Cyber Physical System Security Model For Remote Sensing Device Protection: A Technical Review

Dr. Silvance Abeka

School Of Informatics and Innovative Systems
JaramogiOgingaOdinga University of Science & Technology

Abstract

Remote sensing devices have emerged as a significant technology that enables the monitoring of processes and physical environment from far away locations. These remote sensing systems include physiological sensors installed on human body to continuously monitor the health and enable the fast detection of medical emergencies and the delivery of therapies (Body Area Networks-BAN), smart buildings that detect absence of occupants and shut down the cooling unit to save energy, data centers that use solar energy for cooling purposes, and unmanned aerial vehicles (UAVs) that use an image of the terrain to perform surveillance. They have been used to provide services such as automated pervasive health care, smart electricity grid, green cloud computing, and surveillance with UAVs. Since these systems utilize information from the physical environment and in turn affect the physical environment during their operation, any vulnerabilities, threats and attacks can expose the monitored process or physical environment to a number of risks. This paper seeks to develop Cyber Physical System Security Model for protecting remote sensing devices. The paper also investigatedthe tight coupling between the cyber and the physical in Cyber Physical Systems (CPSs), to establish the various forms of risks that have not been considered adequately in the traditional computing domain such as the cyber element adversely affecting the physical environment.

Keywords: *Remote Sensing Systems, Risk Anaysis, Body Area Networks, Supervisory Control and Data Acquisition,Industrial Control System, Cyber Physical System*

I. Introduction

Infrastructure constitutes any physical asset capable of being utilized to produce services or support the structure and operation of a society or an enterprise and include roadways, bridges, airports and airway facilities, mass transportation systems, waste treatment plants, energy facilities, hospitals, public buildings and space or communication facilities. Critical infrastructure on the other hand consist of physical, virtual facilities and services that form the basis for a nation's defense, a strong economy, health and safety of its citizens. It is charged with the provision of necessities such as water and food, electricity and gas, telecommunications and broadcasting, health services, the financial system and the transportation system.

Every critical infrastructure constitute of an Industrial Control System (ICS) that is made up of Supervisory Control and Data Acquisition (SCADA) systems and other types of control systems that monitor processes and control flows of information. ICS serve to regulate the flow of natural gas to a power generation facility or the flow of electricity from a grid to a home. Cyber systems form the central infrastructure of critical sectors as nearly all of them utilize IT to facilitate core business processes. Given their high value nature, the cyber systems of critical infrastructure have become targets for attack, and their disruptions have led to extensive economic, political and social effects.

The cyber systems consist of various software, the development of which, according to Sebastian and Stephan (2018), comprises of diverse activities such as implementing new features, analyzing requirements, and fixing bugs. Universal satellite and data connectivity is one of the major advancements in seafaring. Many critical systems on board rely on the Global Navigation Satellite System (GNSS) for safe navigation, communication, emergency response, and traffic control. However, disrupted or manipulated Global Positioning System (GPS) signals can send ships off their course and cause collisions, groundings, and environmental disasters (Dennis et al., 2017).

Threats to critical infrastructures are perpetrated through electronic, radio-frequency or computer-based attacks on the information components that control these critical infrastructures. In addition, these critical infrastructure systems have vulnerabilities that can be

exploited through threat vectors, which can be technical or non-technical. Despite the growth in application security testing over time, applications executing in critical infrastructures still remain insecure. Scans of thousands of applications and billions of lines of code have found a widespread weakness in applications, which has become a top target of cyber attackers.

According to Tobby (2017),the Internet of Things (IoT) is a vital concept embedded within a larger spectrum of networked products and digital sensors. This technology has caused an explosion of applications, marking a fundamental shift in the way human beings interact with the Internet and presenting both opportunities and challenges, particularly with respect to critical infrastructure. For instance hackers have used IoT devices such as printers, thermostats and videoconferencing equipment to breach security systems.

The Internet-enabled infrastructures have facilitated home automation, energy-management systems, smart homes, network-enabled medical gadgets, intelligent vehicles, networked traffic systems, road and bridge sensors, innovations in agricultural, industrial, energy production and distribution. Although this has opened up numerous avenues for efficiency, the unregulated rise of the IoT raises a plethora of issues such as security and privacy of people, telecoms networks and power utilities. This is due to illegitimate breaches of the networks undergirding critical infrastructure since the efficiency of Internet connectivity also accelerates susceptibility to security violations through the misuse of IoT data.

Although an ICS is air-gapped and hence a closed system, it may not be vulnerable to virtual attacks but is still susceptible to attacks perpetrated through physical access such as from infected removable devices. As technology continues to grow, a number of ICSs have been connected to the Internet, making them vulnerable to multifarious attacks. Computers and communications being critical infrastructures in their own right are increasingly connecting other infrastructures together. The increased connectivity means that a disruption in one network may lead to disruption in another and hence reliance on computers and networks increases critical infrastructure's vulnerability to cyber attacks.

According to Arash and Stuart (2015), CPS provides the control of physical components through cyber based commands and its operations are integrated, monitored, or controlled by a computational core. By integrating actuators, control processing units, sensors, and communication cores, a CPS forms a control loop for each of the physical component of the system. The major components of a CPS are SCADA, distributed control system (DCS), and program logic controller (PLC).

The SCADA systems gather and control geographically dispersed assets ranging from controlling sensors within a plant to controlling power dissemination in a country. They are heavily utilized in various critical infrastructures such as electrical power grids, water distribution systems, and oil refineries. On the other hand, DCS manages the controllers that are grouped together to carry out a specific task within the same geographically location. Both SCADA and DCS employ PLC devices to manage industrial components and processes. PLCs are typically programmed from a Windows-based machine by an operator. The operators utilizes SCADA and DCS for various controlling tasks such as process monitoring and configuring control parameters.

In their paper, Lange et al., (2016) point out that the success of a business mission is highly dependent on the Communications and Information Systems (CIS) that support the mission. As such, cyber attacks on CIS degrade or disrupt the performance and completion of the associated mission capability. On an operational level, an electrical grid's mission is to deliver electricity from suppliers to consumers. For monitoring and control purposes, they are connected to CIS. The operability, performance, or reliability of an application may depend on multiple network services spanning multiple network devices and sub-networks of an infrastructure.

The risks associated with vulnerable software deployed in enterprise environments have exposed customer data or intellectual property and can be caused by attackers exploiting weaknesses in web applications or desktop software. Lack of consistent, proactive policies to manage vulnerabilities associated with the Bring Your Own Device (BYOD) trend. Mobile devices come with one huge challenge of ensuring that all valuable information is secure and the increasing number of these devices elevates the threat of accidental and intentional security breaches. As such, verifying the security of the software being downloaded to those devices is becoming a business priority. This is important since platforms such as Google's Android do minimal vetting of the safety of applications before permitting consumers to download from their App store.

