Vulnerability Management in IIoT-Based Systems: What, Why and How

Geeta Yadav, Kolin Paul, and Praveen Gauravaram

Abstract Industrial Control Systems (ICS) are characterized by large numbers of tightly integrated, interdependent, and heterogeneous components in a network. They act as a base system for safety and mission-critical Industrial Internet of Things (IIoT) applications such as smart grids, nuclear power plants, process control systems and robotics systems. The complex ICS, e.g., Supervisory Control and Data Acquisition (SCADA), consists of many interdependent subsystems. Modern SCADA systems are an amalgam of IIoT and legacy systems. IIoT is essentially a realization of advances in the connectivity of hardware and data networks that SCADA provides. Therefore, modern SCADA has evolved as a use case of IIoT, wherein IIoT improves industrial productivity by analyzing data generated by SCADA systems. The modernization of the SCADA system, standardization of communication protocols and almost ubiquitous interconnectivity courtesy for IIoT has drastically increased the attack surface of the SCADA system. Systematic Vulnerability Management (VM) of these attack surfaces minimizes risks and impacts associated with vulnerability exploitation. In this chapter, we first find the correlation between the IIoT and SCADA systems, followed by security challenges faced by IIoT-based systems. Then we highlight the role of VM in securing the critical systems, followed by the study of the state-of-art approaches for VM. After that, we discuss some future research directions for developing techniques for efficient VM. The chapter underscores the design challenges and research opportunities for efficiently managing the increasing vulnerabilities.

G. Yadav (✉) · K. Paul
Indian Institute of Technology Delhi, New Delhi, India
e-mail: geeta@cse.iitd.ac.in

P. Gauravaram
Tata Consultancy Services (TCS), Brisbane, Australia

S. Pal et al. (eds.), *Secure and Trusted Cyber Physical Systems*, Smart Sensors, Measurement and Instrumentation 43, https://doi.org/10.1007/978-3-031-08270-2_3

1 Introduction

Over the years, an increase in the number of cyberattacks targeting the Industrial Control Systems (ICS) such as Supervisory Control and Data Acquisition (SCADA) systems have drawn the security researchers' attention towards these system's securities. There are several real-world cyber attacks on ICS infrastructures as discussed below.

Ransomware on US fuel pipeline In 2021, a ransomware attack encrypted critical data of Information Servers used in the SCADA stack of US Colonial Pipeline company [45]. As a consequence, Colonial pipeline company suspended all of the pipeline's operations as a precaution and to prevent further cascading impact. The adversaries stole nearly 100 gigabytes of data which led the company to pay 75 Bitcoin ($ 5 Million) to get the decryption tool due to a single compromised password. The pipeline shutdown impacted fuel shortages at airports and filling stations, resulted in canceling flights and panic fuel buying.

Polish airline attack [29] was due to Distributed Denial of Service (DDoS) attack, which overwhelms a network with traffic. The security expert took five hours to resolve the issue, leading to 10 flights cancellation and delays of around 15 flights at Warsaw Chopin airport.

The digital cyber-weapon **Stuxnet** [19] targeted at SCADA systems in 2010 is considered to be the most sophisticated cyber-attack. A malware jumped across air-gapped networks and damaged nuclear centrifuges of Iranian enrichment plants exploiting four unpatched zero-day Microsoft vulnerabilities used for self-replication and privilege escalation. Stuxnet damaged the centrifuges used in the uranium enrichment process by modifying their rotor speed. Vibrations and distortions caused by significant and sudden changes in their speed destroyed a thousand centrifuges, leading to less enriched uranium production.

Ukraine power grid attack [19, 46] in December 2015, where hackers hacked the information systems of three energy distribution companies using BlackEnergy malware. It resulted in rolling power outages for 1–6 h and affected 225,000 users.

In **German Steel Plant cyberattack** [28], the attackers gained unauthorized access to the mill's control systems using spear-phishing social engineering attacks. It led to an abnormal and unscheduled shutdown of the furnace, resulting in massive physical damage to the steel plant.

