K-means algorithm is to minimize the total sum of squared distances between the data points and the centroids of their respective assigned clusters. The representation is as follows.

$$J(C, \mu) = \sum ||x_i - \mu_{c_j}||^2 \quad \dots\dots\dots eq\dots (5.7)$$

The objective function, denoted as  $J(C, \mu)$ , is defined as the sum of the squared Euclidean distances between each data point  $x_i$  and its corresponding cluster center  $\mu_{c_j}$ , summed over all data points. The objective function that requires minimization is denoted as  $J(C, \mu)$ . The set  $C$ , denoted as  $C = \{c_1, c_2, \dots, c_n\}$ , represents the cluster assignments for each individual data point in the dataset  $X$ . On the other hand, the set  $\mu$ , denoted as  $\mu = \{\mu_1, \mu_2, \mu_3\}$ , represents the cluster centroids. The objective function computes the squared Euclidean distance between every data point  $x_i$  and its corresponding cluster centroid  $\mu_{c_j}$ . The objective is to minimize the distance between each data point and its corresponding centroid, thereby indicating a closer proximity.

The K-means algorithm is subject to two primary constraints: (1) It is imperative that every individual data point is allocated to a single cluster. (2) The centroid of each cluster is calculated as the average of the data points assigned to that cluster, denoted by the equations  $x$  and  $y$ , respectively. The summation of  $c_j$  equals  $k$  is equal to  $1$ , for all  $j$  ranging from  $1$  to  $N$ , and  $k$  ranging from  $1$  to  $3$ .

$$\sum (c_j = k) = 1, \text{ for all } j = 1, 2, \dots, N, \text{ and } k = 1, 2, 3. \quad \dots\dots\dots eq\dots (5.8)$$

Constraint 1; guarantees that each data point is exclusively assigned to a single cluster. The expression  $(c_j = k)$  yields a value of  $1$  when the data point  $x_j$  is assigned to cluster  $k$ , and  $0$  otherwise. The total of this term across all clusters should be equivalent to  $1$ , signifying that each data point possesses a distinct cluster assignment. The formula for calculating the mean of a set of values  $x_i$ , where each value is associated with a category  $c_j$ , is given by

$$\mu_k = (1 / |\{j: c_j = k\}|) \sum (x_i, c_j = k), \text{ for all } k = 1, 2, 3. \quad \dots\dots\dots eq\dots (5.9)$$

Constraint 2; involves the updating of cluster centroids in accordance with the assigned data points. The centroid  $\mu_k$  for each cluster  $k$  is determined by calculating the average of the data points  $x_i$  that are assigned to that specific cluster. The expression  $(x_i, c_j = k)$  denotes the

summation over the data points  $x_i$ , subject to the condition that their cluster assignment  $c_j$  is equal to  $k$ . Additionally,  $| \{j: c_j = k\} |$  represents the count of data points that have been assigned to cluster  $k$ .

The K-means algorithm iteratively optimizes the cluster assignments and cluster centroids by minimizing the objective function  $J(C, \mu)$  while ensuring that the constraints are satisfied, until convergence is achieved. The algorithm employs a two-step process, wherein it iteratively updates the assignments by considering the closest centroid and recalculates the centroids based on the assigned data points. The objective is to identify the configuration that minimizes the total sum of squared distances. Figure 5.19 is the K-means clustering algorithm based on  $k=3$

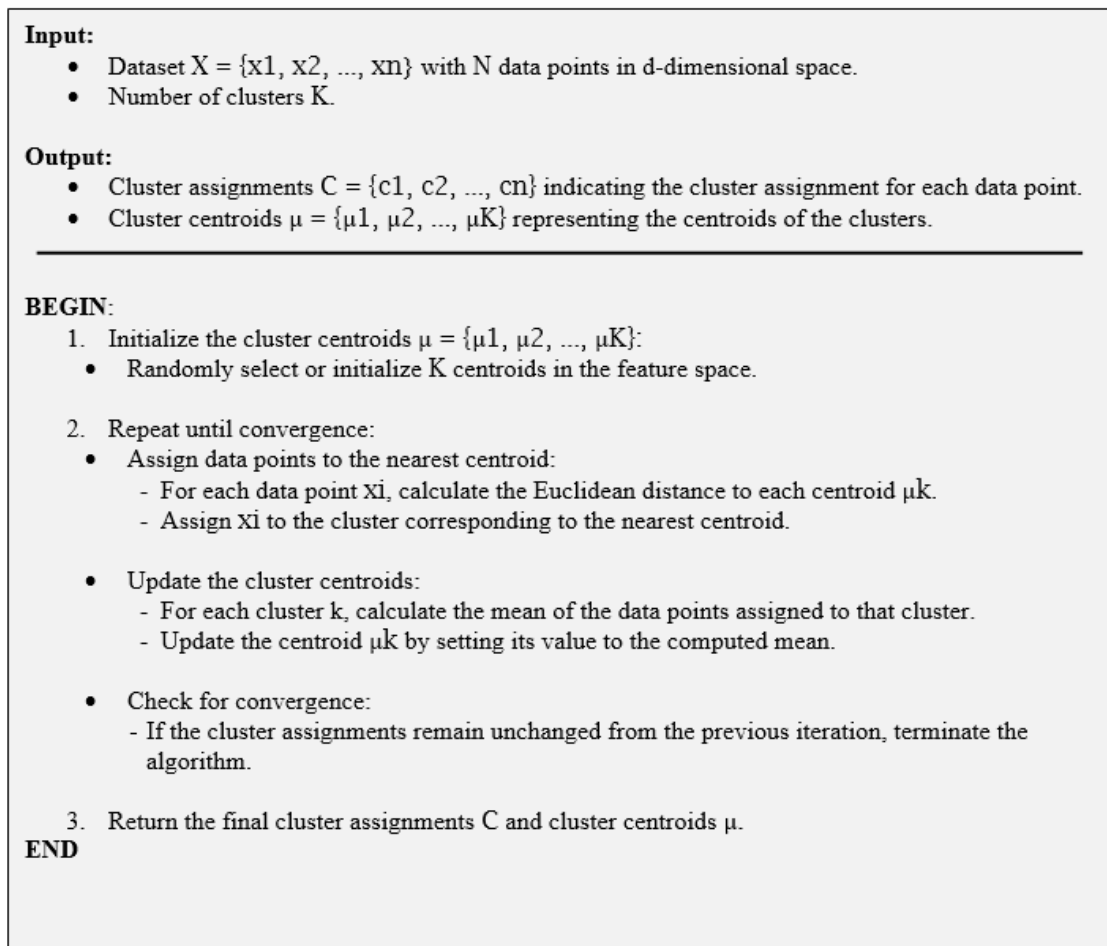


Figure 5. 19. KMeans Clustering flow.

### 5.7.6. Performance Metric

Table 30 below shows output of the accuracy, precision, recall, and F1 score of the SelectKBest feature selection algorithm. the performance metrics for each base learner (K-Nearest Neighbors, Random Forest, and Gaussian Naive Bayes) and the meta learner (Logistic Regression) using the predictions from the base learners in the stacking ensemble.

**Table 5. 30. Performance metrics scores**

Classification Technique	Accuracy	Precision	Recall	F1 Score
Select KBest	0.93466	0.94393	0.93466	0.95387
KNN	0.85767	0.86776	0.85765	0.85734
Random forest	0.92335	0.91254	0.96667	0.95667
Gaussian Naive Bayes	0.88899	0.82785	0.80956	0.79676
<b>*Proposed ensemble</b>	0.96667	0.96967	0.96667	0.96998

## 5.8. Feature Engineering and Feature Selection

The feature engineering process refers to the procedures involved in transforming raw data into useful data for machine learning algorithm(s) to use. Outcome and predictor variables are present in situations such as predictive machine learning algorithms. Through feature engineering, the predictor variables are converted into a format that is suitable with various machine learning methods.

### 5.8.1. Training and Testing Splitting

The data was split into train and test the reason being to prepare it be used as a learning point for the machine learning algorithms and to have a portion of the data that will be used to test the accuracy of the machine learning algorithms . The X-train was used to store the columns that will be used to train the machine learning algorithms. X-test was used to evaluate the accuracy of the output obtained after subjecting the data to the machine learning algorithms. Y-test was used to store the output of the machine learning algorithms when they have been tested with a real-life dataset and y-train was used as the columns that provides the input for the machine learning algorithms to test the final accuracy. In this study, the training test of the



data took 70% while the testing set took 30%. The *random\_state* was set to 101 so that for each instance the code is run, everything is held constant. After that process, the values that describe the splitting structure are obtained.

### 5.8.2. The Select KBest Feature Selection Algorithm

The Select KBest process was used during the feature selection process for the features that are necessary for the machine learning algorithm. It is imported from the *sklearn.feature\_selection* library, it also picks the methods that can be used during the process.

Before the process is carried out, data had to be encoded using various label encoding feature engineering techniques and be split into train and test sets. This can be applied in instances when the data being used has strings in the columns. Appendix 1: Code Box 34 was used to identify columns with object data type and Table 5.31 shows the columns with the object data type. The object data type forms one of the greatest foundations for the data encoding process.

**Table 5. 31. Columns with the object data type**

	srcip	sport	dstip	dsport	proto	state	service	ct_ftp_cmd	attack_cat
19	175.45.176.3	21223	149.171.126.18	32780	udp	INT	-	0	Exploits
20	175.45.176.2	23357	149.171.126.16	80	tcp	FIN	http	0	Exploits
21	175.45.176.0	13284	149.171.126.16	80	tcp	FIN	http	0	Reconnaissance
38	175.45.176.2	13792	149.171.126.16	5555	tcp	FIN	-	0	Exploits
39	175.45.176.2	26939	149.171.126.10	80	tcp	FIN	http	0	Exploits
56	175.45.176.0	39500	149.171.126.15	80	tcp	FIN	http	0	DoS
57	175.45.176.0	29309	149.171.126.14	3000	tcp	FIN	-	0	Generic
76	175.45.176.0	61089	149.171.126.18	80	tcp	FIN	http	0	Exploits
77	175.45.176.0	23910	149.171.126.15	80	tcp	FIN	http	0	DoS
78	175.45.176.3	44762	149.171.126.12	80	tcp	FIN	http	0	Exploits

**Table 5.32. Confirmation of columns with the object data type.**

srcip	object
sport	object
dstip	object
dsport	object
proto	object
state	object
service	object
ct_ftp_cmd	object
attack_cat	object

**dtype: object**

Table 5.32 shows a confirmation that the extraction process of the columns with the object data type had been accurately identified.

For the SelectKBest algorithm to work, it must have an output variable that can be used as a reference point to compare the strength of each column one to the other. The commonly used variable to filter out the column is *Y*. A further confirmation was done to ensure that the number of rows is in the original nature during the entire process which is shown Appendix 1: Code Box 35. A similar process was repeated for the input columns which were to be evaluated one against the other. The columns were stored in the variable *X*. The top 10 columns in *X* were the ones to be used in the machine learning algorithm. A further test was done to confirm the number of rows in *X* is equal to that of *Y* to avoid any form of bias as shown Appendix 1: Code Box 36. For each of the objects identified from the above process, were encoded using the *LabelEncoder()* function which is stored in the variable *lbl\_enc* as shown in Appendix 1: Code Box 37 .

Moving forward, the Chi2 squared test as shown Appendix 1: Code Box 38 was used because it is flexibility to handle data with different types of columns. Also, you can customize the code to enter the exact number of columns that may be required in the model. After that process, the following columns in Table 5.33 were obtained and they were the ones to be subjected to the machine learning algorithms (this could also be achieved by performing correlation matrix of all the features).

**Table 5. 33. Selected features for machine learning algorithm.**

	ct_flw_htt p_mthd	is_ftp_l ogin	ct_ftp_ cmd	ct_srv _src	ct_srv _dst	ct_dst_ ltm	ct_sr c_ ltm	ct_src_dpor t_ltm	ct_dst_spor t_ltm	ct_dst_src _ltm
19	0	0	0	1	1	1	1	1	1	1
20	1	0	0	3	2	2	1	1	1	1
21	1	0	0	5	2	2	1	1	1	1
38	0	0	0	1	1	1	1	1	1	1
39	1	0	0	3	1	1	1	1	1	1

### 5.8.3. Importation of the Machine Learning Algorithms.

The following libraries were imported see Appendix 1: Code Box 39 and used together for the implementation of the machine learning algorithms that were used in this study.

The *Itertools* library facilitates fast and memory-efficient iteration through datasets for various functions. *Numpy* is employed for array manipulation and mathematical operations. *Seaborn* and *Matplotlib* are utilized for creating visualizations to aid comprehension of data outputs. The *sklearn.datasets* module provides datasets for experimental use before deploying algorithms on the final dataset. The *sklearn.linear\_model* library stores predefined functions and parameters, useful for algorithms like logistic regression. The *sklearn.neighbors* module imports the K-Nearest Neighbors algorithm, while *sklearn.naive Bayes* offers functions for implementing the Gaussian Naïve Bayes algorithm. The *sklearn.ensemble* algorithm includes the Random Forest Classifier, which enhances prediction accuracy through decision tree subsets. The stacking classifier serves as an ensemble algorithm using multiple classifiers' predictions as new features. Visualizations of machine learning outputs are created using the *plot\_learning\_curves* and *plot\_decision\_regions* libraries from the *mlexend.plotting* module.

### 5.8.4. Preparation for the Machine Learning Algorithms.

In the code Appendix 1: Code Box 40 was used to store all columns that was to be used for the training and testing of the machine learning algorithms. Though that was the case, the column that was guiding in the process of making the predictions (*attack\_cat*) was dropped and retained in the variable *y*.

The *clf<sub>1</sub>* variable was used to store the function that will assist in the implementation of the K-Nearest Neighbors algorithm. The *clf<sub>2</sub>* variable was used to store the Random Forest Classifier that is composed of decision trees as its main functionality. *clf<sub>3</sub>* was used as a support platform for the Gaussian Naïve Bayes machine learning algorithm. Lastly, *sclf* was used to store the Stacking Classifier algorithm that combines the three algorithms that was used in the study.

Appendix 1: Code Box 41, the data was then split into three clusters for each of the classification in the surface being attacked. The random state is also set to 0 so that for every time the code is repeated, the same output is obtained. The *processed\_plot* variable is used to store the outputs for the K-Means machine learning algorithm. From the process as seen Appendix 1: Code Box 42 three clusters were obtained from the K-Means machine learning algorithm. The three outputs were then mapped to the surface they relate to in terms of attacks as shown in Appendix 1: Code Box 43, a sample is shown in Table 5.34.

**Table 5. 34. Mapping to attacks surfaces.**

results	
0	Hosts
1	Hosts
22213	Users
22214	Hosts
22215 rows × 1 columns	

The mapping was further done to the data subjected to the SelectKBest algorithm. Alongside it is the KMeans cluster that was assigned to it. Appendix 1: Code Box 44 Table 5.35 shows the output.

**Table 5. 35. Sample of Mapping the outputs to the SelectKBest algorithm data**

	0	1	2	3	4	5	6	7	8	KMeans_cluster	results	ys
12917	0	0	6	6	2	2	2	2	2	1	1	Users
12778	0	0	1	4	1	0	0	0	0	0	0	Hosts
9059	0	0	6	6	1	5	1	1	1	1	1	Users
19977	0	0	2	2	2	2	2	2	2	0	0	Hosts
13128	0	0	7	2	0	0	0	0	0	0	0	Hosts
18121	0	0	0	5	1	0	0	0	0	0	0	Hosts
21411	0	0	6	4	0	0	0	0	0	0	0	Hosts
10297	0	0	17	17	10	10	10	2	10	2	2	Media
18457	0	0	11	11	2	2	2	2	2	1	1	Users
3297	0	0	0	0	0	0	0	0	0	0	0	Hos

Appendix 1: Code Box 45 Table 5.36 shows shows the respective code and sample output of the attacks to their attack surface.

**Table 5. 36. sample output for the attack surfaces.**

	srcip	sport	dstip	dsport	proto	state	outputs
170723	175.45.176.0	1334	149.171.126.11	445	tcp	FIN	Hosts
75296	175.45.176.0	42860	149.171.126.11	80	tcp	FIN	Hosts
103361	175.45.176.1	47439	149.171.126.18	53	udp	INT	Users
168423	175.45.176.1	47439	149.171.126.18	53	udp	INT	Users
154214	175.45.176.1	1043	149.171.126.18	53	udp	INT	Hosts
140037	175.45.176.3	39779	149.171.126.11	1723	tcp	FIN	Hosts
120785	175.45.176.1	47439	149.171.126.18	53	udp	INT	Media
163492	175.45.176.1	47439	149.171.126.18	53	udp	INT	Hosts
84095	175.45.176.1	42928	149.171.126.12	111	scps	INT	Users
130654	175.45.176.1	47439	149.171.126.18	53	udp	INT	Users

### 5.8.5. Analysis of Attacks to Their Surfaces.

In order to map and visualize the attacks to the surfaces Appendix 1: Code Box 46 was used. The output was placed in a dataset called *distribution*. The *groupby* function is used to compare one column value to the other column(s) analytically by methods such as sum and mean. In this instance, the *groupby* function counts the number of attacks per surface and sorts them in descending order.

### 5.8.6. Visualization of Attacks to Their Surface.

The process starts by classifying the three attack surfaces to different datasets. The host, media and users' surfaces are placed in *host*, *media* and *users'* datasets respectively as shown in Appendix 1: Code Box 47 the filtered datasets for the attack surface counts are then stored into individual arrays *h*, *m* and *u* as shown in Appendix 1: Code Box 48, Appendix 1: Code Box 49 shows the attacks in *h*, *m* and *u*, Appendix 1: Code Box 50 was used to make visualization for attack surface distribution per attack. Figure 5.20 and Table 5.37 shows the attack distribution for each surface of attack.

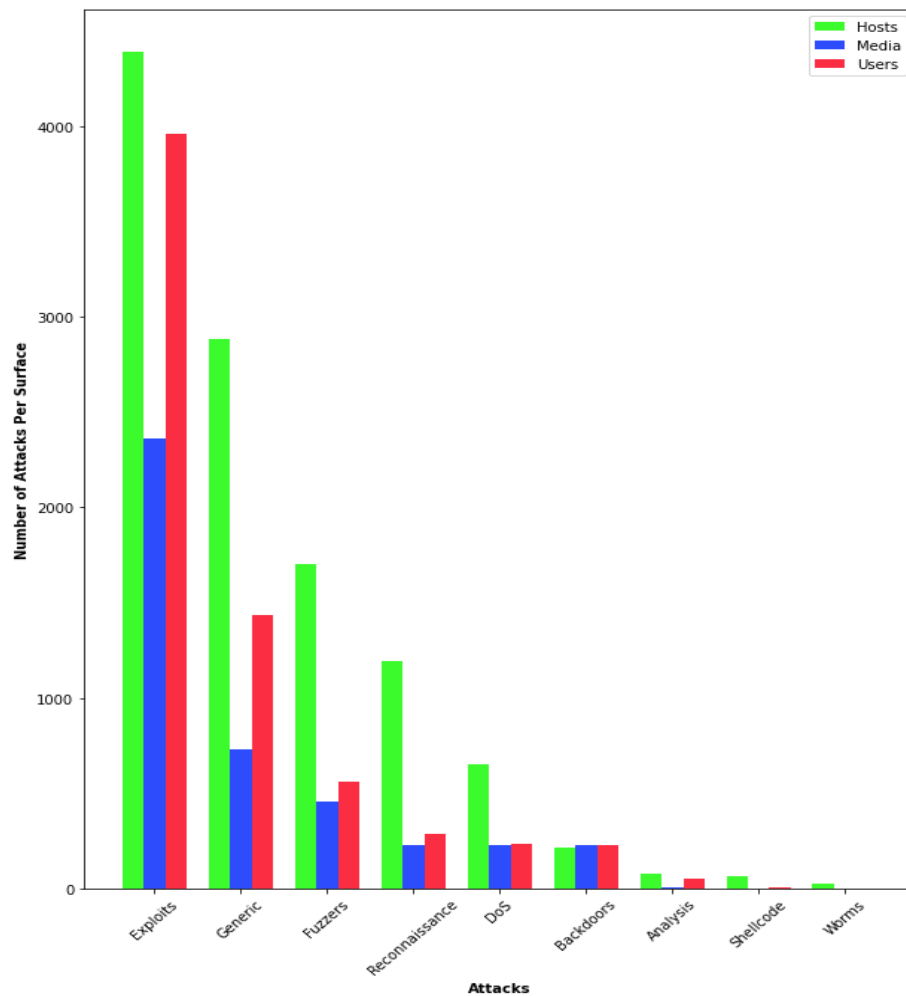


Figure 5. 20. Attack Distribution for Each Surface of Attack.