Application security performance has been noted to reduce greatly owing to a number of factors such as the vulnerabilities in commercial software that permit remote code execution and backdoor functionality,

cross-site scripting and SQL injection inherent in Government applications than other industry sectors, and a number of Android applications that were found to contain hard-coded cryptographic keys.

According to Mark (2017), mission function is increasingly delivered in software. For instance,F-35 aircraft depends on more than twenty million lines of code to fuse information from the JSF's radar, infrared cameras, jamming gear, other planes and ground stations. This aids it to locate and hide from opponents, as well as break through enemy lines to hit targets on the ground. Steel furnaces have been successfully attacked, causingmassive damage to furnace. This is exampled by a targeted APT attack on a German steelworks which ended in the attackers gaining access to the business systems and through them to the production network, including SCADA. Consequently, the attackers gained control of a steel furnace, causing massive damages to the plant.

In addition, electric grids have come under attacks, such as the BlackEnergy Trojan that attacked Ukrainian electric power industry. The weapons platforms have also become potential cyber attacktargets.The Joint Strike Fighter aircraft relies on more than twenty million lines of code and the Pentagon canceled a cyber test due to concerns it would damage the Autonomic Logistics Information System that identifies broken parts and other faults. On their hand, embedded systems present new classes of vulnerabilities owing to their different characteristics compared to other systems. They present more and varied attack surfaces such as sensors, multiple command-and-control masters, embedded firmware, unique internal busses and controllers, size, weight, power and latency that demand tradeoff against defense-in-depth, timing demands that offer potential side channels such as bit and clock cycle level operations, physical resources with real time sensors, safety-critical real-time operating system, confusion between failure resilience and attack, and intermittent communications.

Connecting automotive systems to internet opens system to attack as this system extensions opens vulnerabilities not anticipated during the design. Moreover, optimizations performed assume one attack method and this assumption no longer hold due to additional integrations. This new operational environment is a major cause for the introduction of new vulnerabilities in existing systems. Machine-learning based systems increase systems exposures. For instance Tesla car's driverless technology failed to detect the white side of the tractor-trailer against a brightly lit sky, and hence never activated its brakes. This shortcoming can be attributed to the fact that although conventional code development techniques of modest could have helped, operations are driven by high volume, high velocity sensor data, and that decision making is based on trained models of behaviors, which experience some limits.

II. Related Work

A study by Noel et al., (2016) pointed out that the Vital Infrastructure, Networks, Information and Control Systems Management (VIKING) project was to investigate the vulnerability of SCADA systems and the cost of cyber attacks on society, focusing on systems for transmission and distribution of electric power. This was achieved by adopting a model-based approach to investigating SCADA system vulnerability. According to Motzek and Möller (2017), mission impact has to be considered in the context of what impact the adversary desires, meaning that if knowledge or estimation of the intents, motivations and anticipations of the adversary is possible, then the impact of the adversary on missions or the intended impact would be easier to assess.

Mark (2017) explains that catching software faults early saves money as these faults accounts for 30–50% percent of total software project costs. Tobby (2017) points out that whereasthe change in ICS architecture supports new information system capabilities, it provides significantly less isolation for these systems from the outside world. This introduces vulnerabilities that exist in current networked information systems. Some of the shortcomings of these ICSs include outdated and difficult to update software, sensors and controls running many contemporary facilities and equipment. This means that organizations are unable to incorporate new features and improvements. There is also inadequate integration between internal systems such as managerial apps, plant data sources, and external partners, which creates data silos. Aging operating systems and vulnerable operational technologies pose security risks because they cannot be easily retired or replaced. Moreover, as Daugherty et al. (2015) explains, there is limited embedded computing or intelligence control at the device, product or plant level.

Tobby (2017) discuss that all critical infrastructure systems have vulnerabilities that can be exploited through threat vectors. These vulnerabilities may be technical or non-technical. Whereas technical vulnerabilities are application-based, non technical ones are common Internet protocols vulnerabilities. The core protocols such as IP, TCP and HTTP were created and implemented without factoring in security features since the Internet was initially used to serve academic

and governmental environments, wherein the users were trusted entities.

Randy and Susan (2016) point out thatdevelopers need knowledge on secure coding techniques and tools that render software reliable, robust, and secure. Secure software design is vital to prevent loss of data, premature leaks of data and downtime of resources. The main reason for the escalation of cyber attacks in the field of critical infrastructure (CI) is that most control systems used for CI do not utilize propriety protocols and software any more, but standard solutions. This implies that critical infrastructure systems are becoming vulnerable and exposed to cyber threats.

A major factor for the decreasing security of SCADA networks is the use of commercial off-the-shelf (COTS) hardware and software to develop devices. In their report, Katerina and Jacob (2017) explain that space missions provide valuable services to the society from navigation, to earth observation, weather forecasting, and communication. As such, space missions are part of the critical infrastructure and are regularly targeted by attackers. For instance, NASA experiences 29,000 malicious incidents against its systems, 17,500 suspicious e-mails, and 250 unique incidents against its web sites on a weekly basis. It is therefore vital to utilize software development and assurance practices that account for cyber security concerns.

Chris (2017) analyzed some software vulnerabilities and established that many applications are not being assessed for security at all. This has facilitated cyber attacks on elections in the U.S. and other democracies, demonstrating that most critical systems are in the cross-hairs. Global cyber attacks on a massive scale such as the WannaCry and Petya ransomware attacks, cyber attacks on electric utilities, cyber wars between nation states have all created a sense of urgency of tackling the problem of insecure software. Unfortunately, more attention has drifted away from prevention towards detection and response.

Arman et al., 2018 explains that software architecture includes many variation points that can take on one of a set of possible alternatives such as employing either an encrypted or plain-text data storage, using a relational database, a document database, or a key-value store. A design decision involves the selection of one of these alternatives. During decision making, architects carefully assess each alternative and how it satisfies or affects each of the system's requirements. Unfortunately, this is frequently not done in practice. An illustration of ineffective design-decision impact assessment is the Healthcare.gov portal that resulted to serious technical problems at launch and a development cost. The portal's downtimes of up to 60% were caused by flawed architectural and deployment design decisions. The system was deployed using a single-node NoSQL database that also stored federal government employee information instead of using a distributed database configuration.

III. Approach

This paper utilized secondary data from mission critical security reports, cyber security reports, government website data, security expert opinions and past research papers to get a glimpse of the state of mission critical security issues.

IV. Empirical Analysis of Mission Critical Security Issues

Di Martino et al. (2014) analyzed the failures of the Blue Waters, the Cray hybrid (CPU/GPU) supercomputer, and found that software was the largest contributor to the node repair hours (53%), even though it caused only 20% of the total number of failures. The security vulnerabilities published in the Bugtraq database and CERT advisories have been analyzed. Out of the twelve classes used to classify 5,925 Bugtraq reports on software related vulnerabilities, five classes dominated: input validation errors (23%), boundary condition errors (21%), design errors (18%), failure to handle exceptional conditions (11%), and access validation errors (10%).