These incidents demonstrate the impact of a cyberattack by a determined adversary on such Critical Infrastructure (CI). Such cyberattacks could affect the availability of the software running on the device or can be used to reveal the running application's secrets. Devices under attack could stop working, behave differently, or be

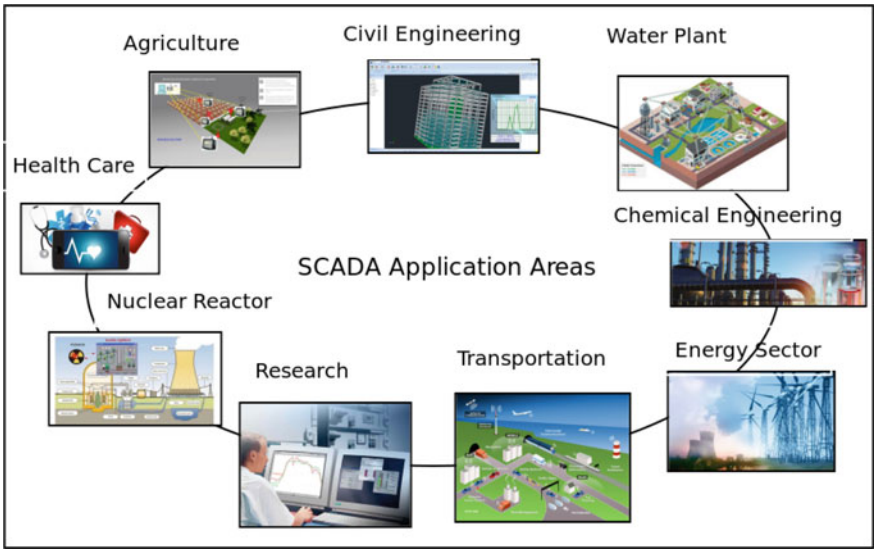

Fig. 1 SCADA application areas [60]

leveraged to pose DDoS attacks either exploiting zero-day or reported yet unpatched vulnerabilities in a system. Moreover, these attacks have been led due to vulnerable SCADA systems by exploiting multiple vulnerabilities on different systems, generally referred to as Multi-host Multi-stage (MhMs) cyberattacks. SCADA systems, a type of ICS, are characterized by large numbers of tightly integrated, interdependent and heterogeneous components in a network [32]. The smooth and genuine operation of the SCADA framework is one of the key concerns for enterprises because the outcome of the breakdown of the SCADA system may range from financial loss to environmental damage to loss of human life [12]. These systems act as the base for safety and mission-critical infrastructures such as smart grids, nuclear power plants, process control systems and robotics systems [60]. These systems have become an essential part of automated control and monitoring of CI such as agriculture, healthcare, nuclear reactor, transportation, energy sector, civil and chemical engineering, water plants, research etc., as depicted in Fig. 1. Considering the significance of SCADA and ICS security that underpin critical national infrastructure, US Government offered policy recommendations for synchronizing foreign and domestic cyber security efforts and realizing a resilient and secure infrastructure [54].

Evolution of SCADA systems: Modern SCADA systems have evolved from standalone systems into sophisticated, complex and open systems connected to the Internet. With Industry 4.0/Industrial Internet of Things (IIoT) evolution, modern SCADA systems have adopted Cyber-Physical System (CPS)/IIoT, cloud technology, big data analytics, artificial intelligence and Machine Learning (ML). IIoT, generally defined as a sub-set of the Internet of Things (IoT) in terms of usage, covers the domains