**Table 5. 37. Attack Distribution for Each Surface of Attack.**

Attacks	Host Attacks	Media Attacks	User Attacks
Exploits	4393(40.98 %)	2364(22.05%)	3963(36.97%)
Generic	2884(57.1%)	730(14.45%)	1437(28.45%)
Fuzzers	1705(62.66%)	454(16.69%)	562(20.65%)
Reconnaissance	1195(69.96%)	227(13.29%)	286(16.74%)
DoS	654(58.65%)	226(20.27%)	235(21.08%)
Backdoors	215(32.14%)	226(33.78%)	228(34.08%)
Analysis	80(59.7%)	3(2.24%)	51(38.06%)
Shellcode	65(89.04%)	1(1.37%)	7(9.59%)
Worms	23(95.83%)	0(0%)	1(4.17%)

### 5.9. Identification and Retrieval of Stride Based Attacks from CAPEC Repository.

To achieve this process the CAPEC repository dataset was uploaded in the machine learning platform, they were three datasets namely *capec1*, *capec2*, *capec3* the process was achieved using code in Appendix 1: Code Box 51. The three datasets had the following respective shapes *Capec1.shape* (546, 20), *capec2.shape* (546, 20), *Capec3.shape* (601, 20) giving it a cumulative of (1693, 20). The column names were renamed to a new format that can be easily managed during the wrangling process. The renaming process starts by subjecting the new column names as seen in Appendix 1: Code Box 52 into a list which were then subjected to the datasets as seen in Appendix 1: Code Box 53 and giving out the dataset features on Table 5.38.

**Table 5. 38. CAPEC Dataset Features.**

ID	Name	Abstraction	Status	Description	AlternateTerms	LikelihoodOfAttack	TypicalSeverity	RelatedAttackPatterns
0	0	Accessing Functionality Not Properly Constrain...	Standard	Draft	In applications, particularly web applications...	NaN	High	High ::NATURE:ChildOf:CAPEC : ID:122::NATURE:CanPrece...

#### 5.9.1. Filtering Of UNSW-NB\_15 Attacks from CAPEC.

The attacks in UNSW-NB\_15 dataset was filtered out of the CAPEC repository datasets. To do so, the filter used a string-based approach to extract the strings that match the data that was entered into the system, this approach was adopted from (Xiaohong Yuan et al., 2014b).

The generic attacks from CAPEC were extracted using the *Generic* string name as shown in as seen in Appendix 1: Code Box 54 and were stored in the *gen* dataset. From this filter one attack was obtained CAPEC ID 468: Generic Cross-Browser Cross-Domain Theft. Table 5.39 shows the output generated.

**Table 5. 39. Generic attacks from CAPEC.**

ID	Name
468	Generic Cross-Browser Cross-Domain Theft

The exploit attacks from CAPEC were extracted using the *Exploit* string name which yielded fourteen attacks as shown in Table 5.39. Appendix 1: Code Box 55 was used for this purpose.

**Table 5. 40. Exploits attacks from CAPEC.**

ID	Name
21	Exploitation of Trusted Identifiers
22	Exploiting Trust in Client
43	Exploiting Multiple Input Interpretation Layers
50	Password Recovery Exploitation
121	Exploit Non-Production Interfaces
149	Explore for Predictable Temporary File Names
160	Exploit Script-Based APIs
180	Exploiting Incorrectly Configured Access Control Security Levels
217	Exploiting Incorrectly Configured SSL
663	Exploitation of Transient Instruction Execution
665	Exploitation of Thunderbolt Protection Flaws
679	Exploitation of Improperly Configured or Implemented Memory Protections
680	Exploitation of Improperly Controlled Registers
681	Exploitation of Improperly Controlled Hardware Security Identifiers

The fuzzer attacks from CAPEC were extracted using various string names and were stored in various datasets. The string "Fuzzers" and "Fuzzer" yielded no results in this scenario it is recommended that a finer search is done by reducing the character length of a string, so "fuz" was used and yielded four attacks as seen in as seen in Appendix 1: Code Box 56. Table 5.41 shows the output.



**Table 5. 41. Fuzzers' attacks from CAPEC.**

ID	Name
28	Fuzzing
215	Fuzzing for application mapping
261	Fuzzing for garnering other adjacent user/sensitive data
122	Fuzzing for garnering J2EE/.NET

For denial-of-service attacks (DOS) two strings were used to maximize the search that is “Denial” and “DoS” yielded two attacks Capec ID 341 and ID 582 as seen in as seen in Appendix 1: Code Box 57 and Table 5.42.

**Table 5. 42. DOS attacks from CAPEC for dos and dos2 respectively.**

ID	Name
582	DEPRECATED: Violating Implicit Assumptions Reg...

ID	Name
469	HTTP DoS
582	DEPRECATED: Violating Implicit Assumptions Reg...

The reconnaissance attacks from CAPEC were extracted using the *Reconnaissance* string name and were stored in the *rec* dataset which yielded one attack as seen in Appendix 1: Code Box 58 and Table 5.43.

**Table 5. 43. Reconnaissance attacks from CAPEC.**

ID	Name
529	Malware-Directed Internal Reconnaissance

The analysis attempts from CAPEC were extracted using the *Analysis string* name and were stored in the *an* dataset two attacks were yielded as seen in Appendix 1: Code Box 59 and Table 5.44.

**Table 5. 44. Analysis attacks from CAPEC**

ID	Name
192	Protocol Analysis
621	Analysis of Packet Timing and Sizes

The shell attacks from CAPEC were extracted using the *Shell* string name and were stored in the *sc* dataset as seen in Appendix 1: Code Box 60 and Table 5.45.

**Table 5. 45. Shell attacks from CAPEC.**

ID	Name
650	Upload a Web Shell to a Web Server

A filter of both backdoors attacks and worm’s attacks yielded not a single attack within the CAPEC repository as seen in Appendix 1: Code Box 61.

### 5.9.2. Merging of Attacks In UNSW-NB\_15 Found In CAPEC.

The UNSW-NB\_15 attacks that were found in the CAPEC repository were merged together to form a single dataset named *filt*, as seen in Appendix 1: Code Box 61 .Table 5.46 shows the output, the total number of attacks retrieved were twenty-five.

**Table 5. 46. UNSW-NB\_15 found in CAPEC.**

ID	Attack
21	Exploitation of Trusted Identifiers
22	Exploiting Trust in Client
43	Exploiting Multiple Input Interpretation Layers
50	Password Recovery Exploitation
121	Exploit Non-Production Interfaces
149	Explore for Predictable Temporary File Names
160	Exploit Script-Based APIs
180	Exploiting Incorrectly Configured Access Control Security Levels
217	Exploiting Incorrectly Configured SSL
663	Exploitation of Transient Instruction Execution
665	Exploitation of Thunderbolt Protection Flaws
679	Exploitation of Improperly Configured or Implemented Memory Protections
680	Exploitation of Improperly Controlled Registers
681	Exploitation of Improperly Controlled Hardware Security Identifiers
468	Generic Cross-Browser Cross-Domain Theft
122	Fuzzing for garnering J2EE/.NET
28	Fuzzing
215	Fuzzing for application mapping
261	Fuzzing for garnering other adjacent user/sensitive data
469	HTTP DoS
582	Violating Implicit Assumptions Reg...
529	Malware-Directed Internal Reconnaissance
192	Protocol Analysis
621	Analysis of Packet Timing and Sizes
650	Upload a Web Shell to a Web Server

### 5.9.3. Mapping UNSW-NB\_15 Attacks Identified CAPEC To STRIDE Threat Model

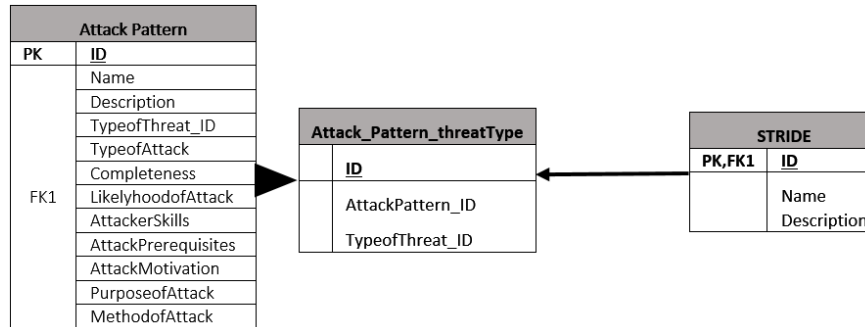
For the researcher to be able to achieve the mapping process the first stem was to look at the description of the twenty-five attacks found in CAPEC. Table 5.47 shows those attacks

**Table 5. 47. UNSW-NB\_15 Attacks Category vis a vis CAPEC Attack Patterns**

UNSWb 15 Attack CAT	CAPEC ID	CAPEC Attacks patterns
Exploits (14 attacks)	121	Exploit Non-Production Interfaces
	149	Explore for Predictable Temporary File Names
	160	Exploit Script-Based APIs
	180	Exploiting Incorrectly Configured Access Contr...
	21	Exploitation of Trusted Identifiers
	217	Exploiting Incorrectly Configured SSL
	22	Exploiting Trust in Client
	43	Exploiting Multiple Input Interpretation Layers
	50	Password Recovery Exploitation
	663	Exploitation of Transient Instruction Execution
	665	Exploitation of Thunderbolt Protection Flaws
	670	Exploitation of Improperly Configured or Imple...
	680	Exploitation of Improperly Controlled Registers
	681	Exploitation of Improperly Controlled Hardware...
Generic (1 attack)	468	Generic Cross-Browser Cross-Domain Theft
Fuzzers (4 attack)	122	Fuzzing for garnering J2EE/.NET-ba...
	215	Fuzzing for application mapping
	261	Fuzzing for garnering other adjacent user/sens...
	28	Fuzzing
DOS (2 attacks)	469	HTTP DoS
	582	Violating Implicit Assumptions Reg...
reconnaissance (1)	529	Malware-Directed Internal Reconnaissance
analysis(2attacks)	192	Protocol Analysis
	621	Analysis of Packet Timing and Sizes
Shellcode (1 attack)	650	Upload a Web Shell to a Web Server

### 5.10. Mapping Process of CAPEC Attacks to STRIDE

To aid in search process, a database schema was developed containing CAPEC attack patterns with their respective mapping to STRIDE categories Figure 5.21 illustrates the schema



**Figure 5. 21. CAPEC/STRIDE database Schema.**

The mapping table makes it simple to search the CAPEC attack patterns associated with a specific STRIDE category. With this in mind, an algorithm was developed with the following presumptions.

- i. In terms of the most valuable patterns in the CAPEC collection are extensive patterns, stud and hook respectively.
- ii. As a general rule, it is preferable to choose an attack pattern that has a high chance of being successful rather than one that has a low chance.
- iii. A severe attack pattern is more useful or noteworthy.
- iv. An attack pattern requiring little or no attacker expertise is more valuable or significant than one requiring a lot of attacker skill.
- v. Depending on their effect values, some STRIDE types prioritize components with high CIA impact more than others. An attack method with a high Confidentiality Impact but low Availability Impact is better for Spoofing threats than Denial of Service threats.
- vi. A more significant benefit can be expected from an attack pattern with a more compelling attack motivation.

As shown by Table 5.48 out of the 25 attacks that were detected only two attacks was not mapped to the threat model that is CAPEC ID 122 *Fuzzing for garnering J2EE/.NET-based stack traces* and ID 582 *violating implicit assumptions regarding XML Content* because it was deprecated in the repository and is now considered privilege abuse

**Table 5. 48. Mapping of STRIDE to CAPEC.**

UNSBw 15 Attack CAT	CAPEC ID	CAPEC Attacks	S	T	R	I	D	E
Exploits	21	Exploitation of Trusted Identifiers						
	22	Exploiting Trust in Client						
	43	Exploiting Multiple Input Interpretation Layers						
	50	Password Recovery Exploitation						
	121	Exploit Non-Production Interfaces						
	149	Explore for Predictable Temporary File Names						
	160	Exploit Script-Based APIs						
		Exploiting Incorrectly Configured Access Control Security						
	180	Levels						
	217	Exploiting Incorrectly Configured SSL						
	663	Exploitation of Transient Instruction Execution						
	665	Exploitation of Thunderbolt Protection Flaws						
	679	Exploitation of Improperly Configured or Implemented Memory Protections						
	680	Exploitation of Improperly Controlled Registers						
	681	Exploitation of Improperly Controlled Hardware Security Identifiers						
Generic	468	Generic Cross-Browser Cross-Domain Theft						
Fuzzers	<b>122</b>	<b>Fuzzing for garnering J2EE/.NET</b>						
	28	Fuzzing						
	215	Fuzzing for application mapping						
	261	Fuzzing for garnering other adjacent user/sensitive data						
DOS	469	HTTP DoS						
	<b>582</b>	<b>Violating Implicit Assumptions Reg...</b>						
reconnaissance	529	Malware-Directed Internal Reconnaissance						
analysis	192	Protocol Analysis						
	621	Analysis of Packet Timing and Sizes						
Shellcode	650	Upload a Web Shell to a Web Server						

### 5.11. Risk Analysis of the Filtered Attacks.

An effective security risk analysis will seek out, assess, and implement key safeguards. Its other primary focus is on avoiding vulnerabilities and bugs in software. Through a risk assessment, a company may get a bird's eye view of its whole portfolio of applications from the perspective of a potential attacker. For managers, this means better choices about budgets, equipment, and the use of security measures. In light of this, doing an analysis is a crucial part of any effective risk management plan.

Risk analysis was done on the identified 23 attacks that were mapped to STRIDE threat model, two parameters within the dataset was used in conducting the analysis that is “likelihood of attacks” and “typical severity”. The first step was a variable fh2 was declared to store the two

parameters of the attacks as show in Appendix 1: Code Box 63. Table 5.49 shows the output obtained.

**Table 5. 49. Risk Analysis of Filtered Attacks.**

ID	Name	Likelihood Of Attack	Typical Severity
121	Exploit Non-Production Interfaces	Low	High
180	Exploiting Incorrectly Configured Access Contr...	High	Medium
21	Exploitation of Trusted Identifiers	High	High
22	Exploiting Trust in Client	High	High
43	Exploiting Multiple Input Interpretation Layers	Medium	High
50	Password Recovery Exploitation	Medium	High
679	Exploitation of Improperly Controlled Registers	Medium	High
215	Fuzzing for application mapping	High	Low
28	Fuzzing	High	Medium
650	Upload a Web Shell to a Web Server	Medium	High

### 5.11.1. Quantitative Risk Analysis.

Quantitative risk analysis involves the use of numeric values to rank the level of various risks to the technological infrastructure. Since the logs in CAPEC were qualitative, they are converted into quantitative values as shown Appendix 1: Code Box 64 . To start the process, a duplicate of the 23 found attacks is made so that referencing to the original version of the data could be easily done. Using a 4\*4 risk matrix each qualitative value was assigned its respective quantitative value. Table 5.50 shows the sample output.

**Table 5. 50. Quantitative Risk Analysis of the Filtered Attacks.**

ID	Name	Likelihood Of Attack	Typical Severity
121	Exploit Non-Production Interfaces	1	3
180	Exploiting Incorrectly Configured Access Contr...	3	2
21	Exploitation of Trusted Identifiers	3	3
22	Exploiting Trust in Client	3	3
43	Exploiting Multiple Input Interpretation Layers	2	3
50	Password Recovery Exploitation	2	3
679	Exploitation of Improperly Controlled Registers	2	3
215	Fuzzing for application mapping	3	1
28	Fuzzing	3	2
650	Upload a Web Shell to a Web Server	2	3

To generate the risk, score the values are obtained by the product of the likelihood of attack and typical severity as illustrated in Appendix 1: Code Box 65 then stored in the *filtfh4* dataset. Table 5.51 shows the risk score obtained.

**Table 5. 51. Risk Score of the Filtered Attacks.**

ID	Name	Likelihood Of Attack	Typical Severity	Risk Score
121	Exploit Non-Production Interfaces	1	3	3
180	Exploiting Incorrectly Configured Access Contr...	3	2	6
21	Exploitation of Trusted Identifiers	3	3	9
22	Exploiting Trust in Client	3	3	9
43	Exploiting Multiple Input Interpretation Layers	2	3	6
50	Password Recovery Exploitation	2	3	6
679	Exploitation of Improperly Controlled Registers	2	3	6
215	Fuzzing for application mapping	3	1	3
28	Fuzzing	3	2	6
650	Upload a Web Shell to a Web Server	2	3	6

**Table 5. 52. Quantitative risk analysis of all the filtered attacks**

ID	Name	Likelihood Of Attack	Typical Severity
468	Generic Cross-Browser Cross-Domain Theft	2	2
121	Exploit Non-Production Interfaces	1	3
149	Explore for Predictable Temporary File Names	2	2
160	Exploit Script-Based APIs	2	2
180	Exploiting Incorrectly Configured Access Control Security Levels	3	2
21	Exploitation of Trusted Identifiers	3	3
217	Exploiting Incorrectly Configured SSL	1	2
22	Exploiting Trust in Client	3	3
43	Exploiting Multiple Input Interpretation Layers	2	3
50	Password Recovery Exploitation	2	3
663	Exploitation of Transient Instruction Execution	1	4
681	Exploitation of Improperly Configured or Implemented Memory Protections	2	4
680	Exploitation of Improperly Controlled Registers	2	3
681	Exploitation of Improperly Controlled Hardware Security Identifiers	2	4
215	Fuzzing for application mapping	3	1
261	Fuzzing for garnering other adjacent user/sensitive data	2	2
28	Fuzzing	3	2
469	HTTP DoS	2	1
582	Violating Implicit Assumptions Reg...Deprecated	2	2
529	Malware-Directed Internal Reconnaissance	2	2
192	Protocol Analysis	1	1
621	Analysis of Packet Timing and Sizes	2	1
650	Upload a Web Shell to a Web Server	2	3

Out of the 23 attacks 13 attacks had missing values either in likelihood of attack or typical severity hence it was hard to compute a risk score. To solve the problem, all the missing values

were replaced with the average score which is 2 as shown in Appendix 1: Code Box 66 and the generated output in Table 5.52.

To generate the final quantitative risk, score the Appendix 1: Code Box 67 and Table5.53 gives the final output of the quantitative risk score.

**Table 5. 53. Quantitative Risk Score of all the Filtered Attacks.**

ID	Name	Likelihood Of Attack	Typical Severity	Risk Score
468	Generic Cross-Browser Cross-Domain Theft	2	2	4
121	Exploit Non-Production Interfaces	1	3	3
149	Explore for Predictable Temporary File Names	2	2	4
160	Exploit Script-Based APIs	2	2	4
180	Exploiting Incorrectly Configured Access Control Security Levels	3	2	6
21	Exploitation of Trusted Identifiers	3	3	9
217	Exploiting Incorrectly Configured SSL	1	2	2
22	Exploiting Trust in Client	3	3	9
43	Exploiting Multiple Input Interpretation Layers	2	3	6
50	Password Recovery Exploitation	2	3	6
663	Exploitation of Transient Instruction Execution	1	4	4
681	Exploitation of Improperly Configured or Implemented Memory Protections	2	4	8
680	Exploitation of Improperly Controlled Registers	2	3	6
681	Exploitation of Improperly Controlled Hardware Security Identifiers	2	4	8
215	Fuzzing for application mapping	3	1	3
28	Fuzzing	3	2	6
582	Violating Implicit Assumptions	2	2	4
469	HTTP DoS	2	1	2
582	Violating Implicit Assumptions Reg...Deprecated	2	2	4
529	Malware-Directed Internal Reconnaissance	2	2	4
192	Protocol Analysis	1	1	1
621	Analysis of Packet Timing and Sizes	2	1	2
534	Upload a Web Shell to a Web Server	2	3	6



### 5.11.2. Qualitative Risk Analysis.

The final risk score was mapped to its respective qualitative value as shown in Appendix 1: Code Box 68 and the output obtained as shown in the Table 5.54

**Table 5. 54. Qualitative Risk Mapping Score of all the Filtered Attacks.**

ID	Name	Likelihood Of Attack	Typical Severity	Risk Score
468	Generic Cross-Browser Cross-Domain Theft	2	2	Medium
121	Exploit Non-Production Interfaces	1	3	Low
149	Explore for Predictable Temporary File Names	2	2	Medium
160	Exploit Script-Based APIs	2	2	Medium
180	Exploiting Incorrectly Configured Access Contr...	3	2	Medium
21	Exploitation of Trusted Identifiers	3	3	High
217	Exploiting Incorrectly Configured SSL	1	2	Low
22	Exploiting Trust in Client	3	3	High
43	Exploiting Multiple Input Interpretation Layers	2	3	Medium
50	Password Recovery Exploitation	2	3	Medium
663	Exploitation of Transient Instruction Execution	1	4	Medium
679	Exploitation of Improperly Configured or Imple...	2	4	High
680	Exploitation of Improperly Controlled Registers	2	3	Medium
681	Exploitation of Improperly Controlled Hardware...	2	4	High
215	Fuzzing for application mapping	3	1	Low
261	Fuzzing for garnering other adjacent user/sens...	2	2	Medium
28	Fuzzing	3	2	Medium
469	HTTP DoS	2	1	Low
582	Violating Implicit Assumptions Reg...	2	2	Medium
529	Malware-Directed Internal Reconnaissance	2	2	Medium
192	Protocol Analysis	1	1	Low
621	Analysis of Packet Timing and Sizes	2	1	Low
650	Upload a Web Shell to a Web Server	2	3	Medium

Table 5.55 shows the 23 attacks categorized in relation to High, Medium and low Risk attacks

**Table 5. 55. High, Medium, low Risk attacks**

ID	High Risk Attacks
21	Exploitation of Trusted Identifiers
22	Exploiting Trust in Client
679	Exploitation of Improperly Configured or Imple...
681	Exploitation of Improperly Controlled Hardware...
ID	Medium Risk Attacks
468	Generic Cross-Browser Cross-Domain Theft
149	Explore for Predictable Temporary File Names
160	Exploit Script-Based APIs
180	Exploiting Incorrectly Configured Access Contr...
43	Exploiting Multiple Input Interpretation Layers
50	Password Recovery Exploitation
663	Exploitation of Transient Instruction Execution
680	Exploitation of Improperly Controlled Registers
261	Fuzzing for garnering other adjacent user/sens...
28	Fuzzing
582	Violating Implicit Assumptions Reg...
529	Malware-Directed Internal Reconnaissance
650	Upload a Web Shell to a Web Server
ID	Low Risk Attacks
217	Exploiting Incorrectly Configured SSL
215	Fuzzing for application mapping
469	HTTP DoS
192	Protocol Analysis
621	Analysis of Packet Timing and Sizes

## 5.12. Defense Mechanism for the Attacks

The following Code in Appendix 1: Code Box 69 was executed to extract the defense mechanisms available for the 23 attacks. Table 5.56 shows the defense mechanism extracted. Out of the 23 attacks only 17 had defense mechanisms six attacks namely Exploit Script-Based APIs, Explore for Predictable Temporary File Names, fuzzing for garnering other adjacent user/sensitive data, Generic Cross-Browser Cross-Domain Theft, Protocol Analysis, and Violating Implicit Assumptions Regarding XML Content (aka XML Denial of Service (XDoS)) had no defense mechanisms.