In their paper, Lange et al., (2016) report that the energy sector reported an increase in frequency and sophistication of cyber attacks on electricity systems. Hambling, (2017) found out that multiple ships outbound from the united states have reported GPS interferences and reports have emerged of more than 20 vessels which noticed spoofed GPS signals that placed them about 25 nautical miles inland. The National Institute of Standards and Technology, (2017) explains that this is alarming considering that these are naval vessels carrying advanced weaponry as well as the commercial shipping sector, which is part of the critical infrastructure and accounts for more than 90% of cargo transported globally.

Navalservices such as voice communications, crew welfare and entertainment systems, guest Wi-Fi, and video monitoring are perceived to be less critical to safety and operations and hence routinely left unpatched and exposed to attacks. The IT networks in ships are employed for accounting, cargo management, customs and shipping, human resource planning, and administration. According to Hudson AnalytixInc, (2017), a malware outbreak in 2017 paralyzed IT networks across the world and caused significant

business disruptions and loss of revenue. The Symantec Security Response, (2017) point out that the NotPetya worm initially infected computers through a malicious update in an accounting software product, which then spread to attached systems, wiping or encrypting files and demanding ransom payments. This can be attributed to the fact that the design and configuration of the links between IT networks seldom take into account authentication and encryption methods, hence exposing potential vulnerable and legacy system to the internet. Since IT systems on vessels are often connected with onshore facilities, this further increases the exposure to systemic and persistent threats.

The Electronic Chart Display Information System (ECDIS) mandated by the International Maritime Organization (IMO) for all commercial vessels is normally installed on the bridge. Unfortunately, the ECDIS software implementations have a number of weaknesses such as running on legacy computers for which no security updates are available, maps get loaded onto the system either via the internet, USB or DVD, sensor feeds comes from a multitude of other onboard systems such as Radar, Navigation Telex (Navtex), ICS, and satellite terminals. This provides a wide surface for any attack. Commercial ECDIS software has some significant security risks that allow attackers to replace or delete files on the system or inject malicious content. Consequently, tampered sensor data could be sent to ECDIS, which would influence decisions for navigation, and may cause collision or grounding.

Santamarta (2014) tested a range of Very Small Aperture Terminal (VSATs) from multiple manufacturers and established that all audited devices are vulnerable at the protocol and implementation level as they transmit in plain text without authentication, encryption, or integrity checks. This can allow an attacker to inject fake signals or malicious code to cause device to shut down or corrupt the system, disabling the ship from navigating safely.

An empirical study by Grottke et al.(2010) based on space mission data analyzed 520 anomalies from the flight software of eighteen JPL space missions and reported that 61% of bugs were Bohrbugs (bugs easily isolated and removed during software testing) and 37% were Mandelbugs (bugs that behave chaotically). Alonso et al. (2013) analyzed the mitigation associated with the Bohrbugs and Mandelbugs. Based on the analysis of bug reports of four open-source software systems, Cotroneo at al. (2013) classified software bugs as Bohrbugs, non-aging-related Mandelbugs, and aging-related bugs.

As shown in Figure 1, software security is a lifecycle issue consisting of three major steps, requirements and acquisition (mission thread, threat analysis and abuse cases), engineering and development (abuse cases, architecture and design principles - A & DP, coding rules and guidelines - C R & G, Testing, validation and verification -T V&V), and deployment and operations (Testing, validation and verification -T V&V, Monitoring, and breach awareness - BA). This figure illustrates that these three steps overlap each other, with the first step overlapping with the second one, and the second one overlapping with both the first and third step.

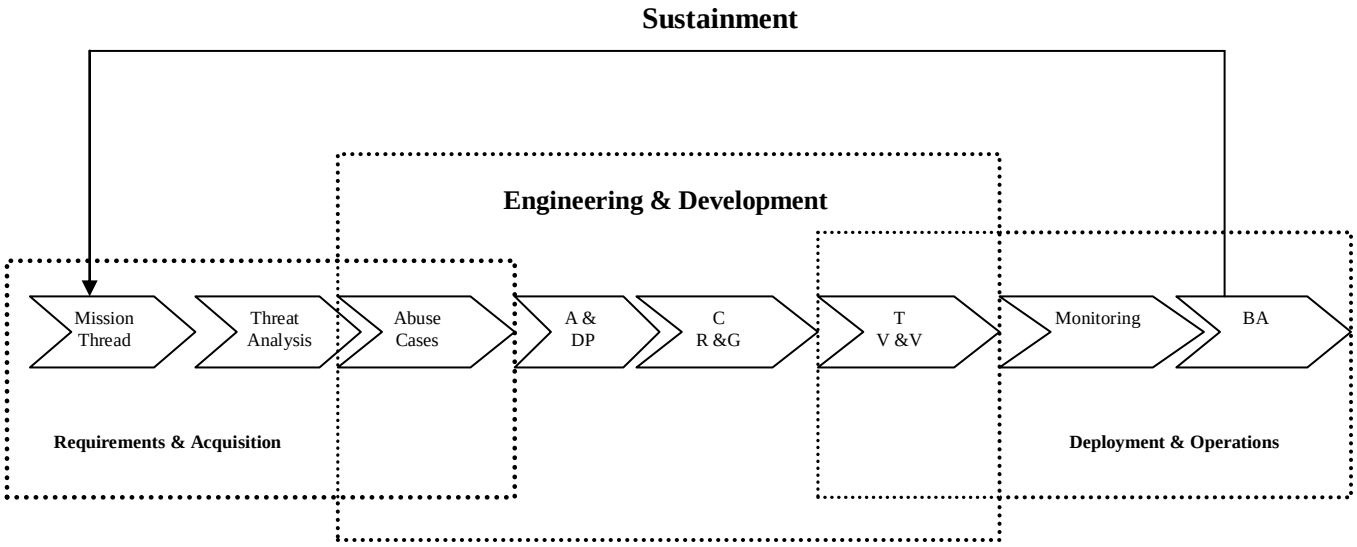

Figure 1: Software Security Life Cycle

However, it was established that 19% of software engineers fail to execute security requirement definition, 27% do not practice secure design, 72% never employ code or binary analysis, 47% never perform acceptance testing for third party code, and more than 81% fail to coordinate their security practices in various stages of the development life cycle as shown in Figure 2.