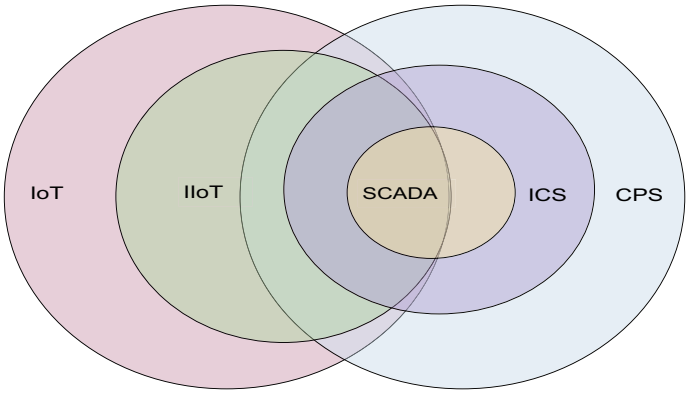

Fig. 2 IIoT and SCADA

of machine-to-machine and industrial communication technologies with automation applications. IIoT paves the way for a better understanding of the manufacturing process, enabling efficient and sustainable production. IIoT allows a higher degree of automation by using cloud computing and data analytics to refine and optimize the process controls [9]. It further enables efficient interaction between the physical world and the cyber world, usually addressed as a CPS. ICS is the critical component to realize CPS. ICS provides control and monitoring functionality in manufacturing and industries.

Correlation of IIoT and SCADA systems: In Fig. 2, we demonstrate the overlap of IoT, IIoT, SCADA, ICS and CPS systems. IIoT is a subset of IoT. ICS such as SCADA is used to control CPS. Modern SCADA has been evolved into a connected IIoT-based system i.e., modern SCADA systems are an amalgam of IIoT and legacy systems as shown in Fig. 3. IIoT is essentially a realization of advances in the connectivity of hardware and data networks that SCADA provides. From the security perspective, the differences between them is not important. Therefore, in this chapter, we consider SCADA systems as a use-case for IIoT-based systems. We use IIoT-based SCADA systems and IIoT-based systems interchangeably.

In brief, integrating these technologies has significantly improved interoperability, eased maintenance and decreased the infrastructure cost. Therefore, modern SCADA systems are leading to a near real-time environment. Although IIoT improves the reachability in ICS, enhances data analytics, assuring ease of access and decision making, it also opens the ICS environment to attackers [14, 60]. The design of IIoT-based SCADA introduces multiple entry points to an isolated system, which is used to protect itself via air-gapping and risk avoidance strategies.

The Confidentiality, Integrity, and Availability (CIA) triad security model provides an excellent way to demonstrate the best practices to protect the data on the network. For the SCADA system, the security goal is generally the data availability that is the reverse of the prioritized security goals for traditional Information Tech-

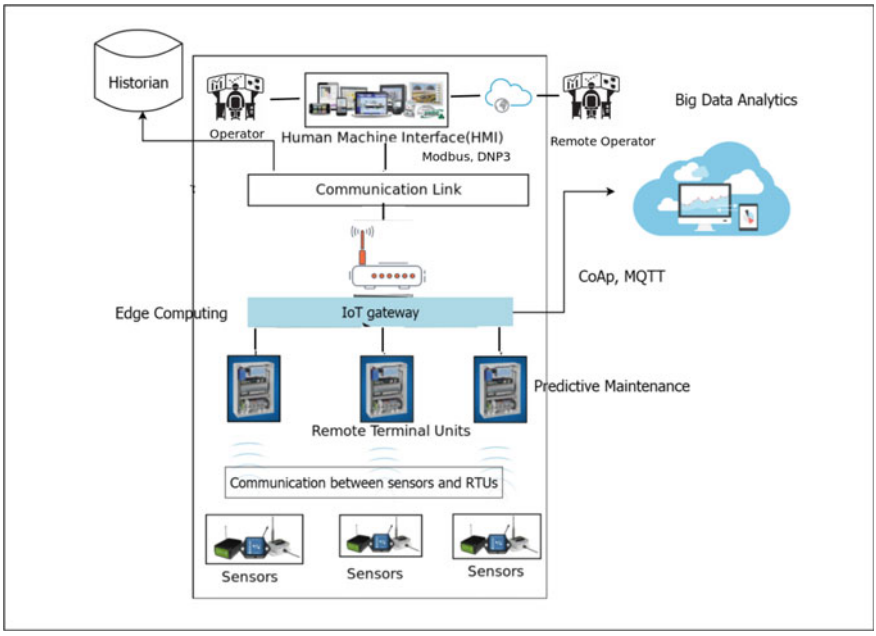

Fig. 3 IIoT-based SCADA [60]