**Table 5. 56. Defense Mechanisms for the Filtered Attacks.**

Attack	Defense Mechanism
Analysis of Packet Timing and Sizes	The solution here is to distort packet sizes and timing at the VPN layer by adding padding to normalize packet sizes and timing delays, this will reduce information leakage through timing.
Exploit Non-Production Interfaces	Ensure that non-production interfaces are not present in production systems and that these interfaces are exclusively utilized in development environments.
Exploitation of Improperly Configured or Implemented Memory Protections	Make sure that the protected and unprotected memory ranges are separated and do not overlap. If memory areas must overlap, take use of memory priority methods. Ensure that the original and mirrored memory areas are protected in the same way. Ensure that only immutable code or data is stored in ROM or write-once memory.
Exploitation of Improperly Controlled Hardware Security Identifiers	Examine the production of security IDs for anomalies in design and frequent flaws. Examine security identifier decoders for design flaws and frequent flaws. In both pre-silicon and post-silicon contexts, test security identifier definition, access, and programming flow.
Exploitation of Improperly Controlled Registers	<p>Create appropriate access control rules for software access to hardware registers and ensure that these policies are implemented in line with the design specifications. Check security lock bit safeguards for design discrepancies and common flaws. In both pre-silicon and post-silicon contexts, test the security lock programming flow. Use automated techniques to ensure that values cannot be reprogrammed and that write-once fields are locked on writing zeros.</p> <p>Ensure that measurement data is kept in read-only registers or has access restrictions to avoid tampering by an untrusted agent.</p>
Exploitation of Transient Instruction Execution	<p>Implement DAWG (Dynamically Allocated Way Guard) - processor cache properly divided between different programs/processes that do not share resources::</p> <p>Implement KPTI (Kernel Page-Table Isolation) to completely separate user-space and kernel space page tables.</p> <p>Configure Architectural Design of Microcode to limit abuse of speculative execution and out-of-order execution::</p> <p>Disable Shared Array Buffer for Web Browsers.</p> <p>Disable Copy-on-Write between Cloud Virtual Machines.</p> <p>Configure Privilege Checks on Cache Flush Instructions.</p> <p>Implement Non-inclusive Cache Memories to prevent Flush+Reload Attacks</p>
Exploitation of Trusted Identifiers	<p>Utilize strong federated identity such as SAML to encrypt and sign identity tokens in transit.</p> <p>Use industry standards session key generation mechanisms that utilize high amount of entropy to generate the session key. Many standard web and application servers will perform this task on your behalf.</p> <p>If the identifier is used for authentication, such as in the so-called single sign on use cases, then ensure that it is protected at the same level of assurance as authentication tokens.</p> <p>If the web or application server supports it, then encrypting and/or signing the identifier (such as cookie) can protect the ID if intercepted.</p> <p>Use strong session identifiers that are protected in transit and at rest.</p>

Utilize a session timeout for all sessions, for example 20 minutes. If the user does not explicitly logout, the server terminates their session after this period of inactivity. If the user logs back in then a new session key is generated.

Verify authenticity of all identifiers at runtime.

Exploiting Incorrectly Configured Access Control Security Levels	Configure the access control correctly.
Exploiting Incorrectly Configured SSL	Usage of configuration settings, such as stream ciphers vs. block ciphers and setting timeouts on SSL sessions to extremely low values lessens the potential impact. Use of later versions of TLS (e.g. TLS 1.1+) can also be effective, but not all clients or servers support the later versions.
Exploiting Multiple Input Interpretation Layers	<p>An iterative approach to input validation may be required to ensure that no dangerous characters are present. It may be necessary to implement redundant checking across different input validation layers.</p> <p>Ensure that invalid data is rejected as soon as possible and do not continue to work with it. Make sure to perform input validation on canonicalized data (i.e. data that is data in its most standard form). This will help avoid tricky encodings getting past the filters. Assume all input is malicious.</p> <p>Create an allow list that defines all valid input to the software system based on the requirements specifications. Input that does not match against the allow list would not be permitted to enter into the system.</p>
Exploiting Trust in Client	<p>Ensure that client process and/or message is authenticated so that anonymous communications and/or messages are not accepted by the system.</p> <p>Do not rely on client validation or encoding for security purposes.</p> <p>Utilize digital signatures to increase authentication assurance.</p> <p>Utilize two factor authentication to increase authentication assurance.</p> <p>Perform input validation for all remote content.</p>
Fuzzing	<p>Test to ensure that the software behaves as per specification and that there are no unintended side effects.</p> <p>Ensure that no assumptions about the validity of data are made. Use fuzz testing during the software QA process to uncover any surprises, uncover any assumptions or unexpected behavior.</p>
Fuzzing for application mapping	Construct a 'code book' for error messages. When using a code book, application error messages aren't generated in string or stack trace form, but are catalogued and replaced with a unique (often integer-based) value 'coding' for the error. Such a technique will require helpdesk and hosting personnel to use a 'code book' or similar mapping to decode application errors/logs in order to respond to them normally. Wrap application functionality (preferably through the underlying framework) in an output encoding scheme that obscures or cleanses error messages to prevent such attacks. Such a technique is often used in conjunction with the above 'code book' suggestion. Obfuscate server fields of HTTP response. Hide inner ordering of HTTP response header. Customizing HTTP error codes such as 404 or 500. Hide HTTP response header software information filed. Hide cookie's software information filed. Obfuscate database type in Database API's error message.
HTTP DoS	Configure web server software to limit the waiting period on opened HTTP sessions Use load balancing mechanisms
Malware-Directed Internal Reconnaissance	<p>Keep patches up to date by installing weekly or daily if possible.</p> <p>Identify programs that may be used to acquire peripheral information and block them by using a software restriction policy or tools that restrict program execution by using a process allow list.</p>

Password Recovery Exploitation	<p>Use multiple security questions (e.g. have three and make the user answer two of them correctly).</p> <p>Let the user select their own security questions or provide them with choices of questions that are not generic.</p> <p>E-mail the temporary password to the registered e-mail address of the user rather than letting the user reset the password online.</p> <p>Ensure that your password recovery functionality is not vulnerable to an injection style attack.</p>
Upload a Web Shell to a Web Server	<p>Make sure your web server is up-to-date with all patches to protect against known vulnerabilities.</p> <p>Ensure that the file permissions in directories on the web server from which files can be execute is set to the least privilege settings, and that those directories contents is controlled by an allow list.</p>

## CHAPTER 6

### DISCUSSIONS AND FINDINGS

#### 6.1. Summary of Findings

The overall objective of this study was to develop a holistic security pattern-based model for the network architecture and test its adoption towards the enhancement of network security. To achieve this the study assessed the techniques, models, frameworks that guide in the design and development of a secure network architecture. conceptualized and developed security pattern-based model for the network architecture, tested the conceptual security pattern-based model and evaluated the generated patterns contribution to network security assurance.

The first specific objective was addressed by conducting an extensive literature review to identify existing techniques, models, and frameworks related to secure network architecture. Identified limitations in the existing literature that the study aimed to address.

the second specific objective was addressed, based on the literature review, the study conceptualized a security pattern-based model for network architecture by defining the key components and principles of the proposed model which constituted the extended OSI, the three-layer Network Security Domain, CAPEC Pattern Repository, STRIDE threat model, and Risk Assessment which were used to develop the model and clearly articulated how the model addressed the problem of network attacks security

In the third specific objective was achieved by conducting a simulation of the developed security pattern-based model with scenarios to test the model's effectiveness in enhancing network security and Gathered data on the model's performance.

The fourth specific objective was achieved by defining the specific criteria for evaluating the contribution of generated patterns to network security assurance, Analyzing the results against the defined evaluation criteria and then discussing the strengths and weaknesses of the generated security patterns

As discussed in the chapter 5 for the purpose of testing the model it was split in three stages as illustrated in Figure 5.1, 5.2 and 5.3. With their respective input and output processes. In the first stage of the model Figure 5.1 the output was to identify the attacks and their respective targeted attacked network surface. Feature selection process using *the select Kbest feature*

*selection algorithm* was conducted on the dataset to identify the appropriate features which was subsequently subjected to machine learning algorithms (*KNeighbors classifier*, *Random Forest classifier* and *GaussianNB*) to achieve the goal in this process. it was noted the 50.5 % (11214) of the attacks targeted the Host layer, 30.5 % (6770) targeted the user layer while 19 % (4231) targeted the media layer.

From UNSW-NB dataset its shows that a higher percentage of the attack were mainly targeting the upper layers of the network architecture that is transport, session, presentation and application layers, followed by attacks that target the users of the network and finally attacks on network hardware. From these findings it depicts generally what you would “typically” expect in terms of attacks in a typical network, since for a hacker the low hanging fruits for target are mainly the upper layers because their mode of operation and communication the cuts across networks and the internet offers a wide surface area of attacks followed by users and finally the hardware since they a hard to access unless you are an insider who is authorized. In relation to risk assessment more emphasis on protection should be placed on the upper layers of this network though it should be noted that networks are different and business operations are different hence attack surface vary hence every network should be analyzed to determine which layer is highly susceptible to attacks.

In the first stage of the model the study also explored the attacks and their distribution on each surface as shown in visualization Figure 5.20 and percentage distribution in each surface on Table 5.37. Generally, the exploits constituted 48% of the attacks followed by generic (22.7%), Fuzzers (12.2%), Reconnaissance (7.69%), Dos (5.02%), Backdoor (3.01%), analysis (0.6%), Shellcode (0.33%) and worms (0.11%). Exploits attacks generally constitute infiltrations or codes that would take advantage of applications security vulnerabilities they are payloads that are generally included within malwares or processes. According to survey conducted by (InfoSec Insights, 2020) the leading motivation of hackers is financial gain, Verizon’s 2020 Data Breach Investigations Report (DBIR) shares that 86% of the data breaches they analyzed were financially motivated through breach of databases and application using malware (Verizon, 2021).

The second stage of the model in Figure 5.2 the output was to identify relevant attacks. To achieve the output in Figure 5.1 (first stage) which is the input in the second stage in Figure

5.2 was subjected to the CAPEC repository to check for their existing attack patterns which was further mapped to STRIDE categories eventually generating relevant attacks. Using a string-based approach a search of the nine categories of attacks as identified in Figure 5.1 (first stage) were done within the CAPEC repository and their respective find of 25 attack patterns shown in Table 5.39 (1 *generic attack*), Table 5.40 (14 *exploits attacks*), Table 5.41 (4 *Fuzzers attacks*), Table 5.42 (3 *Dos attacks*), Table 5.43 (1 *Reconnaissance attacks*), Table 5.44 (2 *Analysis attacks*) and Table 5.45 (1 *Shell Attack*). It should be noted that two attacks namely *backdoors* and *worms* that were in UNSW-NB attacks did not yield any attack patterns within CAPEC repository. Lastly within this part of model a mapping of the 25 CAPEC patterns to STRIDE was done out of where by only two patterns ID: 122 and 582 could not be mapped due to it being deprecated within the repository as shown in Table 5.47.

The third stage Figure 5.3 was to generate defensive patterns in this process a risk analysis process was conducted on the patterns to identify the risk levels of each attack as shown in Table 5.55 Later defense mechanisms for the patterns were generated as shown with Table 5.56.

## **6.2. Generation of Worms and Backdoor Patterns for CAPEC Repository.**

Since the worm and backdoor attack patterns were missing from the CAPEC the study set out to generate the pattern for the two attacks. The patterns in Table 6.1 and 6.2 below were based from literature, concerning the worms the researcher looked at infamous worms, structure of worms, how worms attack, their propagation and activation methods, worm's anti detection techniques, categories of worms, how worms are designed, how to detect worms and defenses against worms. In relation to backdoors the researcher looked how backdoor are used to attack, how they are design, various backdoor techniques and defenses against backdoors respectively.



## 6.2.1. General Worm Pattern

Table 6.1. shows a general worm attack generated by the researcher.

**Table 6. 1. Worm Attack Pattern and Defense**

Name	Worm Attacks
Abstraction	Detailed
Status	Draft
Description	"A worm is a program, which can self-replicate and propagate over the network, with or without human intervention, and has malicious intent. Worms are malware that self-replicate and infect other computers while remaining active on compromised systems. They replicate to infect uninfected machines. It achieves this by abusing automated and unnoticed portions of an operating system. Worms are usually discovered when their uncontrolled reproduction consumes system resources, delaying or stopping other operations.
Alternate Terms	
Likelihood Of Attack	High
Typical Severity	High
Related Attack Patterns	This is a parent pattern.
Execution Flow	The attack takes advantage of an enabling vulnerability in the network, hosts and operating system and installs itself. After gaining access to devices, a worm replicates and selects new targets. After the worm infects the device, the attacker has access to the host—often as a privileged user. Attackers use a local exploit to escalate their privilege level to the administrator level.
Prerequisites	For email worms it requires naivety of email users and lack of email malware filter. For P2P/ file sharing worms is unregulated file sharing software For IM worms click baits through the chat software For internet work is vulnerabilities in the OS such as weak passwords
Skills Required	Generally Social engineering and deception skills from the side of the attacker, conducting network scanning and sniffing.
Resources Required	Unpatched host, High speed Internet, compromised email system, privileged file system properties like read/write.
Indicators	You receive an e-mail or a chat from an entity that that poses as a system administrator that you need to perform an update by downloading a particular software and the link is provided.
Consequences	<b>Scope:</b> Confidentiality <b>Technical Impact:</b> Access to confidential data, Staling of data <b>Scope:</b> Availability <b>Technical Impact:</b> deletion of files, Denial of service <b>Scope:</b> Integrity <b>Technical Impact:</b> alteration of files

Mitigations	<p><b>Detection techniques.</b></p> <p><b>Traffic analysis:</b> Check for growth in traffic volume especially exponential hits on servers, Check for rise in number of scan sweeps especially exponential in unique sources, Check for change in traffic patterns for hosts within the network. Resources required include Packet capture tools, Packet capture tools.</p> <p><b>Honey Pot Monitoring:</b> Set up host with services configured for a worm attack. Take a snapshot using low level tools to provide basic a baseline measurement that can be used any alteration. Use tripwire or any other related tool for recursive examination of file properties, any change on attributes should signal a malicious activity. Use a network monitor to log all inbound and outbound traffic from the host to look for suspicious packet, you can either employ snort or real secure tools to achieve this process. Isolate the host for offline analysis or mount the disk image to another host for analysis low level tools like coroner can be used to achieve this process.</p> <p><b>Black Hole Monitoring:</b> Identify locally unused subnets within your address space and route them to a single router. Monitor the number of request for access to the unallocated network space, using internal routers that advertise routes. View the network or the subnet as hole and anything that is goes into it as a traffic of interest. Using scanners monitor all traffic including SYN packets from worms for the basis of worm analysis. Use LaBrea tool to capture the first data packet of the connection and use it to classify the type of traffic. Data for analysis of this process can be sourced from the exported flow logs of routers and layer three switches, which gives the information about attempts to access the unallocated network space. Place passive network monitor at both the entrance of the network and the router interface that serves the network black hole to collect suspicious traffic.</p> <p><b>Signature based Detection:</b></p> <p><b>Network Payload Signatures:</b> Analyze payload matching signatures based on string comparison of application protocols and network characteristics to detect malicious patterns.</p> <p><b>Log file analysis:</b> analyze the log files both application and system logs to finger print the behavior of a worm by looking at errors issued when it probes a machine.</p> <p><b>File signature analysis:</b> use anti malware tools which contains most of the signature analysis tools</p> <p>Use <i>chkrootkit</i>, to recursively analyzes files on the system and examines for known malicious files and patterns.</p> <p><i>check_wtmpx</i>, to examine the integrity of the login logfiles for signs of tampering.</p> <p><i>chklastlog</i>, to examine the integrity of the file for signs of modification.</p> <p><i>chkproc</i>, a binary tool that looks for differences between the observable process table and the actual kernel mappings in the /proc Filesystem. Differences would</p>
-------------	---

be indicative of a kernel module designed to hide the presence of attackers or malware.

*ifpromisc* and *strings*, two small auxiliary applications that can be used to establish a trustworthy baseline for the entire process.

### **Defense techniques.**

#### **Host based:**

Implement host based firewalls to act as failover for the network firewall. Perform course grained configuration on network ports and services that should be accessed so that worm cannot access the system through unauthorized paths. Perform fine grained configuration on which hosts are allowed to connect to the services

Implement anti-virus with an up to date definitions for both clients and servers to detect worm executables and either quarantine them or remove them from systems on the network. For large networks implement a centralized AV solution for easy of management of configuration and push mechanisms for definitions.

Do not let servers run with system level rights they should be left on unprivileged user ID unless when doing configurations, implement by binding a reserved listening socket to accept inbound connections for system to run in a limited privilege space. Configure ACLs dropping privileges to immediately revoke the elevated user ID value once the operations at that level are complete

On the processes running on elevated rights have the child processes to handle the workload under the control of a privileged process

Sand boxing of applications i.e. portioning system access for applications to minimize any damage an attacker will make to a small subset of the file system

Use network scanners to identify and disable unneeded services and features to reduce the exposure of services running on any host

Patch known vulnerable holes

Set baseline measurements to establish the normal traffic levels for a network and its component hosts to detect hosts that are aggressively seeking new targets.

#### **Network Based Defenses:**

Setup and configure both perimeter firewall and subnet firewalls for failover purposes

Configure firewall security policy that blocks inbound access to workstation systems and denies external access to unauthorized servers not under central administrative control

	<p>Identify the hosts that need to be accessed from the external network, then restrict communication between these hosts and the trusted internal clients, and filter services that do not need to receive external access</p> <p>Implement proxy-based defenses to act as a holding space for data temporarily as it awaits being transferred to the client, while there it can be monitored and selectively passed or modified to remove objectionable material, such as attack data or a potentially malicious payload.</p> <p>Attacking the worm to disable or stagnate its spread like messaging the network to shut down, Forge replies to a query that listening port's the worms are using to shut down.</p> <p>Implement User training and awareness programs to educate users on dangers and how to detect them.</p>
Example Instances	Using electronic mail and exploiting a known vulnerability in the e-mail client; Spreading by open Windows networking file shares, infecting the file system on the target computer; Attacking Web clients by uploading an exploit to the home page of an infected site.
Related Weaknesses	Overflow Buffers (100):: Clickjacking(103):: Brute Force(112):: Sniffing Network Traffic(158)::
Taxonomy Mappings	Worm attacks
Notes	

## 6.2.2. General Backdoor Attack Pattern

**Table 6. 2. General Backdoor Attack Pattern**

Name	Backdoors attack
Abstraction	Standard
Status	Draft
Description	Backdoor attacks seek to circumvent any security measures; and Acquire root access, i.e., administrator privileges, to a system. Once a malicious actor has access to your network or endpoints, they can run a succession of more sophisticated malware. They may do so in stages. An attacker begins with a modest degree of access and gradually increases their privileges until they have complete access to your systems. This sequential method assists the attacker in evading discovery. Once the backdoor is installed, thieves will be able to monitor your every move and get access to your devices. They may take complete control of your digital life without your knowledge.
Alternate Terms	
Likelihood Of Attack	High
Typical Severity	High
Related Attack Patterns	This is a Parent Pattern
Execution Flow	This attacks begin with the discovery of a weak spot or compromised program on a device to exploit, this might be a weakness in an application, an unsecured port on the network, an account with a weak password, or a piece of malware placed on a device. The attacker then employs sophisticated techniques to convince your device, network, or online account that the backdoor is a legitimate program. Once the device has been hacked, the backdoor can be used to install malware such as cryptojackers, rootkits, or ransomware, steal data and monitor user behavior, or just crash your device.
Prerequisites	Weak login credentials, physical access to the machine itself, phishing tricks, unpatched anti malware, Trojan malwares, Key loggers.
Skills Required	The attacker must have Social engineering skills to trick the user into installing a Trojan to open a backdoor, programing skills to perform processes like <b>remote file inclusion</b> , Without extra work or expertise, an attacker can utilize privileged features to which they already have access. The attacker is simply needs to have access to a compromised account. Certain more advanced attacks may involve knowledge of protocols and probing techniques that aid in variable control. A malevolent person may attempt to comprehend the authentication method in order to circumvent it.