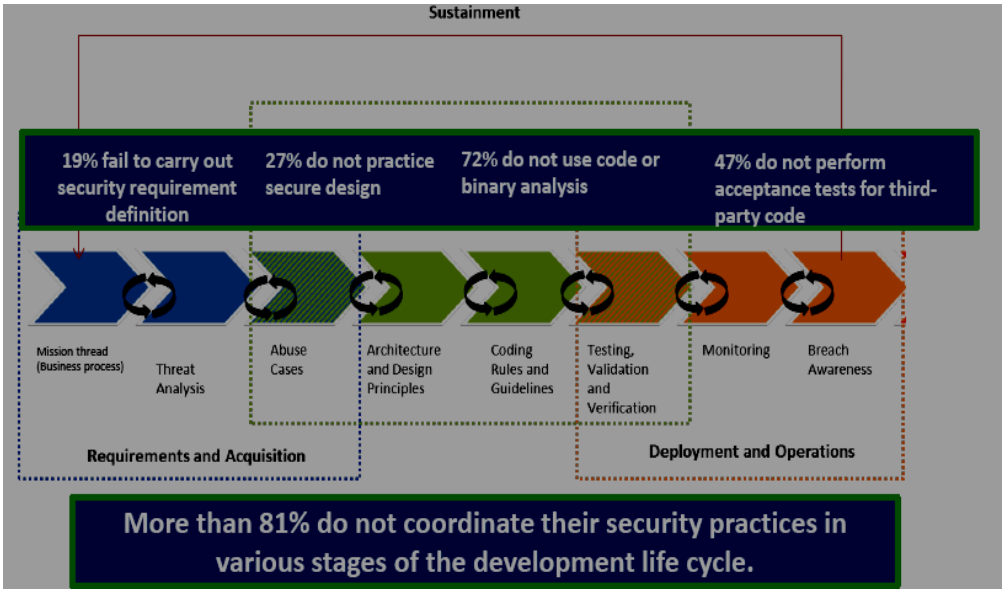

Figure 2: Statistics on Software Security Life Cycle

The rise of open source software is another security challenge as 90% of modern applications were established to be assembled from third party components and at least 75% of organizations depend on open source as the foundation of their applications. Up to 1.8 billion vulnerable open source components had been downloaded in 2015, 26% of which have high risk vulnerabilities

The 2008 explosion of the majority Baku-Tbilisi-Ceyhan pipeline in Turkey was a digital attack in which unidentified hackers infiltrated the pipeline through a wireless network, tampered with the systems and caused substantial physical damage in an explosion. On its part, Stuxnet was a precision attack causing physical damage to Iranian nuclear centrifuges by directing them to spin out of control while simultaneously playing recorded system values that indicated normal functioning centrifuges during the attack. According to Hayden et al., (2014), one of the most touted ICS cyber incidents was the unauthorized release of sewage as the result of malicious operation.

As Kwon, (2015) noted, DarkSeoul that affected 48,000 computers in South Korea disrupted network systems and erased hard disks. It also made attempts to penetrate South Korea's nuclear power plants. Russia-based hackers have been able to cause power blackouts across Ukraine in the first full-fledged attack on an electricity distribution network (Vallance, 2016). The malware was embedded in Microsoft Word documents which once opened, installed itself. Firewalls managed to prevent the attacked computers from gaining control of larger systems. However, an improved version of this malware, BlackEnergy 3, managed to obtain passwords and login details, through which another attack was launched. The attackers remotely logged into SCADA systems, remotely controlled them, cut power at 17 substations, and jammed company communications such that engineers had difficulty gauging the extent of the blackout.

Georgios et al., (2016) explain that developments in the IT world that could spill over to production environments. As such, attacks in the IT infrastructure will find their way into mission critical systems. The OpenSSH implement the secure communications protocol SSH which is then used to securely communicate over the internet. However, the roaming vulnerability, inherent in about 70% of all installed SSH clients facilitates both credential stealing and buffer overflow. The transport layer security (TLS) on its part still supports obsolete, less secure encryption protocols that have enabled DROWN attacks, through which attackers acquire and decrypt session's encryption key. With this knowledge, the whole communication can be decrypted. Domain Name Service (DNS) has been employed to compromise

industrial control installations through DNS Squatting. The Android mobile operating system has been attacked by Triada which infects the OS template utilized to activate applications. As such, it is present in every application that starts to run after the infection. Afterwards, it hijacks and spies normal device operations such as access and control SMS messages. Since it operates in memory, it is intricate to detect by antivirus applications

The Symantec antivirus system was observed to have serious vulnerability in its engine, mainly due to its execution of unpackers, required by the antivirus engines in order to decompress and search into executable code in the kernel. Any successful attack against them would lead to system memory overwriting, which is essentially a buffer overflow, and the subsequent control of the OS.

In his report, Chris (2017) noted that vulnerabilities continue to emerge in previously untested software at alarming rates, with 77% of applications having at least one vulnerability on initial scan, out of which only 14% of very high severity flaws were closed in 30 days or less. Government organizations had the highest prevalence of highly exploitable vulnerabilities such as cross-site scripting (49%) and SQL injection (32%). These vulnerabilities included risky cryptographic practices such as using broken crypto algorithms, improperly validating certificates, storing sensitive information in clear text, and employing inadequate encryption strength.

A number of open source components remain unpatched once built into software, with 88% of Java applications having at least one flaw in a component. Although operations have a role to play in securing production applications, 25% of sites were observed to be running on web servers containing at least one high-severity vulnerability, while 83% of organizations released codes before testing or resolving security issues.

Cloudflare HTML parser designed for improving website performance was found to have exposed one in every 3.3 million HTTP requests. This vulnerability in content delivery network vendor Cloudflare put millions of websites at risk with an information leakage flaw in its software that exposed sensitive data such as passwords, cookies, and authentication cookies for random customers over a five-month period. Panamanian law firm Mossack Fonseca customer-facing website used an old version of SSL that was vulnerable to the DROWN attack, which led to leakage of 11.5 million files and 2.6 TB of secret data. Code quality was observed to eventually impact the security of the application. This includes improper resource shutdown or release, leftover debug code, and using the wrong operator when comparing strings. A zero-day attack, DoubleAgent, took advantage of capability left over in a runtime verification tool in Windows, Microsoft Application Verifier. This tool left open the capability to replace its standard verification execution with a custom verifier that can be injected into any application to give an attacker full remote code execution.

Katerina and Jacob (2017) analyzed data from issue tracking systems of two NASA missions which were organized in three datasets: Ground mission IV and V issues, Flight mission IV and V issues, and Flight mission Developers issues. The results showed that: in IV and V issues datasets the majority of vulnerabilities were code related and were introduced in the implementation phase; for all datasets, close to 90% of the vulnerabilities were located in two to four subsystems; out of 21 primary vulnerability classes, the ones that dominated included exception management, memory access, risky values, and unused entities, which together contributed from around 80% to 90% of vulnerabilities in each dataset. In both the Ground and Flight mission IV and V issues datasets, the majority of security issues, 91% and 85%, respectively, were introduced in the implementation phase. The most security related issues of the Flight mission Developers issues dataset were found during code implementation, build integration, and build verification.

Carriage Return Line Feed (CRLF) injection attack rides on flaws involving improper output neutralization for logs and improper neutralization of CRLF in HTTP headers. Through these flaws, Java and Python applications that poorly filter CRLF have been shown to compromise firewalls. These applications are duped into running rogue FTP connections by using maliciously crafted URLs to trigger unauthorized commands. Mirai malware managed to knock off Twitter, Netflix and GitHub websites through distributed denial of service directed at Dyn, the Domain Name System services provider for those sites. The attack was possible through a botnet of IoT devices using hardcoded passwords. Cross site scripting (XSS) vulnerability in eBay website allowed attackers to embed malicious JavaScript in legitimate listings. This led to their redirection to spoofed eBay login pages that led to the hijacking of eBay accounts, setting off a cascade of costly fraudulent activity on the this site.