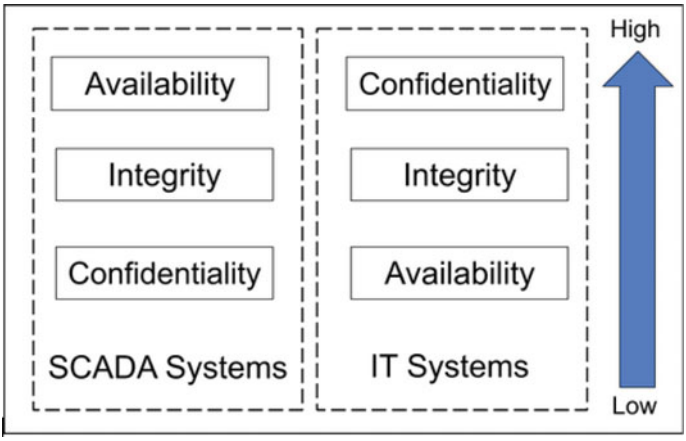

Fig. 4 Priority order for SCADA and general IT

nology (IT) systems, as shown in Fig. 4. Therefore, downtime-constraints security is considered while implementing IIoT-security solutions. The ICSs are also called Operational Technology (OT) devices that control the physical world, while IT systems manage data [6]. Therefore, attackers generally target interrupting the SCADA system availability, causing production loss, financial loss, data loss, system dam-

age, etc., hence tremendously affecting the economy, safety and security of a nation. An attacker needs to think outside the normal operating procedures to discover the unusual behavior, thus identifying vulnerabilities resulting in unauthorized access. The attacker needs only a single security hole, while a defender must defend against all possible security holes. Therefore, the defender needs to be more competent to compete with an attacker. Developing rigorous security layers can help to minimize the impact of attacks. A large number of vulnerabilities in various domains are reported to National Vulnerability Database (NVD) [38] each year. In NVD, 18,103 new vulnerabilities were reported in 2020 itself. With the integration of IIoT and legacy SCADA, the vulnerabilities reported to other domains are also applicable to IIoT-based SCADA [53], in a characterization study of ICS patching behavior, observed a patch delay of approximately 60 days after vulnerability disclosure for 50% of ICS devices. This lack of intime patching gives adversaries ample time to exploit these systems' publicly disclosed vulnerabilities.

Hence, the management of ICS security is becoming a major prevalent challenge due to an increase in system complexity and interdependencies. The progressive nature of ICS further complicates the scenario. On the one hand, the increasing complexity of software usually translates into more software flaws and vulnerabilities to fix. On the other hand, system threats continuously evolve, changing the risk outlook as new vulnerabilities and attack vectors emerge. In brief, to minimize the potential impact of successful cyberattacks, Vulnerability Management (VM) plays a pivotal point in any strategy for system security management.

In this chapter, we highlight the role of VM in securing critical systems, followed by the study of the state-of-art approaches for VM in Sect. 2. After that, we figure out the future research direction for developing techniques for efficient VM in Sect. 4. The chapter concludes by underscoring the design challenges and research opportunities for efficiently managing the increasing vulnerabilities in Sect. 5.

2 Vulnerability Management

VM is an indispensable part of managing an organization's safety and security. VM allows an organization to get a continuous overview of vulnerabilities in their OT environment. It is generally characterized as a cyclical process of five stages, i.e., Vulnerability discovery, Vulnerability analysis, Vulnerability prioritization, Vulnerability remediation and Vulnerability verification and monitoring.