	The attacker may also requires real-time access to network traffic in order to extract required information from the stream. While there are tools available to automate some operations, properly utilizing these technologies in an attack scenario demands a thorough grasp of the underlying principles.
<b>Resources Required</b>	A tool capable of displaying network communication flow like , Abel, tcpdump Wireshark, , Cain etc. must also be able to convey their phishing plan to victims (through email, instant chat, or other means), as well as a website or other platform on which victims may submit personal information.
<b>Indicators</b>	Your personal files are encrypted and you are request to pay to be given a decryption key, high consumption of network resources the CPU and RAM without the corresponding PIDs,
<b>Consequences</b>	SCOPE: Confidentiality TECHNICAL IMPACT: Sniff password. SCOPE: Access Control, TECHNICAL IMPACT: Gain Privileges SCOPE: Authentication, TECHNICAL IMPACT: Bypass Protection Mechanism  SCOPE: Confidentiality: Read Data SCOPE: Access Control: TECHNICAL IMPACT: Read Data: SCOPE: Authorization TECHNICAL IMPACT: Execute Unauthorized Commands: SCOPE: Access Control: TECHNICAL IMPACT: Modify Data: SCOPE: Integrity TECHNICAL IMPACT: Alter Execution Logic:
<b>Mitigations</b>	<b>Detection techniques.</b>  Deploy Anti-malware firewalls to scan files, URLs, and processes against threat databases employ the following steps  On the infected system, install the antivirus software. If the antivirus was already installed on the computer when it became infected, replace it. Verify that both the program and the virus definitions are current. Turn off your computer's internet connection. Detach the network adapter from the computer and physically disconnect the cord. Turn down the router and/or modem as well if you suspect router malware. Reboot into safe mode and do a thorough system scan. Reboot your computer and do another complete system scan. Include any network devices. Restore your computer to a previous date to undo any changes done to files by the backdoor malware. Repeat steps 1-6 for each device connected to the network. Backdoor infections replicate similarly to worms, therefore scan the whole network before lowering the red warning level. Deploy firewalls to continuously monitor all operating programs, communication requests, and file changes.

	<p>Locate the malware through an iterative process of discovery and analysis</p> <p>Deploy VPN with built-in backdoor detection and removal tool</p> <p>In case of a supply chain attacks and hardware backdoors Install a firmware version that is not susceptible to the backdoor. If that fails, you'll need to dispose of the infected hardware.</p> <p>Ascertain that the Cybersecurity team conducts a thorough examination of the site's access records for anything unusual.</p> <p><b>Defense techniques.</b></p> <p>Avoid purchasing hardware from dubious sources. Ascertain that everything you purchase is covered by a manufacturer and seller guarantee.</p> <p>Avoid relying on the system's default login credentials. A strong password that is unique to you is the best defense against backdoors and viruses. Whenever feasible, use multi-factor authentication.</p> <p>Downloading files or installing software from untrusted sources is not recommended.</p> <p>Avoid connecting to the Internet using insecure public connections. Purchase VPNs immediately to secure your connection.</p> <p>Maintain an up-to-date antivirus and perform frequent complete system scans.</p> <p>By periodically upgrading the firmware, you may repair vulnerabilities.</p> <p>Conduct code reviews to detect and seal vulnerabilities related to programing logic</p> <p>Conduct network scans to identify unsecured and unfiltered port and remediate</p> <p>Enforce password complexity policy through a centralized access control directory services.</p>
<p><b>Example Instances</b></p>	<p>Audio and video records can be sent from an enemy to a C2 server or a similar device. A Car Whisperer attack allows an attacker to record and collect sounds from a vehicle's audio peripherals. When a vehicle's Bluetooth hands-free system is in pairing mode, an intruder may try to connect to it if they are nearby. As long as an authentication mechanism is in place, an attacker may be able to play music or voice recordings, as well as start a recording and record conversations taking place within the car. The pairing security key must be reset</p>

	<p>to its default value or brute-forced for authentication to be successful (which may be less practical in an outside environment) According to the level of sensitivity, this scenario might be exceedingly dangerous. If an enemy is within 10-15 meters of the target device, he or she can utilize a technique known as Bluebugging, which is similar to Bluesnarfing. As a result of Bluebugging, attackers are able to listen to and record calls, forward calls, send SMS, and access the phonebook.</p> <p>Automated software upgrades are distributed to government and commercial users that contain a concealed backdoor by a subcontractor working for the developer of the product.</p> <p>An attacker having access to and the capacity to change the configuration/programming of FPGAs in organizational systems presents a Trojan backdoor that may be exploited to alter the behavior of the original system, potentially jeopardizing the secrecy of data being processed.</p>
Related Weaknesses	Authentication Abuse (114)::Authentication Bypass (115) ::Command Injection(248):: Altered Installed BIOS(532):: Data Injected During Configuration(536):: Altered Component Firmware(638):: Alteration of a Software Update(669)::Design for FPGA Maliciously Altered(674).
Taxonomy Mappings	Backdoor attacks
Notes	

Table 6.2 shows a general backdoor attack pattern generated by the researcher.

### 6.3. Qualitative Analysis of Worms and Backdoor Patterns.

The developed worm and backdoor patterns are analyzed in this section, and two unique criteria are used to evaluate them as described below.

Security patterns are used as a means of resolving specific forces. They're designed to deal with security risks and should be adaptive, effective, and simple to implement and use, thus we're attempting to identify these common factors that can be applied to any new security pattern and used as a baseline for a qualitative analysis. These criteria, as well as the outcomes of the comparison, are detailed in Section 5.3.1.

The second set of requirements may be stated as the ability of a certain security pattern to respond to various types of attacks as outlined by (Howard & LeBlanc, 2003).The STRIDE model was proposed by Howard and LeBlanc to represent the many types of attacks that might



occur within a system. We assess the worm and backdoor security patterns based on how effectively a network system adopting a given security pattern would respond to each probable attack type. Section 5.3.2. Provides a quick overview of the attack categories based on the STRIDE model.

### **6.3.1. Evaluating Worm and Backdoor Patterns on The Basis Of Their Forces**

The patterns were initially evaluated using the following criteria, which are often included in the forces associated with each of these patterns:

*Adaptability:* When a pattern can accept shifting threats, it is considered to be adaptable.

For instance, the worm and backdoor patterns should be capable of defending against any multi stage attacks from Media Layer all the way to user layer.

*Auditability:* A pattern is considered to be auditable if it includes a mechanism for maintaining an audit trail. For instance, if all actions are documented, and an external attack occurs, an audit trail facilitates the examination of the events that precipitated the attack, uncover evidence of the attack, and provide recommendations on how to enhance the system. The audit trail may not be defined as part of the pattern itself, but may be included as a feature of a common system. What important is the ability to collect valuable information within the pattern's classes.

*Complexity:* If a pattern is not overly complicated, it is simple to handle. However, if we employ a particularly sophisticated security pattern, the administrators who apply the pattern in a real-time context may have difficulty establishing the system, which can sometimes lead to security flaws.

*Cost:* We can estimate the cost of adopting a security pattern. Depending on deployment requirements or the need for specialist assistance, various patterns or variants of the same pattern may have varied prices. Depending on the application, we might choose the one with the lowest cost, even if it is not as secure as another.

*Deployability:* We can figure out how easy it will be to use the units of these patterns in a real-world system. Because of this, the overhead will change.

*Effectiveness:* There is a good chance that a pattern is going to work well if the infrastructure provided by it can handle a certain type of threat. For example, a worm security pattern should be able to detect a wide range of worm attacks and respond to them in the right way.

*Modifiability:* A pattern is changeable or extendable if it can rapidly be tweaked or expanded to accommodate new threats, functions, or variants of its functionalities.

*Overhead:* The amount of overhead in a pattern may be calculated. A security pattern could contain methods that take a long time, need a long series of operations, or require the exchange of numerous messages, for example. If the overheads of two patterns or variants of the same pattern are different, one pattern may be selected for a given application.

*Scalability:* A pattern is considered scalable if it can easily accommodate extra users, entries, or stakeholder participants. If a corporation uses a security pattern for day-to-day email security, the pattern's units should be able to manage the circumstance when the number of workers or email traffic doubles in a short period of time.

*Usability:* Usability refers to how easily a security pattern's unit may be configured and utilized in a real-world context. Administrators will make mistakes if the pattern does not give a clear security picture.

**Table 6. 3. Evaluating Worm and Backdoor Patterns on The Basis Of Their Forces.**

Pattern	Adaptability	Auditability	Complexity	Cost	Deployability	Effectiveness	Modifiability	Overhead	Scalability	Usability
Worm	H	H	H	M	M	H	H	M	L	H
Backdoor	M	H	H	M	M	H	H	M	H	M

Legend: **L** – Satisfies Low; **M** – Satisfies Medium; **H** – Satisfies High

### **6.3.2. Evaluation of Worm and Backdoor Patterns Based On STRIDE.**

The criteria for this evaluation may be stated as how effectively a certain security pattern responds to various types of attacks as outlined by (Howard & LeBlanc, 2003). The STRIDE model was proposed by Howard and LeBlanc to represent the many types of attacks that might occur in a system. STRIDE categories have been explained in Chapter 2 Table 2.8.

Table 5.4 lists the criteria for evaluating the worm and backdoor patterns using the STRIDE model to determine attack types. However, we have specified the assessment parameters, which are D P and E which represents, is it possible to detect an attack using the security pattern? Is it possible for the security pattern to protect against the attack in question? Is it capable of providing extra or enhanced security? Respectively.

**Table 6. 4. Evaluating the Worm and Backdoor Patterns Using STRIDE**

PATTERN	SPOOFING	TAMPERING	REPUUDIATION	INFORMATION DISCLOSURE	DENIAL OF SERVICE	ELEVATION OF PRIVILEGE
Worm	E	P	D	P	E	D
Backdoor	D	P	D	E	P	E

Legend: D-Detection of threat.  
P-Protection against threats.  
E-Enhances protection.

## CHAPTER 7

### CONCLUSION AND RECOMMENDATIONS

#### 7.1. Conclusions.

The findings of this study clearly demonstrates that the use of this model can go a long way in improving the security of network. It can greatly assist network security specialist to determine the surface of the network architecture is subjected to attacks which is currently a challenge with the convectional Network security tools. After the attack and attack surface has been identified the security specialists can leverage of on an attack pattern repository to determine the available defense mechanism after conducting a risk assessment of the identified attacks. The net effect of all this is that timely decisions can be made in relation to protecting networks against attacks and recovering faster from attacks.

#### 7.2. Contribution of the Research.

This study makes significant contributions to the existing body of knowledge in the field of network security through several key avenues. Firstly, it addresses a crucial gap by advocating for a holistic approach to network defense. This departure from the current fragmented security measures underscores the need for a coordinated and comprehensive strategy, offering a fresh perspective on enhancing network security.

A major contribution lies in the introduction of a security pattern-based model. By integrating diverse frameworks and constructs, such as the Cisco three-layer hierarchical model, OSI network architecture model, CAPEC attack pattern Repository, STRIDE threat Model, Risk Management Framework, and Pattern Theory, the study proposes a unique and adaptable model for safeguarding network architecture. This innovative approach adds depth to existing knowledge by providing a framework that goes beyond traditional, siloed security solutions.

Furthermore, the study brings empirical validation to the forefront. Adopting machine learning techniques, the research provides evidence-based insights into the effectiveness of the proposed security model. This empirical dimension offers a practical foundation for the theoretical framework, ensuring that the proposed model is not only conceptual but also demonstrably effective in real-world scenarios.

The identification and evaluation of two new attack patterns (worms and backdoors) not present in existing repositories contribute to the evolving understanding of cyber threats. This recognition of potential blind spots in current security models emphasizes the need for continuous evolution in defense strategies to effectively counter emerging threats.

Layer-specific analysis adds granularity to the study's contributions. By examining the distribution of cyber-attacks across different network layers and categorizing attacks by exploit type, the research provides nuanced insights. This detailed understanding allows for more targeted and effective security measures, enhancing the practical applicability of the study's findings.

Lastly, the study contributes to knowledge synthesis by integrating diverse frameworks. Acknowledging the multifaceted nature of network security challenges, this integration provides a more comprehensive approach for practitioners and researchers alike. The study, therefore, not only expands the theoretical foundations but also offers practical insights for the development of more robust and adaptive defense strategies in the ever-evolving landscape of network security.

### **7.3. Recommendations.**

Future research should focus on refining and validating the proposed security pattern-based model through iterative testing in diverse network environments. Exploring advanced machine learning techniques, conducting longitudinal analyses of cyber threats, and integrating behavioral analytics are crucial for enhancing the model's sophistication and adaptability. Cross-industry applicability assessments, the development of effective threat intelligence sharing mechanisms, and investigations into human factors in security are essential for broader and more effective implementation. Evaluating the model's robustness against advanced attacks, conducting cost-benefit analyses, and examining regulatory compliance implications are key areas for further exploration. Additionally, a holistic assessment of the impact on end-user experience is recommended to strike a balance between robust security measures and a seamless user interface.

## REFERENCES.

- Abomhara, M., Gerdes, M., & Kjøien, G. M. (2015). A stride-based threat model for telehealth systems. *Norsk informasjonssikkerhetskonferanse (NISK)*, 8(1), 82–96.
- Abraham, R., Arora, D., Coles, M., Eckert, M., Heitman, M., Manion, A., & Yooun, A. (2015). Common vulnerability scoring system v3. 0: Specification document. *First*.
- Abrar, H., Hussain, S. J., Chaudhry, J., Saleem, K., Orgun, M. A., Al-Muhtadi, J., & Valli, C. (2018). Risk analysis of cloud sourcing in healthcare and public health industry. *IEEE Access*, 6, 19140–19150.
- Abuonji, P., & Rodrigues, A. (2018). *A Stratified Cyber Security Vigilance Model: An Augmentation of Risk-Based Information System Security*. <https://doi.org/10.14738/TNC.65.5166>
- Adams, M., & Coplien, J. (1996). *Fault-tolerant telecommunication system patterns. Pattern Languages of Program Design*. Addison-Wesley.
- Agerbo, E., & Cornils, A. (1998). How to preserve the benefits of design patterns. *Proceedings of the 13th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications*, 134–143.
- Al Sukkar, G., Saifan, R., Khwaldeh, S., Maqableh, M., & Jafar, I. (2016). *Address resolution protocol (ARP): Spoofing attack and proposed defense*.
- Alabady, S. (2009). Design and Implementation of a Network Security Model for Cooperative Network. *Int. Arab. J. e Technol*, 1(2), 26–36.
- Alberts, C., Dorofee, A., Stevens, J., & Woody, C. (2003). *Introduction to the OCTAVE Approach*. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.
- Alexander, C. (1977). *A pattern language: Towns, buildings, construction*. Oxford University Press. <https://archive.org/details/patternlanguage00chri/page/n5>
- Alexander, C. (1979a). *The Timeless Way Of Building*. Oxford University Press. <https://archive.org/details/TheTimelessWayOfBuilding/page/n1>

- Alexander, C. (1979b). *The timeless way of building* (Vol. 1). New York: Oxford University Press.
- Alferd, N. G., & M, S. (2017). *Equifax data breach may affect nearly half the US population*.  
<https://www.cnet.com/news/equifax-data-leak-hits-nearly-half-of-the-us-population/>
- Algaley, S. A., & Yousif, Y. A. (2022).
- AlgoSec, Y. B. (2013). *Uncovering the Dangers of Network Security Complexity*.  
<https://www.wired.com/insights/2013/01/uncovering-the-dangers-of-network-security-complexity/>
- Almubairik, N. A., & Wills, G. (2016). Automated penetration testing based on a threat model. *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, 413–414.
- Altman, N., & Krzywinski, M. (2017). Ensemble methods: Bagging and random forests. *Nature Methods*, 14(10), 933–935.
- Altuhhova, O., Matulevičius, R., & Ahmed, N. (2012). Towards definition of secure business processes. *International Conference on Advanced Information Systems Engineering*, 1–15.
- Alzahrani, B. A., Vassilakis, V. G., & Reed, M. J. (2013). Mitigating brute-force attacks on Bloom-filter based forwarding. *2013 Conference on Future Internet Communications (CFIC)*, 1–7.
- Ambler, S. W. (1998). *Process patterns*. Cambridge University Press.
- Amini, A., Jamil, N., Ahmad, A. R., & Zaba, M. R. (2015). Threat modeling approaches for securing cloud computing. *Journal of Applied Sciences*, 15(7), 953–967.
- Ammar, Y., & Hany, H. (2003). *Pattern-oriented Analysis and Design: Composing Patterns to Design Software Systems*. Pearson Education Inc.
- Ansar, S. A., & Khan, R. A. (2018). A phase-wise review of software security metrics. In *Networking Communication and Data Knowledge Engineering* (pp. 15–25). Springer.
- Anu, P., & Vimala, S. (2017). A survey on sniffing attacks on computer networks. *2017 International Conference on Intelligent Computing and Control (I2C2)*, 1–5.

- Appleton, B. (2000). *Patterns and Software: Essential Concepts and Terminology*.  
<http://www.bradapp.com/docs/patterns-intro.html>
- Arora, V. (2010). Comparing different information security standards: COBIT vs. *ISO, 27001*.
- Atighetchi, M., Soule, N., Watro, R., & Loyall, J. (2014). *The Concept of Attack Surface Reasoning*.  
<https://doi.org/10.13140/2.1.1491.6484>
- Awan, I. (2014). DEBATING THE TERM CYBER-TERRORISM: ISSUES AND PROBLEMS. *Undefined*. <https://www.semanticscholar.org/paper/DEBATING-THE-TERM-CYBER-TERRORISM%3A-ISSUES-AND-Awan/18066b03bef29209009b55bfd6301f88a0e0f949>
- Aydinli, D. (2015). *Software project management anti-patterns in innovation projects*.
- Aytug, H., Khouja, M., & Vergara, F. E. (2003). Use of genetic algorithms to solve production and operations management problems: A review. *International Journal of Production Research*, *41*(17), 3955–4009.
- Baker, Y. S., Agrawal, R., & Bhattacharya, S. (2013). Analyzing security threats as reported by the united states computer emergency readiness team (US-CERT). *2013 IEEE International Conference on Intelligence and Security Informatics*, 10–12.
- Barnum, S., & Sethi, A. (2007). Attack patterns as a knowledge resource for building secure software. In *OMG Software Assurance Workshop: Cigital*.
- Bauer, B., & Patrick, A. (2014). *A Human Factors Extension to the Seven-Layer OSI Reference Model*. <https://www.andrewpatrick.ca/OSI/10layer.html>
- Beal, V. (2021). *What is Open Systems Interconnection Model? | Webopedia [Blog]*. Open Systems Interconnection Model. <https://www.webopedia.com/definitions/osi/>
- Beck, K. (1987). *Using Pattern Languages for Object-Oriented Programs. OOPSLA-87 workshop on the Specification and Design for Object-Oriented Programming*. Oxford University Press.
- Beck, K., & Coplien, J. O. (1996). Industrial experience with design patterns. *Proceedings of 8th International Conference on Software Engineering*, 103–114.