The WannaCry ransomware rode on an input validation error in a transport protocol used by Windows machines called Server Message Block (SMB). This vulnerability involved open redirect and unsafe reflection. Encapsulation flaws such as trust boundary violations,

protection mechanism failures, and deserialization of untrusted data have facilitated ransomware attack against the San Francisco Metropolitan Transit Agency's Municipal Rail. Specifically, the attack exploited a Java deserialization flaw.

V. Current State of Software Security Models

As Erlingsson (2016) explains, security models that enable simple, useful security policies for large classes of software are ideal. However, their simplicity prevents them from addressing many real-world security concerns. An ideal security model that admits simple policies is the one that includes mechanisms to thwart the exploits of low-level software vulnerabilities.

Stack-based buffer overflows and memory-corruption vulnerabilities have become a primary exploit vector and a critical software security issue. In defending against such attacks, the security model, programmer intent software security, has been particularly effective at defining simple, useful security policies that successfully prevent exploits. This model permits only low-level executions that programmers intended to be possible, unless given explicit, special permission. Here, security policies are automatically derived from software source code or binaries by identifying simple program properties that are obviously true based on the programming-language abstractions and semantics and the clear intent of the programmers.

According to Tice et al., (2014), examples of this security model is the enforcement of the programmer's intended control and data flow, termed as Control-Flow Integrity (CFI) and Data-Flow Integrity. The policies in this model greatly constrain the attacker from exploiting low-level vulnerabilities. Unfortunately, many of these instantiations tie policy and mechanism too closely and intricately together for the underlying model to be clearly identifiable. In addition, this model leaves other vulnerabilities such as actual logic errors made by programmers.

Another software security model is the permit only executions that historical evidence shows to be common enough, unless given explicit, special permission. This model prohibits all novel security-relevant behavior, unless especially permitted and in so doing, prevent many software attacks, such as privilege-escalation exploits of the vulnerabilities regularly discovered in esoteric operating system services.

As Erlingsson (2016) point out, data-driven security model is a new attractive model that is natural basis for software security enforcement. It considers how a software has behaved in the past, and can be naturally combined with existing security models by simply ensuring that operations proceed and information flows in accordance with historical audit logs as shown in Figure 3 and Figure 4.

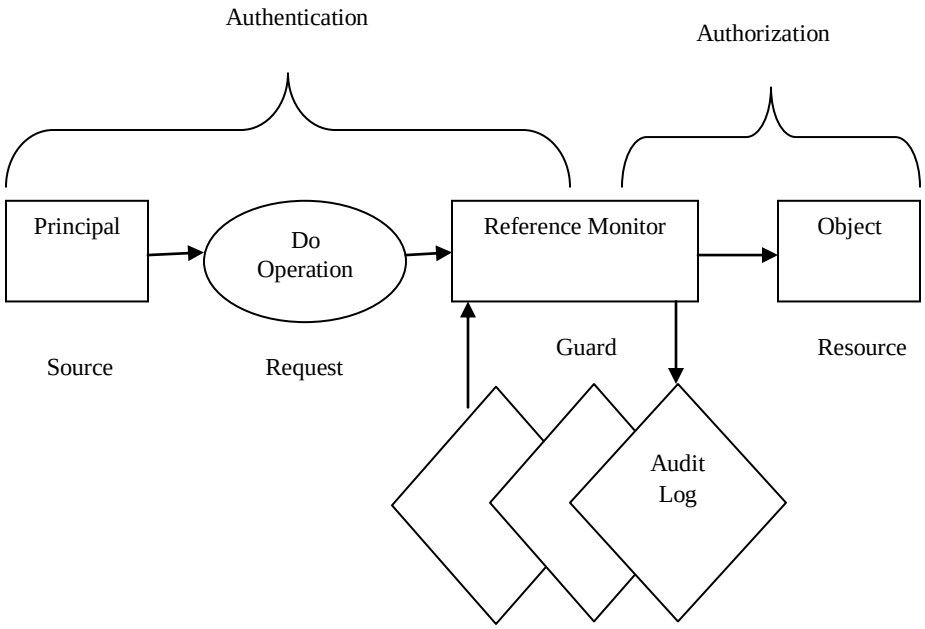

Figure 3: Data-Driven Software Security Model Tied to Access Control

The primary parameters of data-driven security policies are event abstraction employed in execution traces, such as network service requests, API or system calls, function calls, or simply security privileges as well as the historical frequency by which security-relevant events must have been seen, to be permitted in the current execution.

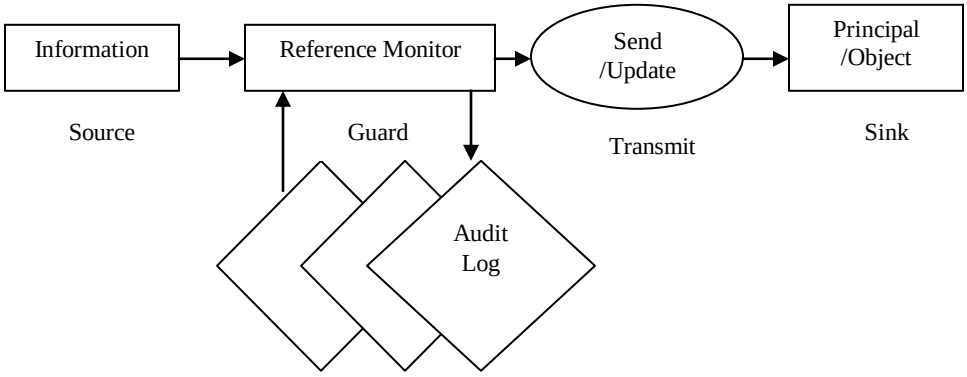

Figure 4: Data-Driven Software Security Model Tied to Information Flow

An event is considered to be supported by historical evidence if it has occurred at least once (or k times for some fixed, low threshold k) in the execution traces; otherwise, it is prohibited by the security policy. Such policies are employed in network firewalls and operating-system sandboxing. They are also ideal in large-scale, popular software applications, which are normally big in size, composed of innumerable platforms, modules, and libraries, and full of arcane or unused functionality. Since this software have various vulnerabilities as well as embedded interpreters, dynamic-library loaders, and reflection APIs that attackers can exploit to perform arbitrary behavior, by simply disallowing previously unseen security-relevant events, such attacks can effectively be thwarted.

Thedata-driven software security model relies on the empirical program abstraction to avoid such falsely-reported security violations. Here, empirical programs include all execution traces, not just those from training runs. These traces also encompass all executions performed during the software's development and testing, which ensures that any latent, actual software feature is represented, even for the first use of unpopular software. Moreover, data-driven security techniques are partially integrated into engineering processes and preferably used throughout the software development lifecycle. This software engineering integration is helpful in maintaining security policies as software is updated for security, stability, or behavior.

The results of the data analysis have been combined with the source code examination to develop finite state machine (FSM) models that can be used to reason about security vulnerabilities. In a closely related work, the analysis of 107 CERT advisories showed that vulnerabilities of the following four types dominated: buffer overflow (44%), integer overflow (6%), heap corruption (8%), and format-string vulnerabilities (7%).

VI. Current Mission Critical Software Security Countermeasures

In the maritime environment, some strategic directions towards securing cyber technology on ships include defense –in-depth, security policies and procedures, and technical security solutions. In-depth includes policies, physical security, perimeter security, network security, application security and data security. Technical solutions include firewalls and intrusion prevention systems that monitor and block the data traffic as it leaves and enters the ship's IT network. The dataflow between all nodes on the network including ICS traffic and satellite and radio communications should be mapped out and encrypted, for instance by using VPN.

Firewalls, routers, switches, servers, voice communication equipment, and any other device on the network should be network hardened, which involves secure configuration of hardware and software and the deactivation of unused features and accounts. The usage of secure communications protocols like SSH, HTTPS, and SFTP is another way of protecting mission critical infrastructure. Multi-Factor Authentication (MFA) offer an additional layer of access security to sensitive systems and applications while application white-listing prevents staff from installing unapproved and potentially malicious programs. As Mertens (2014) explains, data-loss-prevention software can mitigate the threat of intentional or accidental data leakage. Figure 5 presents a threatanalysis tools that can help derive abuse and misuse cases.

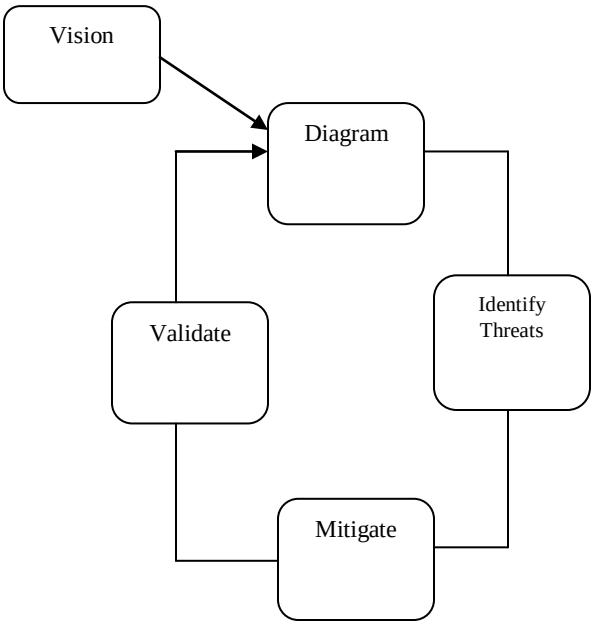

Figure 5: Microsoft SDL Threat Modeling Tool

Since Katerina and Jacob (2017) report showed that code related security issues dominated both the Ground and Flight mission IV and V security issues, with 95% and 92%, respectively, enforcing secure coding practices, verification and validation focused on coding errors are cost effective methods of improving critical missions' security.

VII. Challenges of Current Software Security Protection

In their paper, Johansson et al., (2009) point out that conventional IT security techniques can be applied to protect a CPS such as a critical infrastructure system against cyber threats or threats imposed by malicious insiders. Unfortunately, due to the unique features of a CPS, these security strategies and approaches are not sufficient enough to address the security challenges of a CPS. For instance, installing security patches or numerous system updates requiring the taking of the system offline is not economically justifiable, difficult, and infeasible. In addition, new updates or security patches may create other challenges that may a nuclear power plant to accidentally shutdown after a software update.

Another challenge is that there is a lack of a framework for assessing the security in designing a CPS or evaluating the level of the security guarantee in a functional CPS at the design level. Consequently, it has been demonstrated that attackers can take control of air planes by having access to Wi-Fi services provided by the planes. In addition, most approaches for securing CPS consider the security of individual components of the CPS such as sensors, PLCs, actuators, or communication protocols. Using this isolation, they then adopt standard practices to secure individual components against security threats such as input validation or firmware tampering. This is insufficient as CPS can be attacked by compromising the interaction between components without hacking the individual components within a CPS. By changing the interaction of components, attackers create different outputs than what was requested by the operators. As an illustration, adversaries can cause delays in transferring the information from sensors to SCADA, activating unwanted actions imposed by the delay in receiving the requested results.

Unified security enforcement mechanism is infeasible in mission critical infrastructure since these systems employ different devices from different vendors. The conventional approaches for safety analysis in CPS include Fault Tree Analysis (FTA), Failure Mode and Effects Analysis (FMEA), Hazard Analysis and Critical Control Points (HACCP), and Hazard and Operability Study (HAZOP). All these approaches are based on risk assessment and risk analysis of a system and none of them is geared towards addressing the threats that compromise the interactions among components in a CPS. This is because all of them consider individual components or subsystems in isolation in addressing the safety of a CPS.

It is noted that these approaches were designed for safety analysis and hence cannot be used to effectively address the security concerns in a CPS. Safety and security are different in nature as a system may be safe but not secure. As an illustration, a system can permit unauthorized modifications of the control parameters within the safe range without being detected by system safety controllers, creating undesirable output that was not requested by the operator.

In the software security domain, methods such as Microsoft's STRIDE/DREAD or attack tree have been developed for threat and vulnerability analysis. Unfortunately, the application of these methods to analysis of the security and safety-related incidents in CPS fails to consider the interactions among different components as well as that of the control loops. The current practice of constructing models for Mission Impact Assessment (MIA) is accomplished manually, making model construction very time consuming, expensive, difficult to document, to inspect and to validate.

The security approaches for IT systems do not cover embedded system security as virus definitions and

operating guidelines do not always apply, firewalls and IDS or IPS are of limited value and centralized account control is not possible. This is because network tools and assessment techniques are unaware of embedded systems architecture and interfaces such as unique and insecure protocols, maintenance backdoors, hard-coded credentials, unique architectures of embedded controllers, and unplanned connectivity and upgrades. The application of the data driven model in a practical security enforcement mechanism is infeasible owing to the challenges in the selection of the security policy to be enforced on the first program execution.

As Arman et al., (2018) discuss, architects need to understand the effects of the decisions on the final system in order to make effective design decisions. Regrettably, current assessment approaches for such systems rely on static or dynamic analysis of system models. Static analysis techniques call for the development of complicated mathematical models which require steep learning curves and significant modeling effort, limiting the resulting system's scalability. Esfahani et al., (2013) further explain that based on the mathematical models they rely on, these techniques are confined to particular category of software system models, or are heavily dependent on error-prone and sometimes inaccessible expert inputs.