- Bell, D., & LaPadula, L. (1975). *Secure computer systems: Mathematical foundations and model. Technical Report M74–244*. MITRE Corporation.
- Berczuk, S. P., & Appleton, B. (2003). *Software configuration management patterns: Effective teamwork, practical integration*. Addison-Wesley Professional.
- Bergin, J. (2000). Fourteen pedagogical patterns. *Proceedings of the 5th European Conference on Pattern Languages of Programming*.
- Bhalekar, P. D., & Shaikh, M. Z. (2019). *A NOVEL SURVEY ON DoS ATTACKS*.
- Biskup, J. (2009). Elements of a Security Architecture. In *Security in Computing Systems: Challenges, Approaches and Solutions* (pp. 303–354). Springer Berlin Heidelberg.  
[https://doi.org/10.1007/978-3-540-78442-5\\_10](https://doi.org/10.1007/978-3-540-78442-5_10)
- Bocetta, S. (2019). *Network Security*. <https://www.networkcomputing.com/network-security/problem-complex-networks-getting-harder-secure>
- Bodeau, D. J., & McCollum, C. D. (2018). *System-of-systems threat model*. MITRE CORP MCLEAN VA HOMELAND SECURITY SYSTEMS ENGINEERING AND DEVELOPMENT INSTITUTE.
- Bodeau, D. J., McCollum, C. D., & Fox, D. B. (2018a). *Cyber threat modeling: Survey, assessment, and representative framework* (M. I. T. R. E. C. O. R. P. M. C. L. E. A. N. V. A. MCLEAN, Ed.).
- Bodeau, D. J., McCollum, C. D., & Fox, D. B. (2018b). *Cyber threat modeling: Survey, assessment, and representative framework*. MITRE CORP MCLEAN VA MCLEAN.
- Bodeau, D. J., McCollum, C. D., & Fox, D. B. (2018c). *Cyber threat modeling: Survey, assessment, and representative framework* (M. I. T. R. E. C. O. R. P. M. C. L. E. A. N. V. A. MCLEAN, Ed.).
- Bogdanoski, M. (2013). CYBER TERRORISM– GLOBAL SECURITY THREAT. : :  
*CONTEMPORARY MACEDONIAN DEFENCE - INTERNATIONAL SCIENTIFIC DEFENCE, SECURITY AND PEACE JOURNAL*.

[https://www.academia.edu/11151330/CYBER\\_TERRORISM\\_GLOBAL\\_SECURITY\\_THREAT](https://www.academia.edu/11151330/CYBER_TERRORISM_GLOBAL_SECURITY_THREAT)

- Booth, H., Rike, D., & Witte, G. A. (2013). *The national vulnerability database (nvd): Overview*.
- Borchers, J. O. (1999). Designing interactive musicsystems: A pattern approach. *Proceedings of the 8th International Conference on Human Computer Interaction Ergonomics and User Interfaces*, 276–280.
- Bourgeois, D., & Bourgeois, D. T. (2014). *Chapter 6: Information Systems Security*.  
<https://bus206.pressbooks.com/chapter/chapter-6-information-systems-security/>
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. Wiley Publishing.
- Cain, B. G., & Coplien, J. O. (1996). Social patterns in productive software development organisations. *Annals of Software Engineering*, 2(1), 259–286.
- Campbell, M. (2019). Smart edge: The effects of shifting the center of data gravity out of the cloud. *Computer*, 52(12), 99–102.
- CAPEC. (2021). *CAPEC - CAPEC List Version 3.4*. <https://capec.mitre.org/data/index.html>
- capec.mitre.org. (2021). *CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC)*. <https://capec.mitre.org/>
- Chandola, V., Eilertson, E., Ertoz, L., Simon, G., & Kumar, V. (2006). Data Warehousing and Data Mining Techniques for Computer Security. In *In ch. Data mining for cyber security*. Springer.
- Checkpoint. (2021). *Network Security Architecture*. Check Point Software.  
<https://www.checkpoint.com/cyber-hub/network-security/what-is-network-security/network-security-architecture/>
- Chugh, A. (2020). *Is Kaggle Worth It For Data Scientists? | Built In*. <https://builtin.com/data-science/is-kaggle-worth-it-data-scientists>
- Ciampa, M. (2011). *Security+ Guide to Network Security Fundamentals*. Cengage Learning.

- Ciampa, M. (2017a). *Comptia security+ guide to network security fundamentals*. Cengage Learning.
- Ciampa, M. (2017b). Understanding importance of Information Security. In *Comptia security+ guide to network security fundamentals* (pp. 24–27). Cengage Learning.
- Cio-wiki. (2021). *OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation)*—*CIO Wiki*. [https://cio-wiki.org/wiki/OCTAVE\\_\(Operationally\\_Critical\\_Threat,\\_Asset\\_and\\_Vulnerability\\_Evaluation\)](https://cio-wiki.org/wiki/OCTAVE_(Operationally_Critical_Threat,_Asset_and_Vulnerability_Evaluation))
- Claise, B., Trammell, B., & Aitken, P. (2013). Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information. In *RFC 7011 (Internet Standard)*. Internet Engineering Task Force.
- C.L.A.S.P. (2019). *CLASP Concepts*. [https://www.owasp.org/index.php/CLASP\\_Concepts](https://www.owasp.org/index.php/CLASP_Concepts)
- Cline, M. (1996). The pros and cons of adopting and applying design patterns in the real world. *Communications of the ACM*, 39(10), 47–49.
- Coad, P. (1992). *Object-oriented patterns. Communications of the ACM - Special issue on analysis and modeling in software development* (Vol. 35, Issue 9, pp. 152–159). <https://courses.cs.washington.edu/courses/cse503/04sp/readings/designpattern.pdf>
- Cockcroft, S. (2020). What is the NIST Framework? *ITNOW*, 62(4), 48–49.
- Cole, e, Fossen, j, Northcutt, c, & Pomeranz, h. (2003). *SANS Security Essentials with CISSP CBK*. SANS Press.
- Cole, B. (2020). *What is Risk Management and Why is it Important?* SearchCompliance. <https://searchcompliance.techtarget.com/definition/risk-management>
- Cool, C., & Xie, H. (2000). Patterns of information use, avoidance and evaluation in a corporate engineering environment. *PROCEEDINGS OF THE ANNUAL MEETING-AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 37, 462–472.
- Coplien, J. (1991). *Advanced C++ programming styles and idioms*. Addison-Wesley Longman Publishing Co., Inc.

- Coplien, J. (2000). C++ idioms patterns. *Pattern Languages of Program Design*, 4, 167–197.
- Coplien, J., & Schmidt, D. (1995). *Pattern languages of program design*. ACM Press/Addison-Wesley Publishing Co.
- Corgi. (2020). *Defense in Depth and Layered Network Security | Free Essay Example*. StudyCorgi.Com. <https://studycorgi.com/defense-in-depth-and-layered-network-security/>
- Crutchley, S. (2002). *Information Security: Addressing the human factor*. SC Infosecurity News. [http://www.infosecnews.com/opinion/2002/06/19\\_03.htm](http://www.infosecnews.com/opinion/2002/06/19_03.htm)
- curry. (2013). *Engineering Security Solutions at Layer 8 and Above « Speaking of Security – The RSA Blog and Podcast*. <https://web.archive.org/web/20130524214239/http://blogs.rsa.com/engineering-security-solutions-at-layer-8-and-above/>
- CVSS. (2019). *CVSS v3.1 Specification Document*. FIRST — Forum of Incident Response and Security Teams. <https://www.first.org/cvss/specification-document>
- cyberoam. (nd). *Cyberoam Layer 8 Technology – Cyberoam*. <https://www.cyberoam.com/layer8.html>
- Daş, R., Karabade, A., & Tuna, G. (2015). Common network attack types and defense mechanisms. *2015 23rd Signal Processing and Communications Applications Conference (Siu)*, 2658–2661.
- Dauch, K., Hovak, A., & Nestler, R. (2009). Information assurance using a defense in-depth strategy. *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, 267–272.
- den Braber, F., Hogganvik, I., Lund, M. S., Stølen, K., & Vraalsen, F. (2007). Model-based security analysis in seven steps—A guided tour to the CORAS method. *BT Technology Journal*, 25(1), 101–117. <https://doi.org/10.1007/s10550-007-0013-9>
- Devanabu, P. T., & S, S. (2000). Software engineering for security. *Proceedings of the Conference on The Future of Software Engineering*, 227–239.

- Dhillon, D. (2011). Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security & Privacy*, 9(4), 41–47.
- Disterer, G. (2013). ISO/IEC 27000, 27001 and 27002 for information security management. *Journal of Information Security*, 4(2).
- Dolbeau, A. (2022, April 28). *Threat modeling: Which method should you choose for your company? (Stride, Dread, QTMM, LINDDUN, PASTA)*. Positive Thinking Company.  
<https://positivethinking.tech/insights/threat-modeling-which-method-should-you-choose-for-your-company-stride-dread-qtmm-linddun-pasta/>
- Donaldson, S. E., Siegel, S. G., Williams, C. K., & Aslam, A. (2015). Cybersecurity frameworks. In *Enterprise Cybersecurity* (pp. 297–309).
- Dougherty, C., Sayre, I., seacord, R., Svoboda, D., & Togashi, K. (2009). *Secure Design Patterns*. Carnegie Mellon University.
- Douglass, B. P. (2002). *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. Addison-Wesley Longman Publishing Co., Inc.
- Eakin, E. (2003). *Architecture's irascible reformer*. *The New York Times*.
- Elejla, O. E., Anbar, M., & Belaton, B. (2017). ICMPv6-based DoS and DDoS attacks and defense mechanisms. *IETE Technical Review*, 34(4), 390–407.
- Ellison, R. J., & Woody, C. (2010). Supply-chain risk management: Incorporating security into software development. *2010 43rd Hawaii International Conference on System Sciences*, 1–10.
- Engbretson, P. H., & Pauli, J. J. (2009). Leveraging Parent Mitigations and Threats for CAPEC-Driven Hierarchies. *2009 Sixth International Conference on Information Technology: New Generations*, 344–349. <https://doi.org/10.1109/ITNG.2009.24>
- Engbretson, P. H., Pauli, J. J., & Streff, K. (2008). Abstracting Parent Mitigations from the CAPEC Attack Pattern Dictionary. In *Security and Management* (pp. 245–250).

- Eric. (2016, August 23). Importance of OSI Model WALT Labs WALT Labs. *WALT Labs*.  
<https://waultlabs.io/osi-model-security/>
- Eskierka, J. A. (2011). *Proposing a succession planning and leadership development program for the St. Paul fire department* (Doctoral dissertation, The College of St. Scholastica).
- Fernandez, E. (2009). Security Patterns and A Methodology to Apply them. *Advances in Information Security*, 45.
- Fernández, E., Jurjens, J., Vanhilst, M., & Pernul, G. (2010). Using Security Patterns to Develop Secure Systems. *Software Engineering for Secure Systems: Industrial and Research Perspectives*. <https://doi.org/10.4018/978-1-61520-837-1.ch002>
- Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3(Mar), 1289–1305.
- Fricke, A., & Volter, M. (2000). A pedagogical pattern language about teaching seminars effectively. *Proceedings of the 5th European Conference on Pattern Languages of Programs*.
- Galloway, L.-A. (2019). *Application Security*. <https://www.darkreading.com/application-security/a-secure-development-approach-pays-off/a/d-id/1331154>
- Gamma, e, Helm, R., Johnson, R., & J, V. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. *ECOOP'93—Object-Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings* 7, 406–431.
- Ganssle, J. (1992). *The art of programming embedded systems*. Academia Press.
- Gardner, K. M., & Rush, A. (1998). *Cognitive patterns*. Cambridge University Press.
- Gegick, M., & Williams, L. (2005a). Matching attack patterns to security vulnerabilities in softwareintensive designs. *ACM SIGSOFT Software Engineering Notes*, 30, 1–7.  
<https://doi.org/10.1145/1082983.1083211>

- Gegick, M., & Williams, L. (2005b). Matching attack patterns to security vulnerabilities in software-intensive system designs. *Proceedings of the 2005 Workshop on Software Engineering for Secure Systems—Building Trustworthy Applications*, 1–7.
- Goldberg, S. (2017, June 22). *Surveillance without Borders: The “Traffic Shaping” Loophole and Why It Matters*. The Century Foundation. <https://tcf.org/content/report/surveillance-without-borders-the-traffic-shaping-loophole-and-why-it-matters/>
- Gonzalez, C. (2022, February 7). *8 Threat Modeling Methodologies: Prioritize & Mitigate Threats*. Exabeam. <https://www.exabeam.com/information-security/threat-modeling/>
- Graft, D., Pabrai, M., & Pabrai, U. (1990). Methodology for Network Security Design. In *IEEE Communications Magazine—IEEE Commun. Mag.* (Vol. 28, p. 682). <https://doi.org/10.1109/PCCC.1990.101685>
- Greg, M. (2019). *OSI: Securing the Stack, Layer 8 -- Social engineering and security policy*. SearchNetworking. <https://searchnetworking.techtarget.com/tip/OSI-Securing-the-Stack-Layer-8-Social-engineering-and-security-policy>
- Gregg, M., Bandes, R., Franklin, B., Mays, G., Ries, C., & Watkins, S. (2006). *Hack the Stack: Using Snort and Ethereal to Master the 8 Layers of an Insecure Network*. Syngress Publishing.
- Gregoire, J., Buyens, K., De Win, B., Scandariato, R., & Joosen, W. (2007). On the Secure Software Development Process CLASP and SDL Compared. *Proc. of the 3rd International Workshop on Software Engineering for Secure Systems*, 1.
- Grenander, E. P. D. A. M. U., Grenander, U., Miller, U. G. M., Miller, M. I., & Miller, M. (2007). *Pattern Theory: From Representation to Inference*. OUP Oxford. <https://books.google.co.ke/books?id=0MsTDAAAQBAJ>
- Groat, S., Tront, J., & Marchany, R. (2012). Advancing the defense in depth model. *2012 7th International Conference on System of Systems Engineering (SoSE)*, 285–290.

- Hamid, B. (2014). Modeling of Secure and Dependable Applications Based on a Repository of Patterns: The SEMCO Approach. *Reliability Digest, Special*, 9–17.
- Hanmer, R. (2007). *Patterns for fault-tolerant software*. John Wiley & Sons, Ltd.
- Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10), 993–1001.
- Harrison, N. (2006). The Language of Shepherding, A Pattern Language for Shepherds and Sheep. In A. D. Manolescu, M. Voelter, & J. Noble (Eds.), *Pattern Languages of Program Design 5* (Vol. 5). Addison-Wesley.
- Hassan, A. B., Lass, F. D., & Makinde, J. (2012). Cybercrime in Nigeria: Causes, Effects and the Way Out. *Undefined*. <https://www.semanticscholar.org/paper/Cybercrime-in-Nigeria%3A-Causes%2C-Effects-and-the-Way-Hassan-Lass/59a569f479acd78372c7f58086bde498f526dbe7>
- Hastings, N. E., & McLean, P. A. (1996). TCP/IP spoofing fundamentals. *Conference Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, 218–224.
- Hewitt, K. (2020). *Importance of Network Security Risk Management | SecurityScorecard*. <https://securityscorecard.com/blog/importance-of-network-security-risk-management>
- Hnamte, V., & Hussain, J. (2021). An Extensive Survey on Intrusion Detection Systems: Datasets and Challenges for Modern Scenario. *2021 3rd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*, 1–10.
- Hoglund, G., & McGraw, G. (2005). *Exploiting Software: How to Break Code* (Vol. 1). Addison-Wesley.
- Holl, K. (2003). OSI defense in depth to increase application security. In *SANS Security Essentials GSEC Practical Assignment Version* (p. 1).
- Horkoff, J., & Yu, E. (2016). Interactive goal model analysis for early requirements engineering. *Requirements Engineering*, 21(1), 29–61.



- Howard, M., & LeBlanc, D. (2003). *Writing secure code*. Pearson Education.
- Howard, M., & Lipner, S. (2006). *The security development lifecycle* (Vol. 8). Microsoft Press.
- Hunt, C. (2002). *TCP/IP network administration* (Vol. 2). O'Reilly Media, Inc."
- Hussain, S., Kamal, A., Ahmad, S., Rasool, G., & Iqbal, S. (2014). Threat modelling methodologies: A survey. *Sci. Int.(Lahore)*, 26(4), 1607–1609.
- Iguer, H., Medromi, H., Sayouti, A., & TALLAL, S. (2013). A new architecture multi-agents based combining EBIOS and ISO 27001 in IT risk management. *Proc. ICEER*, 13.
- Infosec Insights. (2020, December 31). Hacker Motivation: Why Do Hackers Hack? *InfoSec Insights*. <https://sectigostore.com/blog/hacker-motivation-why-do-hackers-hack/>
- Iqbal, M. (2004). Defining Cyberterrorism, 22 J. Marshall J. Computer & Info. L. 397 (2004). *UIC John Marshall Journal of Information Technology & Privacy Law*, 22(2).  
<https://repository.law.uic.edu/jitpl/vol22/iss2/2>
- ISO/IEC 27005:2011. (2018). *ISO/IEC 27005:2011(en), Information technology—Security techniques—Information security risk management*.  
<https://www.iso.org/obp/ui/es/#iso:std:iso-iec:27005:ed-2:v1:en>
- ITU. (2014). *Understanding cybercrime: Phenomena, challenges and legal response*. ITU.  
<https://www.itu.int:443/en/publications/ITU-D/Pages/publications.aspx>
- Izurieta, C., & Bieman, J. M. (2013). A multiple case study of design pattern decay, grime, and rot in evolving software systems. *Software Quality Journal*, 21, 289–323.
- Jasud, P. V. (2017). The OSI Model: Overview on the Seven Layers of Computer Networks. *International Journal for Innovative Research in Science & Technology*, 4(3), 116–124.
- Jézéquel, J. M., Train, M., & Mingins, C. (2000). *Design patterns and contracts*. Addison-Wesley Professional.
- Jufri, M. T., Hendayun, M., & Suharto, T. (2017). Risk-assessment based academic information System security policy using octave Allegro and ISO 27002. *2017 Second International Conference on Informatics and Computing (ICIC)*, 1–6.

- Jureta, I. J., Borgida, A., Ernst, N. A., & Mylopoulos, J. (2010). Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling. *2010 18th IEEE International Requirements Engineering Conference*, 115–124.  
<https://doi.org/10.1109/RE.2010.24>
- Jürjens, J. (2005). *Secure Systems Development with UML*. Springer.
- Kaiya, H., Kono, S., Ogata, S., Okubo, T., Yoshioka, N., Washizaki, H., & Kaijiri, K. (2014). Security Requirements Analysis Using Knowledge in CAPEC. In L. Iliadis, M. Papazoglou, & K. Pohl (Eds.), *Advanced Information Systems Engineering Workshops* (Vol. 178, pp. 343–348). Springer International Publishing. [https://doi.org/10.1007/978-3-319-07869-4\\_32](https://doi.org/10.1007/978-3-319-07869-4_32)
- Kamiri, J., & Mariga, G. (2021). Research Methods in Machine Learning: A Content Analysis. *International Journal of Computer and Information Technology* (2279-0764, 10(2)).
- Kanakogi, K., Washizaki, H., Fukazawa, Y., Ogata, S., Okubo, T., Kato, T., Kanuka, H., Hazeyama, A., & Yoshioka, N. (2021). *Tracing CAPEC Attack Patterns from CVE Vulnerability Information using Natural Language Processing Technique* (p. 6996).  
<https://doi.org/10.24251/HICSS.2021.841>
- Karahasanovic, A., Kleberger, P., & Almgren, M. (2017). Adapting threat modeling methods for the automotive industry. *Proceedings of the 15th ESCAR Conference*, 1–10.
- Kaspersky. (2020). *The Human Factor in IT Security: How Employees are Making Businesses Vulnerable from Within*. <https://www.kaspersky.com/blog/the-human-factor-in-it-security/>
- Kaur, D., & Singh, P. (2014). Various OSI layer attacks and countermeasure to enhance the performance of WSNs during wormhole attack. *International Journal on Network Security*, 5(1), 62.
- Kaur, H., & Kumari, V. (2020). *Predictive modelling and analytics for diabetes using a machine learning approach*. *Applied computing and informatics*.
- Kavianpour, A., & Anderson, M. C. (2017). An overview of wireless network security. *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 306–309.

- KDD Cup 1999 Data*. (1999). <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- Keerthi, V. K. (2016). Taxonomy of SSL/TLS attacks. *International Journal of Computer Network and Information Security*, 8(2), 15.
- Kevin, D. (2017). *The 5 Biggest Data Breaches of 2017*. <https://www.observeit.com/blog/biggest-data-breaches-2017/>
- Khan, R., McLaughlin, K., Lavery, D., & Sezer, S. (2017a). STRIDE-based threat modeling for cyber-physical systems. *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 1–6.
- Khan, R., McLaughlin, K., Lavery, D., & Sezer, S. (2017b). STRIDE-based threat modeling for cyber-physical systems. *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 1–6.
- Khosrowshahi, D. (2017). *2016 Data Security Incident*. <https://www.uber.com/newsroom/2016-data-incident>
- Kizza, J. M., Kizza, W., & Wheeler. (2013). *Guide to computer network security*. Springer.
- Klingmann, A. (2010). *Brandscapes: Architecture in the experience economy*. Mit Press.
- Koch, R., Golling, M., & Rodosek, G. D. (2014). Towards comparability of intrusion detection systems: New data sets. *TERENA Networking Conference*, 7.
- Kocher, P., Lee, R., B., M., G., & Raghunath, A. (2004). *Security as a New Dimension in Embedded System Design*. <https://doi.org/10.1145/996566.996771>
- Kohnfelder, L., & Garg, P. (1999). *The threats to our products*. Microsoft Interface, Microsoft Corporation.
- Köksal, Ö., & Tekinerdogan, B. (2019). Architecture design approach for IoT-based farm management information systems. *Precision Agriculture*, 20(5), 926–958.
- Koltuksuz, A. (2013). Cover Use of Cyberspace and Technology by Terrorists. In *Technological Dimensions of Defence against Terrorism* (Vol. 115, pp. 106–109).