On the other hand, architectural modes based on dynamic analysis techniques are capable of capturing the randomness reflective of reality and are ideal in constructing models tailored to the task at hand. However, as Langhammer et al., (2016) point out, these models have false negatives and longer execution times. In addition, Aleti et al., (2013) explain that simulations of software architectural models have not been extensively utilized compared with static analyses since creating simulatable system designs is intricate, and running simulations on multifarious models is time consuming and requires explicitly addressing scalability issues. Me et al., (2016) states that trade-offs in system properties caused by design decisions complicate quantitative assessment while Shahbazian et al., (2016) elaborate that analysis of system behavior requires massive datasets.

VIII. Proposed Software Security Models

In their paper, Luigi et al., (2017) explain that defense software development process is aa complicated activity due to the complex domain, requirement for a very high quality solution and the need to satisfy very specific and complex needs. Specifically, the command and control software has to satisfy functional requirements that are extremely detailed as the situations that create them involve a large number of assets and human resources that have to be coordinated in a way that minimizes losses, as any trade-offs may result in the loss of lives.

Mission critical software in any Army software must meet integration, safety, security, real time response and be reliable. Integration enables the software to operate in highly interdependent ecosystem. For instance, command and control software must allow seamless integration with communications software to enable inter-force communication. Security is crucial during the development of armed forces software to prevent unwanted intrusions and cyber attacks. Here, vulnerabilities in systems are simply unacceptable. Safety measures are significant to thwart inadvertent authorization of measures which could result in severe consequences. Real time response is critical to adapt to an evolving situation. Reliability is important in order to operate in hazardous conditions, in a disruptive environment with a high level of dependability and robustness.

Critical control networks should be in a secured zone such as DMZ to isolate them from the corporate IT network and the internet. Since CPSs are complex, a system-theoretic approach that takes into consideration the system complexity should be adopted to address the security of a multifaceted CPS at the design level. This enables the identification of vulnerable points, subsystem interactions and their effects on vulnerable points and provides recommendations on how to increase the security of a CPS. Game- theoretic approaches are suggested here to account for the highly adversarial nature of cyber operations.

Team Software Process (TSP) is also suggested in mission critical infrastructure software development. It instills engineering discipline in software developers, and builds high-performance trusted teams. Extending TSP with security ensures safe design that minimizes attack surfaces, ensures defense in depth for software development, assures secure coding, provides tools for supporting automated conformance checking, tracks security defects, and monitors results of tests with respect to security. The software supply chain need to keep risk factors to some acceptable levels as illustrated in Figure 6.

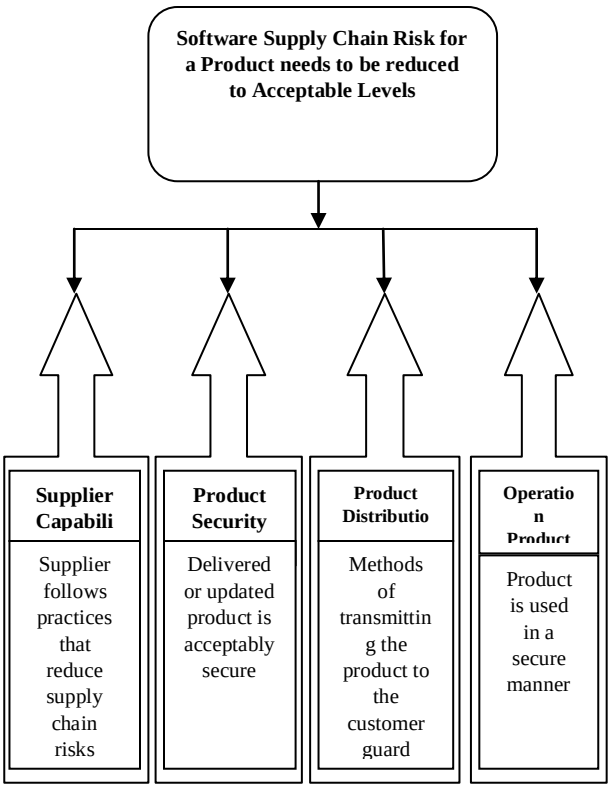

Figure 6: Secure Software Supply Chain Model

The software source code needs to be tested by performing rule conformance checking, thread safety analysis, information flows across applications, and operating system call flows. The industrial internet of things (IIoT) should apply non-invasive techniques to patch remote assets, and use industrial control and automation systems that cannot easily be shut down. Obsolete and legacy operating systems, hosts and devices that have limited or no security built into them should be properly managed. In addition, network connections should be monitored and controlled to ensure that only appropriate ones exist between sensitive industrial equipment. The software should have inbuilt fail-safe mechanisms to ensure that compromised systems that run ICSs cause no physical harm to people and property, or other severe consequences.

Mobile devices from any manufacturer must be controlled centrally using an Enterprise Mobile Device Management (MDM) that restricts the functionality of such devices to the lowest necessary to execute their legitimate purposes, ensures that the devices have updated operating system, installed antivirus, control and filter web sites visited, restrict files downloaded, centrally and control software that can be installed. Policies and practical technical controls should be put in place to prevent users from connecting their own devices to Enterprise equipment as any of these devices can be part of an attack vector to mission critical systems.

Figure 7 shows the proposed security requirements the cyber physical systems. As this figure demonstrates, there are five security aspects, namely the sensing security, communication security, actuation control security, storage security, and feedback security. Sensing security ensures validity and accuracy of the sensing process while communication security protects both inter- and intra-CPS communication from both active and passive attackers. Actuation control security ensures that no activation occurs without appropriate authorization, while storage security thwarts both cyber and physical tampering of any data stored by the CPS. The feedback security on its part protects the control systems in a CPS which provide the necessary feedback for effecting actuation.

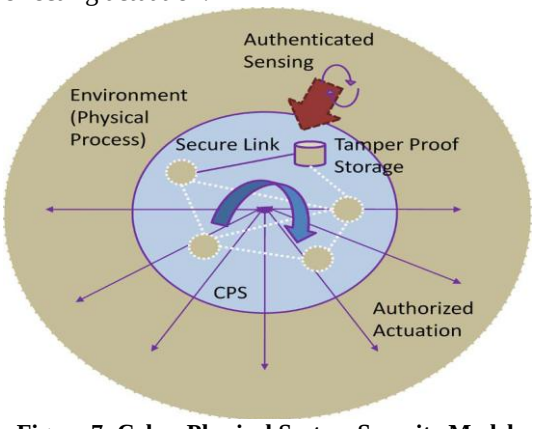

Figure 7: Cyber Physical System Security Model

The deployment of anti-malware software throughout the ICS environment where possible, is also proposed. The usage of a bastion host to avert unauthorized access to protected locations in the ICS environment, application of application white-listing to thwart the execution of unauthorized applications, deployment of a breach detection system, and configuring USB lockdown on all SCADA environments are suggested.

IX. Conclusions

This paper has investigated software security issues in mission critical systems from which a number of security issues that expose these systems to attacks have been discussed. Many weak spots in mission critical infrastructure such as ship and shore-based cyber systems have been observed. Failure to identify these vulnerabilities has led many entities into taking shortcuts in regard to applying and policing appropriate security measures. In addition, it has been established

that rapid cycles of software product development, implementation, maintenance, and decommissioning are overwhelming for the majority of these systems. Based on these challenges, rafts of measures and techniques that can be deployed to protect these systems have been proposed. Future work lies on the practical implementation of these measures into mission critical software development life cycle and the actual infrastructure.

References

[1] Sebastian B., & Stephan D. (2018).Towards a Theory of Software Development Expertise. In Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Pp. 1-14.

[2] Dennis B., Guanglou Z., & Craig V. (2017). A critical analysis of security vulnerabilities and countermeasures in a smart ship system. Proceedings of the 15th Australian Information Security Management Conference. Pp.81-87.

[3] Hambling D. (2017). Ships fooled in GPS spoofing attack suggest Russian cyberweapon.

[4] National Institute of Standards and Technology. (2017). Framework for Improving Critical Infrastructure Cybersecurity. Draft Version 1.1.

[5] Hudson Analytix Inc. (2017). Global Threats: Cybersecurity in Ports (Donald Duck, Daughters & Dollars. Hemispheric Conference on Port Competitiveness & Security: Finding the Right Balance, University of Miami, Center for International; Business Education & Research.

[6] Symantec Security Response. (2017). Petya ransomware outbreak: Here's what you need to know.

[7] Santamarta R. (2014). SATCOM terminals: Hacking by air, sea, and land.

[8] Mertens M. (2014). Securing VSAT Terminals.

[9] Arash N., & Stuart M. (2015). A Systems Theoretic Approach to the Security Threats in Cyber Physical Systems Applied to Stuxnet. IEEE Transactions On Dependable And Secure Computing. Pp. 1-20.

[10] Johansson E., Sommestad T., &Ekstedt M.(2009). Issues of cyber security in scada-systems-on the importance of awareness. In 20th International Conference and Exhibition onElectricity Distribution-Part 1, IET. Pp. 1–4.

[11] Motzek A., &Möller R. (2017). Context- and bias-free probabilistic mission. Computers & Security. Vol. 65, pp. 166-186.

[12] Lange M., Kuhr F., &Möller R. (2016). Using a Deep Understanding of Network Activities for Network Vulnerability Assessment. In Proceedings of the 1st International Workshop on AI for Privacy and Security.

[13] Noel S., Ludwig J., Jain P., Johnson D., Thomas R., McFarland J., King B., Webster S., &Tello B.(2016). Analyzing Mission Impacts of Cyber Actions. In NATO IST-128 Workshop on Cyber Attack Detection, Forensics and Attribution for Assessment of Mission Impact, Istanbul.

[14] Mark S. (2017). Building Secure Software for Mission Critical Systems. Software Solutions Symposium. Pp. 1-50.

[15] Tobby S. (2017). Critical Infrastructure and the Internet of Things.Centre for International Governance Innovation and Chatham House. Pp. 1-20.

[16] Daugherty, Paul, Prith Banerjee, WalidNegm and Allan E. Alter. 2015. "Driving Unconventional Growth through the Industrial Internet of Things." Accenture.

[17] Hayden E., Michael A., & Tim C. (2014). An Abbreviated History of Automation & Industrial Controls Systems and Cybersecurity. A SANS Analyst Whitepaper.

[18] Kwon J. 2015. Smoking Gun: South Korea Uncovers Northern Rival's Hacking Codes. CNN.

[19] Vallance C. 2016. Ukraine cyber-attacks 'could happen to UK. BBC.com

[20] Randy H., & Susan S.(2016). Secure Software Engineering Best Practices. NSF Cybersecurity Summit. Pp. 1-140.

[21] Georgios K., Georgios G., &Athina M. (2016). Cyber Security Trends and their implications in ICS. JRC Technical Reports.Pp.1-28.

[22] Erlingsson U. (2016). Data-driven Software Security: Models and Methods. ArXiv. Pp.1-7.

[23] Tice C., Roeder T., Collingbourne P., Checkoway S., Erlingsson U., Lozano L., and Pike G.(2014). Enforcing forward-edge control-flow integrity in GCC & LLVM. In Proceedings of the 23rd USENIX Conference on Security Symposium, ser. SEC'14. Pp. 941–955.

[24] Katerina G., & Jacob T. (2017). Experience Report: Security Vulnerability Profiles of Mission Critical Software: Empirical Analysis of Security Related Bug Reports. IEEE 28th International Symposium on Software Reliability Engineering. Pp. 152-163.

[25] Alonso J., Grottke M., Nikora A., & Trivedi K. (2013). An empirical investigation of fault repairs and mitigations in space mission system software. In 43rd IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Pp. 1–8.

[26] Cotroneo D., Grottke M., Natella R., Pietrantuono R., & Trivedi K. (2013). Fault triggers in open-source software: An experience report. In 24th IEEE International Symposium on Software Reliability Engineering (ISSRE). Pp. 178–187.

[27] Grottke M., Nikora A., & Trivedi K. (2010). An empirical investigation of fault types in space mission system software. In 40th IEEE/IFIP International Conference on Dependable Systems Networks (DSN). Pp. 447–456.

[28] Di Martino C., Kalbarczyk Z., Iyer R., Baccanico F., Fullop J., & Kramer J.(2014). Lessons learned from the analysis of system failures at petascale: The case of Blue Waters. In 44th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Pp. 610–621.

[29] Luigi B., Angelo M., & Alberto S. (2017). iAgile: Mission Critical Military Software Development. International Conference on High Performance Computing & Simulation. Pp. 545- 552.

[30] Chris W. (2017). The State Of Software Security Today. Veracode. Pp. 1-44.

[31] Arman S., Youn K., Yuriy B., Nenad M. (2018). Poster: MakingWell-Informed Software Design Decisions. ACM/IEEE 40th International Conference on Software Engineering: Companion Proceedings. Pp. 262-263.

[32] Esfahani N., Malek S., &Razavi K. (2013). GuideArch: Guiding the exploration of architectural solution space under uncertainty. In International Conference on Software Engineering (ICSE). Pp. 43–52.

[33] Langhammer M., Shahbazian A., Medvidovic N., and Reussner R. (2016). Automated extraction of rich software models from limited system information. In IEEE/IFIP Working Conference on Software Architecture (WICSA).Pp. 99–108.

[34] Aleti A., Buhnova B., Grunske L., Koziolek A., and Meedeniya I. (2013). Software architecture optimization methods: A systematic literature review. IEEE Transactions on Software Engineering (TSE).Vol. 39, Issue 5, pp.658–683.

[35] Me G., Calero C., and Lago P.(2016). Architectural patterns and quality attributes interaction. In IEEE Workshop on Qualitative Reasoning about Software Architectures (QRASA). IEEE.

[36] Shahbazian A., Edwards G., and Medvidovic N. (2016). An end-to-end domain specific modeling and analysis platform. In Proceedings of the 8th International Workshop on Modeling in Software Engineering, ACM. Vol.16, pp. 8–12.