- Konev, A., Shelupanov, A., Kataev, M., Ageeva, V., & Nabieva, A. (2022). A Survey on Threat-Modeling Techniques: Protected Objects and Classification of Threats. *Symmetry*, *14*(3).  
<https://doi.org/10.3390/sym14030549>
- Korba, A. A., Nafaa, M., & Salim, G. (2013). Survey of routing attacks and countermeasures in mobile ad hoc networks. *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, 693–698.
- Kostova, B., Gürses, S., & Troncoso, C. (2020). *Privacy engineering meets software engineering. On the challenges of engineering privacy by design.*
- Kumar, A., Sharma, A. K., & Singh, A. (2012). Performance Evaluation of DVMRP Multicasting Network over ICMP Ping Flood for DDoS. *Performance Evaluation*, *4*(6).
- Kumar, G. (2016). Denial of service attacks—an updated perspective. *Systems Science & Control Engineering*, *4*(1), 285–294.
- Kumar, S., Dalal, S., & Dixit, V. (2014). The OSI model: Overview on the seven layers of computer networks. *International Journal of Computer Science and Information Technology Research*, *2*(3), 461–466.
- Kumar, V., & Minz, S. (2014). Feature selection: A literature review. *SmartCR*, *4*(3), 211–229.
- Lakhani, F., & Faisal, N. (2015). Design patterns—from architecture to embedded software development. *International Journal of Computer Science Issues (IJCSI)*, *12*(1), 146.
- Lamarca, B. I. (2020). Cybersecurity Risk Assessment of the University of Northern Philippines using PRISM Approach. *IOP Conference Series. Materials Science and Engineering*, *769*(1).
- Lamsweerde, A. (2004a). Elaborating Security Requirements by Construction of Intentional Anti-Models. *Proceedings of the 26th International Conference on Software Engineering*, 148–157.

- Lamsweerde, A. (2004b). Elaborating security requirements by construction of intentional anti-models. In *Proceedings—International Conference on Software Engineering* (Vol. 26, p. 157). <https://doi.org/10.1109/ICSE.2004.1317437>
- LeBlanc, D. (2007). *DREADful*. <http://blogs>.
- Leveson, N. G. (1995). *Safeware: System safety and computers*. ACM.
- Li, T., & Horkoff, J. (2014). *Dealing with Security Requirements for Socio-Technical Systems: A Holistic Approach*. <https://doi.org/10.13140/2.1.1708.1604>
- Li, T., Horkoff, J., Beckers, K., Paja, E., & Mylopoulos, J. (2015). A Holistic Approach to Attack Modeling and Analysis. *iStar*.
- Li, T., Horkoff, J., Paja, E., Beckers, K., & Mylopoulos, J. (2015). Analyzing Attack Strategies Through Anti-goal Refinement. In J. Ralyté, S. España, & Ó. Pastor (Eds.), *The Practice of Enterprise Modeling* (pp. 75–90). Springer International Publishing. [https://doi.org/10.1007/978-3-319-25897-3\\_6](https://doi.org/10.1007/978-3-319-25897-3_6)
- Li, T., Paja, E., Mylopoulos, J., Horkoff, J., & Beckers, K. (2015). *Holistic Security Requirements Analysis: An Attacker's Perspective*. <https://doi.org/10.1109/RE.2015.7320439>
- Li, X., Li, K., Qiao, D., Ding, Y., & Wei, D. (2019). Application research of machine learning method based on distributed cluster in information retrieval. *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, 411–414.
- Li, Y., Li, D., Cui, W., & Zhang, R. (2011). Research based on OSI model. *2011 IEEE 3rd International Conference on Communication Software and Networks*, 554–557.
- Liebowitz, M. (2011). *2011 set to be worst year ever for security breaches*. <http://www.securitynewsdaily.com/756-2011-worst-year-ever-security->
- linddun. (2020). *LINDDUN*. LINDDUN. <https://www.linddun.org/linddun>
- Lund, M. S., Solhaug, B., & Stølen, K. (2010). *Model-driven risk analysis: The CORAS approach*. Springer Science & Business Media.

- MacKinnon, L., Bacon, L., Gan, D., Loukas, G., Chadwick, D., & Frangiskatos, D. (2013). Cyber security countermeasures to combat cyber terrorism. In B. Akhgar & S. Yates (Eds.), *Strategic Intelligence Management: National Security Imperatives and Information and Communications Technologies* (pp. 234–256). Butterworth-Heinemann.  
<http://gala.gre.ac.uk/id/eprint/9957/>
- Madory, D. (2018, July 12). Shutting down the BGP Hijack Factory. *APNIC Blog*.  
<https://blog.apnic.net/2018/07/12/shutting-down-the-bgp-hijack-factory/>
- Maghrabi, L., Pfluegel, E., & Noorji, S. F. (2016). Designing utility functions for game-theoretic cloud security assessment: A case for using the common vulnerability scoring system. *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*, 1–6.
- Maheshwari, V., & Prasanna, M. (2016). Integrating risk assessment and threat modeling within SDLC process. *2016 International Conference on Inventive Computation Technologies (ICICT, 1)*, 1–5.
- Mahmood, S., Mohsin, S. M., & Akber, S. M. A. (2020). Network security issues of data link layer: An overview. *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 1–6.
- Manavi, M. T. (2018). Defense mechanisms against distributed denial of service attacks: A survey. *Computers & Electrical Engineering*, 72, 26–38.
- Mang, P., & Reed, B. (2012). Designing from place: A regenerative framework and methodology. *Building Research & Information*, 40(1), 23–38.
- Manninen, O. (2018). *Cybersecurity in agricultural communication networks: Case dairy farms*.
- Manns, M. L., & Rising, L. (2000). *Fearless change: Patterns for introducing new ideas*. Addison Wesley.
- Marback, A., Do, H., He, K., Kondamarri, S., & Xu, D. (2013a). A threat model-based approach to security testing. *Software: Practice and Experience*, 43(2), 241–258.

- Marback, A., Do, H., He, K., Kondamarri, S., & Xu, D. (2013b). A threat model-based approach to security testing. *Software: Practice and Experience*, 43(2), 241–258.
- Martin, A. (2006). *The Common Attack Pattern Enumeration and Classification (CAPEC) Initiative*.
- Martin, Y. S., & Kung, A. (2018). Methods and tools for GDPR compliance through privacy and data protection engineering. *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 108–111.
- Martinović, M., Lovaković, D., & Ćosić, T. (2014). Network Security Issues in Regard to OSI Reference Model Layers. *6th International Scientific and Expert Conference TEAM2014*.
- Mayer, N. (2009). *Modelbased Management of Information System Security Risk*.
- Mayer, N., Aubert, J., Grandry, E., Feltus, C., Goettelmann, E., & Wieringa, R. (2018). An integrated conceptual model for information system security risk management supported by enterprise architecture management. *Software & Systems Modeling*.  
<https://doi.org/10.1007/s10270-018-0661-x>
- Mayr, A., Binder, H., Gefeller, O., & Schmid, M. (2014). The evolution of boosting algorithms. *Methods of Information in Medicine*, 53(06), 419–427.
- McGraw, G., & Hoglund, G. (2004). Exploiting software: How to break code. *Invited Talk, Usenix Security Symposium, San Diego*, 9–13.
- McGraw, G., & Young, L. (2016). *Build Security In Maturity Model (BSIMM) – Practices from Seventy Eight Organizations*. <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=450642>
- Mead, N. R., Shull, F., Vemuru, K., & Villadsen, O. (2018a). *A hybrid threat modeling method*. Carnegie Mellon University-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002.
- Mead, N. R., Shull, F., Vemuru, K., & Villadsen, O. (2018b). *A hybrid threat modeling method*. Carnegie Mellon University-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002.

- Mell, P., Scarfone, K., & Romanosky, S. (2007). A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-forum of incident response and security teams* (Vol. 1, p. 23).
- Microsoft. (2019). *Microsoft Security Development Life Cycle*. <https://www.microsoft.com/en-us/securityengineering/sdl/practices>
- Mihailescu, V. L. (2012). Risk analysis and risk management using MEHARI. *J. Appl. Bus. Inf. Syst.*, 3(4), 143–162.
- Miller, C. (2013). *Security Considerations in Managing COTS Software | CISA*. <https://www.cisa.gov/uscert/bsi/articles/best-practices/legacy-systems/security-considerations-in-managing-cots-software>
- Millett, S. (2010). *Professional ASP.Net Design Patterns*. Wiley Publishing Inc.
- Möckel, C., & Abdallah, A. E. (2010). Threat modeling approaches and tools for securing architectural designs of an e-banking application. *2010 Sixth International Conference on Information Assurance and Security*, 149–154.
- Mohanakrishnan. (2021). *Top 10 Threat Modeling Tools in 2021 | Spiceworks It-security*. <https://www.spiceworks.com/it-security/vulnerability-management/articles/top-threat-modeling-tools/>
- Mouratidis, H., & Giorgini, P. (2007). Secure Tropos: A Security-Oriented Extension of the Tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2), 285–309.
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference (MilCIS)*, 1–6.
- Nagpal, R. (2002). *Cyber terrorism in the context of globalization*. II World Congress on Informatics and Law.



- Najafabadi, M. M., Khoshgoftaar, T. M., Kemp, C., Seliya, N., & Zuech, R. (2014). Machine learning for detecting brute force attacks at the network level. *2014 IEEE International Conference on Bioinformatics and Bioengineering*, 379–385.
- Neil, T. (2012). *Mobile Design Pattern Gallery*. O'Reilly Media Inc.
- Neumann, P. (2004). *Principled Assuredly Trustworthy Composable Architectures (Final Report to DARPA, CDRL A001)*. SRI International.
- Niekerk, J., V., Solms, R., & V. (2006). Understanding Information Security Culture: A Conceptual Framework. *Conference: Proceedings of the ISSA 2006 from Insight to Foresight*. Sandton.  
[https://www.researchgate.net/publication/220803272\\_Understanding\\_Information\\_Security\\_Culture\\_A\\_Conceptual\\_Framework](https://www.researchgate.net/publication/220803272_Understanding_Information_Security_Culture_A_Conceptual_Framework)
- Nishant, S. (2012). *Security Assessment via Penetration Testing: Network and System Administrator's Approach: Security, Network and System Administrator, Penetration Testing* [(Masters Thesis).]. <https://www.duo.uio.no/handle/10852/34904>
- Noble, J. (1998). Towards a pattern language for object oriented design. *Proceedings Technology of Object-Oriented Languages*, 2–13.
- Norman, J., & Joseph, P. (2017). Security in Application Layer Protocols of IoT: Threats and Attacks. In *Security Breaches and Threat Prevention in the Internet of Things* (pp. 76–95). IGI Global.
- Nweke, L. O., & Wolthusen, S. (2020). *A review of asset-centric threat modelling approaches*.
- Olagunju, R. E., Aremu, R. C., & Ogundele, J. (2013). Incessant Collapse Of Buildings In Nigeria: An Architect's View. *IISTE Journal of Civil and Environmental Research*, 3(4), 49–54.
- Osakwe, M. (2020, September 17). *4 Reasons Why the OSI Model Still Matters*. Nightfall AI.  
<https://nightfall.ai/4-reasons-why-the-osi-model-still-matters>
- Pace, K. (2014). A Layered Security Model: OSI and Information Security. *SANS Institute*.  
<https://www.giac.org/paper/gsec/3908/layered-security-model-osi-information-security/106272>

- Pahwa, K., & Agarwal, N. (2019). Stock market analysis using supervised machine learning. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 197–200.
- Parmar, H., & Gosai, A. (2015). Analysis and study of network security at transport layer. *International Journal of Computer Applications*, *121*(13), 35–40.
- Pauli, J. J., & Engebretson, P. H. (2008). Hierarchy-driven approach for attack patterns in software security education. *Fifth International Conference on Information Technology: New Generations (Itng 2008)*, 1156–1157.
- Pearson. (2016). *7 Popular Layer 2 Attacks | 7 Popular Layer 2 Attacks | Pearson IT Certification*.  
<https://www.pearsonitcertification.com/articles/article.aspx?p=2491767>
- Petter, S., Khazanchi, D., & Murphy, J. D. (2010). A design science based evaluation framework for patterns. *ACM SIGMIS Database: The DATABASE for Advances in Information Systems*, *41*(3), 9–26.
- Poston, H. (n.d.). *Role and purpose of threat modeling in software development—Infosec Resources*. Retrieved August 17, 2022, from <https://resources.infosecinstitute.com/topic/role-and-purpose-of-threat-modeling-in-software-development/>
- Potteiger, B., Martins, G., & Koutsoukos, X. (2016). Software and attack centric integrated threat modeling for quantitative risk assessment. *Proceedings of the Symposium and Bootcamp on the Science of Security*, 99–108.
- Prakash, A. (2020). *Cloud IaaS Threat Modelling*. <https://www.linkedin.com/pulse/cloud-iaas-threat-modelling-atul-prakash>
- Prechelt, L. B., & Unger, L. (2001). A controlled experiment in maintenance, comparing design patterns to simpler solutions. *IEEE Transactions on Software Engineering*, *27*(12), 1134–1144.

- Progoulakis, I., Rohmeyer, P., & Nikitakos, N. (2021). Cyber Physical Systems Security for Maritime Assets. *Journal of Marine Science and Engineering*, 9(12), Article 12. <https://doi.org/10.3390/jmse9121384>
- przybyla, M. (2020). *What are the possible/potential benefits of solving/completing Kaggle competitions for a prospective graduate student?* Quora. <https://www.quora.com/What-are-the-possible-potential-benefits-of-solving-completing-Kaggle-competitions-for-a-prospective-graduate-student>
- Reed, D. (2003). *Applying the OSI Seven Layer Network Model To Information Security*. SANS Institute Reading Room. <https://www.sans.org/reading-room/whitepapers/protocols/applying-osi-layer-network-model-information-security-1309>
- Rehman, S., & Mustafa, K. (2013). SOFTWARE SECURITY RISK MITIGATION USING OBJECT ORIENTED DESIGN PATTERNS. *International Journal of Research in Engineering and Technology*, 2(7).
- Repository, P. P. (2014). <http://wiki.c2.com/?PortlandPatternRepository>
- Reuters. (2017). <http://fortune.com/2017/10/10/deloitte-clients-hacking/>
- Riaz, R., Rizvi, S. S., Riaz, F., Hameed, N., & Shokat, S. (2007). Analysis of Web based Structural Security Patterns by. *International Journal of Computer Science and Network Security*, 7(10).
- Richardson, S. (2022, June 3). *Modularizing Security Design—Network Design*. Cisco Certified Expert. <https://www.ccexpert.us/network-design-2/modularizing-security-design.html>
- Ring, M., Landes, D., & Hotho, A. (2018). Detection of slow port scans in flow-based network traffic. *PloS One*, 13(9), 0204507.
- Ring, M., Wunderlich, S., Grüdl, D., Landes, D., & Hotho, A. (2017). A toolset for intrusion and insider threat detection. In *Data Analytics and Decision Support for Cybersecurity* (pp. 3–31). Springer.

- Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A Survey of Network-based Intrusion Detection Data Sets. *Comput. Secur.*  
<https://doi.org/10.1016/j.cose.2019.06.005>
- Rising, L. (1998). *The Pattern Handbook: Techniques, Strategies and Application* (L. Rising, Ed.). Cambridge University Press.
- Ruggiero, P., & Foote, J. (2011). [https://www.us-cert.gov/sites/default/files/publications/cyber\\_threats-to\\_mobile\\_phones.pdf](https://www.us-cert.gov/sites/default/files/publications/cyber_threats-to_mobile_phones.pdf)
- Rushby, J. (2001). *Security Requirements Specifications: How and What ? Extended*.  
<https://www.semanticscholar.org/paper/Security-Requirements-Specifications-%3A-How-and-What-Rushby/3f59a1dec34ab5f92933d1878c61e64722bac7eb>
- Saini, H., & Rao, Y. S. (2012). *Cyber-Crimes and their Impacts: A Review 1*.
- Saitta, P., Larcom, B., & Eddington, M. (2005). *Trike v*. <http://dymaxion>.
- Salerno, s, Sanzgiri, A., & Upadhyaya, S. (2011). Exploration of Attacks on Current Generation Smartphones. *Procedia Computer Science*, 5, 546–553.  
<https://doi.org/10.1016/j.procs.2011.07.071>.
- Santos, V., Goldman, A., & De Souza, C. R. (2015). Fostering effective inter-team knowledge sharing in agile software development. *Empirical Software Engineering*, 20, 1006–1051.
- Sapronov, K. (2015). *The human factor and information security*. <https://securelist.com/the-human-factor-and-information-security/36067/>
- Sarkar, D., Bali, R., & Sharma, T. (2018). Machine Learning Basics. In D. Sarkar, R. Bali, & T. Sharma (Eds.), *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems* (pp. 3–65). Apress. [https://doi.org/10.1007/978-1-4842-3207-1\\_1](https://doi.org/10.1007/978-1-4842-3207-1_1)
- Sas, A. (2019). *Number of cyber security incidents handled by CERT*.  
<https://www.statista.com/statistics/1028557/poland-cybersecurity-incidents/>

- Scandariato, R., Wuyts, K., & Joosen, W. (2015). A descriptive study of Microsoft's threat modeling technique. *Requirements Engineering*, 20(2), 163–180.
- Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2), 197–227.
- Schmidt, D. (1995). Using design patterns to develop reusable object-oriented communication software. *Communication ACM*, 38(10), 65–74.
- Schmidt, D. C., Stal, M., Rohnert, H., & Buschmann, F. (2013). *Pattern-oriented software architecture, patterns for concurrent and networked objects* (Vol. 2). John Wiley & Sons.
- Schneier, B. (1999). Attack trees. *Dr. Dobbs's Journal*, 24(12), 21–29.
- Schumacher, M. (2003a). *Security Engineering with Patterns* (G. Goos, J. Hartmanis, & J. Leeuwen, Eds.). Springer-Verlag.
- Schumacher, M. (2003b). *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. Springer-Verlag.
- Schumacher, M. (2003c). *Security engineering with patterns: Origins, theoretical models, and new applications* (Vol. 2754). Springer Science & Business Media.
- Schumacher, M., E.D, B., Hybertson, B., F, S., & P. (2006). Security Patterns, Integrating Security and Systems Engineering. In *Security Patterns, Integrating Security and Systems Engineering*. John Wiley & Sons Ltd.
- Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2013a). *Security Patterns: Integrating security and systems engineering*. John Wiley & Sons.
- Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2013b). *Security Patterns: Integrating security and systems engineering*. John Wiley & Sons.
- Seehusen, F. (2015). *Using CAPEC for Risk-Based Security Testing* (p. 92).  
[https://doi.org/10.1007/978-3-319-26416-5\\_6](https://doi.org/10.1007/978-3-319-26416-5_6)
- Selin, J. (2019). *Evaluation of threat modeling methodologies*.

- Sena, J., & Geus, P. (2002). A Protection Model for Network Communications Based on Security Levels. *International Conference on Security and Management (SAM'02)*. International Conference on Security and Management, Las Vegas, NV, USA.
- Shah, N. (2018, May 1). *The Security Risks Presented by Complex Networks*. Fortinet Blog. <https://www.fortinet.com/blog/business-and-technology/decrease-your-customers--network-complexity-for-increased-securi.html>
- Shakiba-Herfeh, M., Chorti, A., & Poor, H. V. (2021). Physical layer security: Authentication, integrity, and confidentiality. In *Physical Layer Security* (pp. 129–150). Springer.
- Sharafaldin, I., Gharib, A., Lashkari, A. H., & Ghorbani, A. A. (2018). Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2018(1), 177–200.
- Shaw, K. (2022, March 14). *The OSI model explained and how to easily remember its 7 layers*. Network World. <https://www.networkworld.com/article/3239677/the-osi-model-explained-and-how-to-easily-remember-its-7-layers.html>
- Shevchenko, N. (2018). *Threat Modeling: 12 Available Methods*. [https://insights.sei.cmu.edu/sei\\_blog/2018/12/threat-modeling-12-available-methods.html](https://insights.sei.cmu.edu/sei_blog/2018/12/threat-modeling-12-available-methods.html)
- Shevchenko, N., Chick, T. A., O’Riordan, P., Scanlon, T. P., & Woody, C. (2018a). *Threat modeling: A summary of available methods*. Carnegie Mellon University Software Engineering Institute Pittsburgh United States.
- Shevchenko, N., Chick, T. A., O’Riordan, P., Scanlon, T. P., & Woody, C. (2018b). *Threat modeling: A summary of available methods*. Carnegie Mellon University Software Engineering Institute Pittsburgh United States.
- Shevchenko, N., Frye, B. R., & Woody, C. (2018a). *Threat modeling for cyber-physical system-of-systems: Methods evaluation*. Carnegie Mellon University Software Engineering Institute Pittsburgh United States.

- Shevchenko, N., Frye, B. R., & Woody, C. (2018b). *Threat modeling for cyber-physical system-of-systems: Methods evaluation*. Carnegie Mellon University Software Engineering Institute Pittsburgh United States.
- Shi, L., Li, Y., Loo, B. T., & Alur, R. (2021). Network Traffic Classification by Program Synthesis. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 430–448.
- Shostack, A. (2014a). *Threat modeling: Designing for security*. John Wiley & Sons.
- Shostack, A. (2014b). *Threat Modeling: Designing for Security* (1st edition). Wiley.
- Shull, F. (2016). *Evaluation of threat modeling methodologies*. Software Engineering Institute, Carnegie Mellon University. <https://securityintelligence.com/threat-modeling-in-the-enterprise-part-2-understanding-the-process/>
- Sietz, D. (2011). *Dryland vulnerability: Typical patterns and dynamics in support of vulnerability reduction efforts* [(Doctoral dissertation,)]. Universitätsbibliothek der Universität Potsdam.
- Simeonova, S. (2016). *Threat Modeling in the Enterprise, Part 2: Understanding the Process*. Security Intelligence. <https://securityintelligence.com/threat-modeling-in-the-enterprise-part-2-understanding-the-process/>
- Small, J. (2012). *PATTERNS IN NETWORK SECURITY: AN ANALYSIS PATTERNS IN NETWORK SECURITY RECURSIVE INTER-NETWORK ARCHITECTURE NETWORKS*( *Doctrol Thesis*. B.S., University of Massachusetts.
- Smith, E. (2008). *Using secondary data in educational and social research*. McGraw-Hill Education (UK.
- Smith, J. M. (2012). *Elemental design patterns*. Addison-Wesley.
- Smys, S., Basar, A., & Wang, H. (2020). Hybrid intrusion detection system for internet of things (IoT). *Journal of ISMAC*, 2(04), 190–199.
- Solomon, S. (2016, February 4). *Application Layer Security Within the OSI Model*. Checkmarx. <https://www.checkmarx.com/blog/application-layer-security-within-osi-model/>

- Stanganelli, J. (2016a). *Selecting a Threat Risk Model for your Organization, Part Two. eSecurity Planet.*
- Stanganelli, J. (2016b). *Selecting a Threat Risk Model for your Organization, Part Two. eSecurity Planet.*
- Stewart, R., & Metz, C. (2001). Sctp: new transport protocol for TCP/IP. *IEEE Internet Computing*, 5(6), 64–69.
- Stojkovic, V., & Steele, G. (2005). Modeling, Simulation, and Visualization of a Network Using the Easel Programming Language. *Proceedings of the 9th Colloquium for Information Systems Security Education*, 6, 9.
- Stoneburner, G., Goguen, A., & Feringa, A. (2002). Risk management guide for information technology systems. *Nist Special Publication*, 800(30), 800–830.
- Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018). Mitre att&ck: Design and philosophy. In *Technical report*. The MITRE Corporation.
- Surman, G. (2002). Understanding security using the OSI model. *SANS Institute InfoSec Reading Room.*
- Swire, P. (2018). A pedagogic cybersecurity framework. *Communications of the ACM*, 61(10), 23–26.
- Syarif, I., Zaluska, E., Prugel-Bennett, A., & Wills, G. (2012). Application of bagging, boosting and stacking to intrusion detection. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (pp. 593–602). Springer.
- Szabo, R., Kind, M., Westphal, F. J., Woesner, H., Jocha, D., & Csaszar, A. (2015). Elastic network functions: Opportunities and challenges. *IEEE Network*, 29(3), 15–21.
- Taibi, T. (Ed.). (2007). *Design pattern formalization techniques*. Igi Global.
- Talen, E. (2015). Do-it-yourself urbanism: A history. *Journal of Planning History*, 14(2), 135–148.
- Tarandach, I., & Coles, M. J. (2020). *Threat Modeling*. O'Reilly Media, Inc."



- Tatam, M., Shanmugam, B., Azam, S., & Kannorpatti, K. (2021). *A review of threat modelling approaches for APT-style attacks | Elsevier Enhanced Reader*.  
<https://doi.org/10.1016/j.heliyon.2021.e05969>
- Thuraisingham, B. (2004). *Chapter 3 Data Mining for Counter-Terrorism*.
- Tiso, J. (2011). *Designing Cisco network service architectures (ARCH): Foundation learning guide*. Cisco press.
- UcedaVelez, T. (2012). *Real world threat modeling using the pasta methodology* (Technical Report. Open Web Application Security Project (OWASP)). OWASP App Sec EU.  
[https://www.owasp.org/images/a/aa/AppSecEU2012\\_PASTA.pdf](https://www.owasp.org/images/a/aa/AppSecEU2012_PASTA.pdf)
- UcedaVelez, T., & Morana, M. M. (2015). *Risk Centric Threat Modeling: Process for attack simulation and threat analysis*. John Wiley & Sons.
- Unger, B., & Tichy, W. F. (2000). Do design patterns improve communication? An experiment with pair design. *Proceedings of the International Workshop on Empirical Studies of Software Maintenance*.
- upravnik. (2016, January 26). *Cisco three-layer hierarchical model*. Study CCNA. <https://study-ccna.com/cisco-three-layer-hierarchical-model/>
- Verizon. (2021). *2020 DBIR Summary of Findings*. Verizon Enterprise.  
<https://enterprise.verizon.com/resources/reports/dbir/2020/summary-of-findings/>
- Verizon 2013 Annual Review*. (2013).  
[https://www.verizon.com/about/sites/default/files/annual\\_reports/2013/index.html](https://www.verizon.com/about/sites/default/files/annual_reports/2013/index.html)
- Vesiluoma, S. (2009). *Understanding and supporting knowledge sharing in software engineering*.
- Vokac & M. (2004). *On the practical use of software design patterns* [PhD Thesis.]. University of Oslo.
- Walker, K. (2021, January 28). *Cloud Security Alliance's New Internet of Things (IoT) Security Controls Framework Allows for Easier Evaluation, Implementation of Security Controls within IoT Architectures*.

- <https://www.businesswire.com/news/home/20210128005024/en/Cloud-Security-Alliance%E2%80%99s-New-Internet-of-Things-IoT-Security-Controls-Framework-Allows-for-Easier-Evaluation-Implementation-of-Security-Controls-within-IoT-Architectures>
- wallarm. (2020). *What is threat modeling ? Definition, Methods , Example*.
- <https://www.wallarm.com/what/what-is-threat-modeling>
- Warburton, D. (2017). *Terror threat as Heathrow Airport security files found dumped in the street*.
- <https://www.mirror.co.uk/news/uk-news/terror-threat-heathrow-airport-security-11428132>
- Wideli, W., Audinot, M., Fila, B., & Pinchinat, S. (2019). Beyond 2014: Formal Methods for Attack Tree-based Security Modeling. *ACM Computing Surveys (CSUR)*, 52(4), 1–36.
- Wildcard. (2021). *Threat Modeling* [New Service]. Wildcard Corp.
- <https://wildcardcorp.com/services/cybersecurity/threat-modeling>
- Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., & Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data*, 3(1), 1–9.
- Wuyts, K. (2015). *Privacy Threats in Software Architectures*.
- Wuyts, K., Sion, L., & Joosen, W. (2020a). Linddun go: A lightweight approach to privacy threat modeling. *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 302–309.
- Wuyts, K., Sion, L., & Joosen, W. (2020b). Linddun go: A lightweight approach to privacy threat modeling. *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 302–309.
- Wuyts, K., Van Landuyt, D., Hovsepyan, A., & Joosen, W. (2018). Effective and efficient privacy threat modeling through domain refinements. *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 1175–1178.

- Xiong, W., & Lagerström, R. (2019). Threat modeling—A systematic literature review. *Computers & Security, 84*, 53–69.
- Yang, P., & Zhu, Q. (2011). Finding key attribute subset in dataset for outlier detection. *Knowledge-Based Systems, 24*(2), 269–274.
- Yao, S., Guan, J., Ding, S., Zhang, H., & Song, F. (2014). Modeling and Analysis of Network Survivability under Attack Propagation. *International Conference on Applications and Techniques in Information Security*, 96–108.
- Yazar, Z. (2002). A qualitative risk analysis and management tool—CRAMM. *SANS InfoSec Reading Room White Paper, 11*, 12–32.
- Yin, C., Zhu, Y., Liu, S., Fei, J., & Zhang, H. (2018). An enhancing framework for botnet detection using generative adversarial networks. *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 228–234.
- Yoder, J., & Barcalow, J. (1998). Architectural Patterns for Enabling Application Security. *PLoP 1997 Conference*. <https://www.idi.ntnu.no/emner/tdt4237/2007/yoder.pdf>
- Yuan, X., Nuakoh, E. B., Beal, J. S., & Yu, H. (2014a). Retrieving relevant CAPEC attack patterns for secure software development. *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, 33–36. <https://doi.org/10.1145/2602087.2602092>
- Yuan, X., Nuakoh, E., Beal, J., & Yu, H. (2014b). *Retrieving relevant CAPEC attack patterns for secure software development*. <https://doi.org/10.1145/2602087.2602092>
- Yuan, X., Nuakoh, E., Williams, I., & Yu, H. (2015). Developing Abuse Cases Based on Threat Modeling and Attack Patterns. *J. Softw.* <https://doi.org/10.17706/jsw.10.4.491-498>
- Zhang, C. (2011). *An empirical assessment of the software design pattern concept*. Durham University.
- Zhao, J., Gu, J., & Liu, J. (2011). Research on Layer 2 Attacks of 802.11-Based WLAN. *International Conference on Information Computing and Applications*, 503–509.
- Zhou, Z. H. (2012). *Ensemble methods: Foundations and algorithms*. CRC press.

- Zhu, H. (2014). Cyberpatterns: Towards a Pattern Oriented Study of Cyberspace. In C. Blackwell, H. Zhu, & Cyberpatterns (Eds.), *Unifying Design Patterns with Security and Attack Patterns* (p. 7). Springer international.
- Zhu, Q., & Basar, T. (2012). *17 A hierarchical security architecture for smart grid*.
- Zhu, Y. (2015). *Attack Pattern Ontology: A Common Language for Cyber-Security Information Sharing*. TU Delft Publication, Master Thesis.

# APPENDIX

## Appendix 1. Code Boxes

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
#import matplotlib.gridspec as gridspec
import seaborn as sns
import squarify
from sklearn import model_selection

from sklearn import preprocessing
from sklearn import metrics
from sklearn import feature_selection
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import f_classif, f_regression

from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
```

**Code Box 1. Importation of Libraries and Modules.**

```
un_1 = pd.read_csv('../input/outlier-detection-datasets/backdoor/UNSW-NB15_1.csv', low_memory = False)
```

```
un2 = pd.read_csv('../input/outlier-detection-datasets/backdoor/UNSW-NB15_2.csv', low_memory = False)
```

```
un3 = pd.read_csv('../input/outlier-detection-datasets/backdoor/UNSW-NB15_3.csv')
```

```
un4 = pd.read_csv('../input/outlier-detection-datasets/backdoor/UNSW-NB15_4.csv')
```

**Code Box 2. Importation on un\_1, un\_2, un\_3, un\_4 datasets.**

```
cols = ['srcip', 'sport', 'dstip', 'dsport', 'proto', 'state', 'dur', 'sbytes', 'dbytes', 'sttl',
        'dttl', 'sloss', 'dloss', 'service', 'Sload', 'Dload', 'Spkts', 'Dpkts', 'swin', 'dwin',
        'stcpb', 'dtpcb', 'smeansz', 'dmeansz', 'trans_depth', 'res_bdy_len', 'Sjit', 'Djit',
        'Stime', 'Ltime', 'Sintpkt', 'Dintpkt', 'tcprrt', 'synack', 'ackdat', 'is_sm_ips_ports',
        'ct_state_ttl', 'ct_flw_http_mthd', 'is_ftp_login', 'ct_ftp_cmd', 'ct_srv_src',
        'ct_srv_dst', 'ct_dst_ltm', 'ct_src_ltm', 'ct_src_dport_ltm', 'ct_dst_sport_ltm',
        'ct_dst_src_ltm', 'attack_cat', 'Label']
```

**Code Box 3. Showing Variable Col Listing Column Header Names.**

```
un_1.columns = cols
un_1.head()
un2.columns = cols
un2.head()
un3.columns = cols
un3.head()
un4.columns = cols
```

**Code Box 4. Assigning Of Column Header Name to Individual Data Frame.**

```
mal = un_1['Label'] == 1
mal_1 = un_1[mal]
mal_1.sample(5)
```

**Code Box 5. Filtering of Malicious logs from the un\_1 Dataset.**

```
mal_2 = un2['Label'] == 1
mals2 = un2[mal_2]
mals2.sample(5)
```

**Code Box 6. Filtering of Malicious logs from the un\_2 Dataset.**

```
mal_3 = un3['Label'] == 1
mals3 = un3[mal_3]
mals3.sample(5)
```

**Code Box 7. Filtering of Malicious logs from the un\_3 Dataset.**

```
mal_4 = un4['Label'] == 1
mals4 = un4[mal_4]
mals4.sample(5)
```

**Code Box 8. Filtering of Malicious logs from the un\_4Dataset.**

```
merge = [un_1, un2, un3, un4]
df = pd.concat(merge)
```

**Code Box 9. Merging of four datasets.**

```
merge = [un_1, un2, un3, un4]
df = pd.concat(merge)
```

```
df.shape
```

**Code Box 10. Cumulative Malicious dataset.**

```
mvl = pd.DataFrame(un1.isnull().sum(), columns = ['no of missing values'])
mlf = mvl['no of missing values'] > 0
mvl[mlf]
```

**Code Box 11. Cumulative Malicious dataset.**

```
pvt = pd.DataFrame(un1.proto.value_counts())
pvt.shape
```

```
(129, 1)
```

**Code Box 12. Count of Protocols Used to Perpetuate Malicious Activities.**

```
udp = malicious['proto'] == 'udp'
mal_udp = malicious[udp]
```

```
unas = malicious['proto'] == 'unas'
mal_unas = malicious[unas]
```

```
maltcp = (malicious['proto'] == 'tcp')
mal_tcp = malicious[maltcp]
```

**Code Box 13. Classification of malicious protocols into dataFrames.**

```
m2 = [mal_tcp, mal_udp, mal_unas]
tmalpt = pd.concat(m2)
tmalpt
```

**Code Box 14. Merging of Highly Significant Malicious Protocols.**

```

ntcp = malicious[malicious.proto != 'tcp']
nudp = ntcp[ntcp.proto != 'udp']
nunas = nudp[nudp.proto != 'unas']
nunas.sample(10)

```

**Code Box 15.** Boolean Filtering For Low Significant Malicious Protocols.

```

nunas.loc[:, 'proto'] = 'others'

```

**Code Box 16.** Cumulating of Other Protocols.

```

tot = [nunas, tmalpt]
summary = pd.concat(tot)
totals = pd.DataFrame(summary['proto'].value_counts())
totals

```

**Code Box 17.** Value Count of Nunas and Tmalpt Dataframe.

```

attct = pd.DataFrame(un1['attack_cat'].value_counts(100) * 100)
attct

```

**Code Box 18.** Computing the % of Attack Distribution.

```

# pip install squarify

# Import Data
#df_raw = pd.read_csv("https://github.com/selva86/datasets/raw/master/mpg_ggplot2.csv")
#un_1.apply(lambda x: str(x[0]) + "\n (" + str(x[1]) + ")", axis=1)
#attct['attack_cat'].values.tolist()

# Prepare Data
dfx = un1.groupby('attack_cat').size().reset_index(name='counts')
labels = attct.index
colors = [plt.cm.Spectral(i/float(len(labels))) for i in range(len(labels))]

# Draw Plot
plt.figure(figsize=(12,8), dpi= 80)
squarify.plot(sizes=attct['attack_cat'], label=labels, color=colors, alpha=.9)

# Decorate
plt.title('TREEMAP OF ATTACK DISTRIBUTION IN THE UN1 DATASET')
plt.axis('off')
plt.show()

```

**Code Box 19.** Tree map visualization code of malicious traffic.



```
col = ['port', 'attack_counts']
txd.columns = col
txd2 = txd.sort_values('attack_counts', ascending = False)
txd2.head(10)
```

**Code Box 20.** Box 4. 19. Attack Count per Port

```
total_pkts_time = pd.DataFrame
(df.groupby('attack_cat')['dur'].sum().sort_values(ascending = False))

total_pkts_time
```

**Code Box 21.** Attack Category Time Duration in the Malicious Dataset.

```
tcp_time = pd.DataFrame(mal_tcp.groupby('attack_cat')['dur'].sum().sort_values(ascending =
False))

tcp_time

udp_time = pd.DataFrame(mal_udp.groupby('attack_cat')['dur'].sum().sort_values(ascending =
False))

udp_time

unas_time = pd.DataFrame(mal_unas.groupby('attack_cat')['dur'].sum().sort_values(ascending =
False))

unas_time

dur_nunas =
pd.DataFrame(nunas.groupby('attack_cat')['dur'].sum().sort_values(ascending =
False))

dur_nunas
```

**Code Box 22.** Attack category time duration for TCP UDP UNAS.

```

tcp_avg_time =
pd.DataFrame(mal_tcp.groupby(' attack_cat' )[' dur' ].mean().sort_values(ascendi
ng = False))

tcp_avg_time

udp_avg_time =
pd.DataFrame(mal_udp.groupby(' attack_cat' )[' dur' ].mean().sort_values(ascendi
ng = False))

udp_avg_time

unas_avg_time =
pd.DataFrame(mal_unas.groupby(' attack_cat' )[' dur' ].mean().sort_values(ascend
ing = False))

unas_avg_time

nunas_avg_time = pd.DataFrame(nunas.groupby(' attack_cat' )[' dur' ].mean())

nunas_avg_time

```

**Code Box 23.** Mean Duration of Attacks per Protocol.

```

Code:
mvs = un1[[' ct_flw_http_mthd' , ' is_ftp_login' ]]
mvs.shape
Output:
(321283, 2)

Code:
mvs2 = mvs.dropna ()
mvs2.shape
Output:
(22215, 2)

```

**Code Box 24.** Extracting the column with missing Values.

```

mvs2.describe ()

```

**Code Box 25.** Describing the column with missing Values.

**Code:**

```
mvsmean = mvs.copy()

mvsmean['ct_flw_http_mthd'].fillna(mvs['ct_flw_http_mthd'].mean(), inplace= True)

mvsmean['is_ftp_login'].fillna(mvs['is_ftp_login'].mean(), inplace = True)

mvsmean.columns = ['mean_ct_flw_http_mthd', 'mean_is_ftp_login']

mvsmean.isnull().sum()
```

**Output:**

```
mean_ct_flw_http_mthd    0
mean_is_ftp_login        0
dtype: int64
```

**Code Box 26.** Filling missing Values with Mean.

```
mvsmedian = mvs.copy()

mvsmedian['ct_flw_http_mthd'].fillna(mvsmedian['ct_flw_http_mthd'].median(),
inplace = True)

mvsmedian['is_ftp_login'].fillna(mvsmedian['is_ftp_login'].median(), inplace =
True)

mvsmedian.columns = ['median_ct_flw_http_mthd', 'median_is_ftp_login']

mvsmedian.sample(5)
```

**Code Box 27.** Filling missing Values with Median.

```
mvsffill = mvs.copy()

mvsffill.columns = ['ffill_ct_flw_http_mthd', 'ffill_is_ftp_login']

mvsffill['ffill_ct_flw_http_mthd'].fillna(method='ffill', inplace=True)

mvsffill['ffill_is_ftp_login'].fillna(method='ffill', inplace=True)

mvsffill.isnull().sum()

ffill_ct_flw_http_mthd    0
ffill_is_ftp_login        0
dtype: int64
```

**Code Box 28.** Backward filling missing values.

```
Code

dcols = ['ct_flw_http_mthd', 'is_ftp_login']

un1_dropped = un1.drop (dcols, axis =1)

un1_dropped. shape

Output:
(321283, 47)
```

**Code Box 29. Dropping and replacing missing values with new dataset**

```
#Confirmation

un1. shape [1] - un1_dropped. shape [1]

2
```

**Code Box 30. Confirming the columns of dropped and new dataset**

```
un1_mean = un1_dropped. copy()

un1_mean = un1_dropped. append(mvsmean)
```

**Code Box 31. Adding of new filled columns with mean**

```
Code:

#Confirmation

un1_mean. columns[-2]

Output:

'mean_ct_flw_http_mthd'

Code:

#Confirmation

un1_mean. columns[-1]

Output:

'mean_is_ftp_login'
```

**Code Box 32. Confirmation of adding of new filled columns with mean**

```
Code:
    un1_ff = un1_dropped.copy()
    un1_ff = un1_dropped.append(mvsffill)

Code:
    un1_ff.columns[-2]

Output:
    'ffill_ct_flw_http_mthd'

Code:
    un1_bf = un1_dropped.copy()
    un1_bf = un1_dropped.append(mvsbfill)

Code:
    un1_bf.columns[-2]

Output:
    'bfill_ct_flw_http_mthd'

Code:
    un1_median = un1_dropped.copy()
    un1_median = un1_dropped.append(mvsmedian)

Code:
    un1_median.columns[-2]

Output:
    'median_ct_flw_http_mthd'
```

**Code Box 33. Confirmation of the adding of new filled columns with forward filling, backward filling and median**

```
un_obj = un1.dtypes == object

un1_objects = un1.loc[:, un_obj]

un1_objects.head(10)
```

**Code Box 34 Identifying columns with object data type**

```
Y =un1_enc['Label' ]
Y. shape
(321283,)
```

**Code Box 35 Creation of the evaluation column Y**

```
X = un1_enc.drop(['Label' ], axis=1)
X. shape
(321283, 48)
```

**Code Box 36 Creation and Confirmation of the test column X**

```
lbl_enc = preprocessing. LabelEncoder()
un1_enc[' srcip' ]= lbl_enc. fit_transform(un1_enc[' srcip' ])
un1_enc[' sport' ]= un1_enc[' sport' ]. astype(int)
un1_enc[' dstip' ]= lbl_enc. fit_transform(un1_enc[' dstip' ])
#un1_enc[' dsport' ]= lbl_enc. fit_transform(un1_enc[' dsport' ])
un1_enc[' proto' ]= lbl_enc. fit_transform(un1_enc[' proto' ])
un1_enc[' state' ]= lbl_enc. fit_transform(un1_enc[' state' ])
un1_enc[' service' ]= lbl_enc. fit_transform(un1_enc[' service' ])
#un1_enc[' ct_ftp_cmd' ]= un1_enc[' ct_ftp_cmd' ]. astype(float)
#un1_enc[' ct_ftp_cmd' ]. fillna(0, inplace=True)
#un1_enc[' ct_ftp_cmd' ] = un1_enc[' ct_ftp_cmd' ]. astype(float)
un1_enc[' attack_cat' ]= lbl_enc. fit_transform(un1_enc[' attack_cat' ])
```

**Code Box 37. Encoding objects with Label Encoder**

```
from sklearn. feature_selection import SelectKBest, chi2
KBest = SelectKBest(chi2, k=10). fit(X3, y2)
Feature selection using SelectKBest algorithm
f = KBest. get_ support(1) #the most important features
X_new = X[X. columns[f]] # final features`
X_new
```

**Code Box 38 Deploying Chi2 on X and Y sets**

```

#Importation of the libraries and modules for the machine learning process

import itertools

#Handling of arrays
import numpy as np
#Data visualization
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
#Importation of experimental datasets eg. Iris
from sklearn import datasets
#Logistic regression algorithm
from sklearn.linear_model import LogisticRegression
#KNN algorithm
from sklearn.neighbors import KNeighborsClassifier
#Gaussian
from sklearn.naive_bayes import GaussianNB
#Random forest
from sklearn.ensemble import RandomForestClassifier
#Ensemble/algorithm combiner
from mlxtend.classifier import StackingClassifier
#Accuracy measurement and splitting of algorithms respectively
from sklearn.model_selection import cross_val_score, train_test_split
#Visualizations for the algorithm outputs
from mlxtend.plotting import plot_learning_curves
from mlxtend.plotting import plot_decision_regions

```

**Code Box 39 Importing Machine Learning Algorithms**

```

X, y = cpdf3.drop(['attack_cat'], axis=1), cpdf3['attack_cat']
clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
sclf = StackingClassifier(classifiers=[clf1, clf2,
clf3], meta_classifier=lr)

#Preparation for the machine learning algorithm

```

**Code Box 40 Preparation for the machine learning algorithms**

```

#KMeans algorithm was used n_clusters divide the data into three parts
clusterer_final = KMeans(n_clusters=3, random_state=0).fit(X)
processed_plot = pd.DataFrame(X25) #to use pandas methods
## add a column with the predictions
processed_plot['KMeans_cluster'] = clusterer_final.labels_
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
processed_plot['results'] =
le.fit_transform(processed_plot['KMeans_cluster'].values)
## sort the samples (axis 0) by cluster
#processed_plot.sort_values('KMeans_cluster', axis = 0, inplace=True)
#The K-Means machine learning algorithm

```

**Code Box 41. Splitting of data into three clusters for classification of attack surface.**

```

processed_plot['results'].value_counts ()

0    11214
1     6770
2     4231

```

**Code Box 42. Three clusters obtained from K-means**

```

pp4 = processed_plot['results'].replace({0: 'Hosts', 1: 'Users', 2:
'Media'})
pp4.to_frame()

```

**Code Box 43. Three clusters mapped to their respective surface**

```

processed_plot['ys'] = pp4
processed_plot.sample(20)

```

**Code Box 44. Mapping the outputs to the select Kbest algorithm**

```

logs = unidrrrows[['srcip', 'sport', 'dstip', 'dsport', 'proto', 'state',
'outputs']]
logs.sample(10)

```



**Code Box 45. Simple output for the attack surfaces**

```
distribution =  
pd.DataFrame(unldrrows.groupby(' attack_cat')[ ' outputs' ].value_counts().sort_values  
(ascending = False))  
  
distribution  
  
#Code for mapping surface of attacks to the attacks
```

**Code Box 46. Mapping surface of attacks to the respective attacks**

```
ht = dist[' surface' ] == ' Hosts'  
host = dist[ht]  
md = dist[' surface' ] == ' Media'  
media = dist[md]  
us = dist[' surface' ] == ' Users'  
users = dist[us]
```

**Code Box 47. Classifying the three-attack surface to different datasets.**

```
h = np.array(host[' outputs' ])  
m = np.array(media[' outputs' ])  
u = np.array(users[' outputs' ])
```

**Code Box 48 47. Filtering surface to individual dataset.**

```
h  
array([4393, 2884, 1705, 1195, 654, 215, 80, 65, 23])  
#Attack distribution in the host surface  
  
m  
array([2364, 730, 454, 227, 226, 226, 3, 1, 0])  
#Attack distribution in the media surface  
  
u  
array([3963, 1437, 562, 286, 235, 228, 51, 7, 1])  
#Attack distribution in the users' surface
```

**Code Box 49 Attack distribution per surface**

```

plt.figure(figsize = (10, 13))
barWidth = 0.25
pos1 = np.arange(len(h))
pos2 = [x + barWidth for x in pos1]
pos3 = [x + barWidth for x in pos2]
plt.bar(pos1, h, color = '#FBC02D', width = barWidth, label = 'Hosts')
plt.bar(pos2, m, color = '#F57F17', width = barWidth, label = 'Media')
plt.bar(pos3, u, color = '#E65100', width = barWidth, label = 'Users')
plt.xlabel('Attacks', fontweight = 'bold')
plt.ylabel('Number of Attacks Per Surface', fontweight = 'bold')
plt.xticks([i + barWidth for i in range(len(h))], ['Generic', 'Exploits',
'Fuzzers', 'Reconnaissance', 'DoS', 'Backdoors', 'Analysis', 'Shellcode',
'Worms'], rotation = 45)
plt.legend()
plt.show()
np.arange(len(h))

```

**Code Box 48. Visualizing attack distribution per surface.**

```

capec1 = pd.read_csv('../input/capecs/1000.csv')
capec2 = pd.read_csv('../input/capecs/2000.csv')
capec3 = pd.read_csv('../input/capecs/3000.csv')
capec3.reset_index(level=0, inplace=True)
#Importation of the CAPEC datasets

```

**Code Box 49 Importation of CAPEC datasets.**

```

ccols = ['ID', 'Name', 'Abstraction', 'Status', 'Description',
'AlternateTerms', 'LikelihoodOfAttack', 'TypicalSeverity',
'RelatedAttackPatterns', 'ExecutionFlow', 'Prerequisites',
'SkillsRequired', 'ResourcesRequired', 'Indicators',
'Consequences',
'Mitigations', 'ExampleInstances', 'RelatedWeaknesses',
'TaxonomyMappings', 'Notes']
#List for storing the column names

```

**Code Box 50 List for storing the column names**

```

capec3.columns = ccols
#Subjection of the new columns to the dataset

```

**Code Box 51 Subjecting New Columns to the dataset**

```
gen = capec3[capec3['Name'], str.contains("Generic")]
gen
```

**Code Box 52. Extracting generic attacks**

```
exp = capec3[capec3['Name'], str.contains("Exploits")]
exp
```

**Code Box 53 Extracting exploits attacks**

```
fuz = capec3[capec3['Name'], str.contains("Fuzzers")]
fuz
```

ID	Name	Abstraction	Status	Description	AlternateTerms	LikelihoodOfAttack	TypicalSeverity	RelatedAttackPatterns	ExecutionFlow	Prerequisites	SkillsRequired	ResourcesRequired	Indicators	Consequences
<														>

```
fuz2 = capec3[capec3['Name'], str.contains("FUZZING")]
fuz2
```

ID	Name	Abstraction	Status	Description	AlternateTerms	LikelihoodOfAttack	TypicalSeverity	RelatedAttackPatterns	ExecutionFlow	Prerequisites	SkillsRequired	ResourcesRequired	Indicators	Consequences
<														>

```
fuz3 = capec3[capec3['Name'], str.contains("Fuz")]
fuz3
```

**Code Box 54 Extracting Fuzzer attacks**

```
dos = capec3[capec3['Name'], str.contains("Denial")]
dos

dos2 = capec3[capec3['Name'], str.contains("DoS")]
dos2
```

**Code Box 55 Extracting DoS attacks.**

```
rec = capec3[capec3['Name'], str.contains("Reconnaissance")]
rec
```

**Code Box 58. Extracting reconnaissance attacks**

```
an = capec3[capec3['Name' ]. str. contains("Analysis")]
an
```

**Code Box 59. Extracting analysis attacks**

```
sc = capec3[capec3['Name' ]. str. contains("Shell")]
```

**Code Box 60. Extracting Shell attacks.**

```
bd = capec3[capec3['Name' ]. str. contains("Backdoor")]
bd
bd. shape
(0, 20)
wm = capec3[capec3['Name' ]. str. contains("Virus")]
wm
wm. shape
(0, 20)
```

**Code Box 56 Extracting Backdoor and virus attacks.**

```
findings = [gen, exp3, fuz3, dos, dos2, rec, an, sc, wm]
filt = pd.concat(findings)
filt. shape
(25, 20)
```

**Code Box 57. Merging of attacks in UNSW-NB\_15 found in CAPEC**

```
fh = (filt['LikelihoodOfAttack'] == 'High') |
(filt['TypicalSeverity'] == 'High')
fh2 = filt[fh]
filtfh2 = fh2[['Name', 'LikelihoodOfAttack', 'TypicalSeverity']]
filtfh2
```

**Code Box 58. STRIDE/CAPEC Mapping Algorithm.**

```

filtfh3 = filtfh2.copy()

filtfh3['LikelihoodOfAttack'] = filtfh3['LikelihoodOfAttack'].replace({'Very
High':4,
                                                                    'High' : 3,
                                                                    'Medium' :2,
                                                                    'Low' : 1,
                                                                    None : 2})
filtfh3['TypicalSeverity'] = filtfh3['TypicalSeverity'].replace({'Very High' : 4,
                                                                    'High' : 3,
                                                                    'Medium' : 2,
                                                                    'Low' : 1})

filtfh3

```

**Code Box 59. Converting CAPEC Qualitative log to Quantitative**

```

filtfh4 = filtfh3.copy()
filtfh4['RiskScore'] = filtfh4['LikelihoodOfAttack'] *
filtfh4['TypicalSeverity']
filtfh4

```

**Code Box 60 Converting CAPEC Qualitative log to Quantitative.**

```

un_cp = filt[['Name', 'LikelihoodOfAttack', 'TypicalSeverity']]

un_cp.loc[:, 'LikelihoodOfAttack'] =
un_cp.loc[:, 'LikelihoodOfAttack'].replace({'Very High' : 4,
                                                                    'High' : 3,
                                                                    'Medium' : 2,
                                                                    'Low' : 1,
                                                                    None : 2})

un_cp.loc[:, 'TypicalSeverity'] = un_cp.loc[:, 'TypicalSeverity'].replace
                                                                    ({'Very High' : 4,
                                                                    'High' : 3,
                                                                    'Medium' : 2,
                                                                    'Low' : 1,
                                                                    None : 2})

```

**Code Box 61. Missing Value with average of 2.**

```

un_cp3 = un_cp2.copy()

un_cp3['RiskScore'] = un_cp3['LikelihoodOfAttack'] *
un_cp3['TypicalSeverity']
un_cp3

```

**Code Box 67. Replacing Missing Value with Average of 2.**

```

un_cp4 = un_cp3.copy()
un_cp4.dropna()
un_cp4['RiskScore'] = un_cp4['RiskScore'].replace({1 : 'Low', 2 : 'Low', 3 :
'Low', 4 : 'Medium', 5 : 'Medium', 6 : 'Medium', 7 : 'Medium', 8 : 'High', 9 :
'High', 10 : 'High', 11 : 'High', 12 : 'Very High', 13 : 'Very High', 14 : 'Very
High', 15 : 'Very High', 16 : 'Very High', None : 'Medium'})

```

**Code Box 68 Final risk Score Mapping**

```

mitigation = filt[['Name', 'Mitigations']]
mitigation

```

**Code Box 69 Extracting defense mechanisms**

## Appendix 2: Machine learning libraries and modules

### 1. Data Handling:

- Pandas library: Used for handling DataFrames.
- NumPy library: Used for mathematical operations and array handling.
- Matplotlib.pyplot: Used for basic data visualizations.
- Seaborn: Utilized for more customizable and flexible visualizations compared to Matplotlib.
- Squarify: Used for treemaps to provide a comparative visualization of values.

### 2. Data Splitting and Preprocessing:

- model\_selection module: Applied for train-test split, especially recommended for balanced large datasets like UNSW-NB15.
- preprocessing module: Used to transform data into various formats before applying machine learning algorithms.

### 3. Model Evaluation:

- metrics module: Utilized for evaluating the accuracy of various machine learning algorithms.

### 4. Feature Selection:

- feature\_selection module: Used to identify and select columns that contribute significantly to the final output.
- SelectKBest: Another method for selecting the best-performing columns for machine learning algorithms, also applicable for dimensionality reduction.

### 5. Algorithms Classifiers and clustering:

#### **K-Nearest Neighbors (KNN):**

- **Type:** Supervised Learning (Classification, Regression)
- **Summary:** KNN is a simple, instance-based learning algorithm. It classifies or predicts the output of a data point based on the majority class or average value of its k-nearest neighbors in the feature space. The choice of 'k' determines the number of neighbors considered.

#### **Random Forest (RF):**

- **Type:** Supervised Learning (Classification, Regression)
- **Summary:** Random Forest is an ensemble learning method that builds a multitude of decision trees during training. It combines their predictions to improve accuracy and control overfitting. Each tree is constructed using a subset of the data and features, and the final prediction is an aggregate of individual tree predictions.

#### **Gaussian Naive Bayes (GNB):**

- **Type:** Supervised Learning (Classification)
- **Summary:** Naive Bayes is a probabilistic algorithm based on Bayes' theorem. The "Gaussian" variant assumes that the features follow a normal distribution. It is called "Naive" because it assumes that features are conditionally independent given the class label. Despite its simplicity and assumptions, Naive Bayes can perform well in various classification tasks.

#### **Logistic Regression (LR):**

- **Type:** Supervised Learning (Classification)
- **Summary:** Despite its name, logistic regression is used for binary classification problems. It models the probability that an instance belongs to a particular class. The logistic function is applied to a linear combination of input features, mapping the result to a probability between 0 and 1. A threshold is then applied to make the final classification decision.

#### **K-Means Clustering:**

- **Type:** Unsupervised Learning (Clustering)
- **Summary:** K-Means is a clustering algorithm that partitions data points into 'k' clusters based on similarity. It assigns each data point to the cluster whose mean represents the point's features. The algorithm iteratively refines the cluster assignments until convergence. K-Means is widely used for grouping data in unsupervised learning scenarios.

#### **6. Validation and Testing:**

- `cross_val_score` module: Employed to evaluate the accuracy of various machine learning algorithms on a subset of a dataset.
- `chi2` module: Used for statistical testing of the difference between expected and observed outputs.

#### **7. Visualization and Interpretation:**

- `Mlexend` library: Used for creating visualizations, including decision regions after clustering using algorithms like `KMeans`.

#### 8. **Specific Configurations:**

- For the KNeighbors Classifier, the value of K was set to 3 for clustering according to host, media, or users.

#### 9. **Considerations and Limitations:**

- Decision trees can become complex with an increase in attributes and are sensitive to training data changes.
- Optimal K value is crucial in KNeighbors Classifier to avoid overfitting or computational complications.
- LogisticRegression uses a logistic function for prediction, differentiating it from linear regression.



### **Appendix 3: List of Publications**

1. Yoga, C. A., Rodrigues, A. J., & Abeka, S. O. (2023). Holistic Security Pattern-Based Model to Protect Network Architecture. *Holistic Security Pattern-Based Model to Protect Network Architecture*, 130(1), 11-11.
2. Castro A Yoga, Anthony J Rodrigues and Silvance O Abeka. Hybrid Machine Learning Approach for Attack Classification and Clustering in Network Security. *International Journal of Computer Applications* 185(31):45-51, August 2023.

## Appendix 4: Board of Postgraduate Research Approval



**JARAMOGI OGINGA ODINGA UNIVERSITY OF SCIENCE & TECHNOLOGY**  
**BOARD OF POSTGRADUATE STUDIES**  
*Office of the Director*

Tel. 057-2501804  
Email: [bps@jooust.ac.ke](mailto:bps@jooust.ac.ke)

P.O. BOX 210 - 40601  
**BONDO**

**Our Ref:** I162/4608/2014

**Date:** 27<sup>th</sup> August 2020

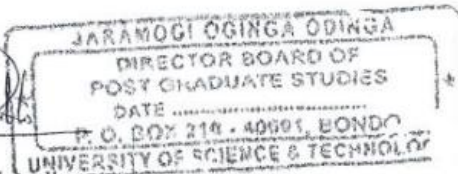
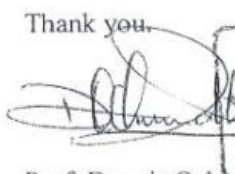
**TO WHOM IT MAY CONCERN**

**RE: YOGA AUMA CASTRO– I162/4608/2014**

The above person is a bonafide postgraduate student of Jaramogi Oginga Odinga University of Science and Technology in the School of Informatics & Innovative Systems pursuing a PhD in Information Technology, Security and Audit. He has been authorized by the University to undertake research on the topic: “*Defensive Procedural Security Design Patterns Model for the Network Architecture.*”

Any assistance accorded to him shall be appreciated.

Thank you.



Prof. Dennis Ochuodho  
**DIRECTOR, BOARD OF POSTGRADUATE STUDIES**

## Appendix 4: Ethical approval



**JARAMOGI OGINGA ODINGA  
UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**DIVISION OF RESEARCH, INNOVATION AND OUTREACH  
JOOUST-ETHICS REVIEW OFFICE**

Tel. 057-2501804  
Email: [erc@jooust.ac.ke](mailto:erc@jooust.ac.ke)  
Website: [www.jooust.ac.ke](http://www.jooust.ac.ke)

P.O. BOX 210 - 40601  
BONDO

OUR REF: JOOUST/DVC-RIO/ERC/E4

10<sup>th</sup> February, 2023

Castro Auma Yoga  
SIIS  
**JOOUST**

Dear Mr. Yoga,

**RE: APPROVAL TO CONDUCT RESEARCH TITLED "DEFENSIVE  
PROCEDURAL SECURITY DESIGN PATTERNS MODEL FOR THE NETWORK  
ARCHITECTURE"**

This is to inform you that JOOUST ERC has reviewed and approved your above research proposal. Your application approval number is **ERC 36/02/23-9/08**. The approval period is from 10<sup>th</sup> January, 2023– 09<sup>th</sup> January, 2024.

This approval is subject to compliance with the following requirements:

- i. Only approved documents including (informed consents, study instruments, MTA) will be used.
- ii. All changes including (amendments, deviations and violations) are submitted for review and approval by JOOUST IERC.
- iii. Death and life threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to NACOSTI IERC within 72 hours of notification.
- iv. Any changes, anticipated or otherwise that may increase the risks of affected safety or welfare of study participants and others or affect the integrity of the research must be reported to NACOSTI IERC within 72 hours.
- v. Clearance for export of biological specimens must be obtained from relevant institutions.
- vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
- vii. Submission of an executive summary report within 90 days upon completion of the study to JOOUST IERC.

Prior to commencing your study, you will be expected to obtain a research permit from National Commission for Science, Technology and Innovation (NACOSTI) <https://oris.nacosti.go.ke> and also obtain other clearances needed.

Yours sincerely,

Prof. Francis Anga'wa  
Chairman, JOOUST ERC

Copy to: Deputy Vice-Chancellor, RIO

Director, BPS

DEAN, SIIS

/dm