What is VM? Strategic vulnerability management reduces the risk associated with vulnerability exploitation. In a generic term, VM tries to answer the following questions:

1. Do vulnerabilities exist on organizations' assets? If yes, what are they?
2. What are the characteristics of the discovered vulnerabilities?

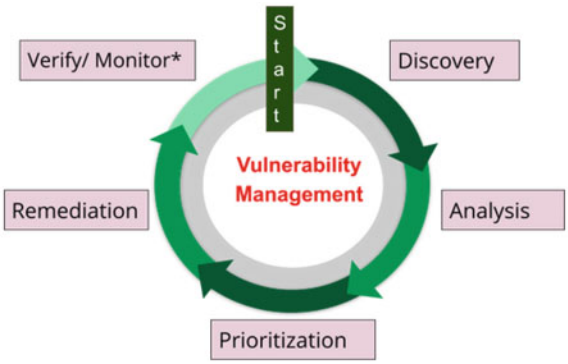

Fig. 5 Generic vulnerability management lifecycle (*Extended stage)

3. What are the efficient strategies to fix the vulnerabilities so that the vulnerability exploitation's impact is minimal? Is there a critical need to patch all vulnerabilities?
4. What are the mechanisms for efficient and safe patch deployment?
5. Are the systems working normally post-patch deployment? Also, what vulnerabilities can not be patched yet have high risk? What are the monitoring strategies for unpatched vulnerabilities?

How is VM performed? VM is a cyclical practice of discovering, analyzing, prioritizing, remediating and verifying/monitoring possible exploitation of vulnerabilities in operating systems (OSs), enterprise applications, browsers and end-user applications, as shown in Fig. 5. In the first step, the vulnerabilities are generally discovered using a vulnerability scanner such as Nessus [50] and Nozomi networks [39]. Then in the second stage, the vulnerability scanner generates a consolidated report of possible known vulnerabilities. The security experts analyzed the report to prioritize the vulnerabilities based on their expertise and network knowledge in the third stage. The high severity vulnerabilities are selected for patching and the respective patch is deployed in the fourth stage. Once the vulnerabilities have been identified and resolved, consistent follow-up audits are required to ensure the mitigation is working in the fifth stage. This stage of vulnerability management is called the verification stage that helps to maintain transparency and accountability over the remediation process. Further, there can be two scenarios (i) the patch[1] is not available, (ii) a patch can not be applied to the system due to resource constraints or availability requirements. This gives an adversary ample time to exploit those vulnerabilities. Therefore, it is highly recommended to monitor the system to detect ongoing exploitation on time to minimize the potential damage. We extend the standard VM cycle by monitoring such a set of vulnerabilities in the fifth stage.

Why is VM needed? In brief, the lack of an appropriate plan for cyber-securing the assets in IIoT-based SCADA can cause organizations to have high risks of losing

[1] A security patch is applied to the system to fix the vulnerability to prevent successful exploitations.

revenue and reputation. VM is crucial to prioritize possible threats, reduce their attack surface and minimize the potential impact of cyber-attacks.

2.1 Challenges of IIoT-Based Systems for VM

The challenges of securing the IIoT-based systems are as follows:

1. One of the critical things that enterprises need to consider ahead of VM in IIoT is constant, uninterrupted availability of the systems except for scheduled maintenance downtime [38]. The security solutions should either work concurrently without interfering with the system's functionality, or any change to the system should only be deployed at the scheduled downtime. This raises constraints on efficiently managing the VM cycle. Among the five stages of VM, patch deployment is the crucial phase, which hinders the system's functionality. Therefore, it becomes challenging for system administrators to effectively manage the scheduled downtime to fix the vulnerabilities issues.
2. The second challenge arises due to the blend of legacy and IIoT infrastructures [25], leading to increased attack surface and increased number of attack paths to exploit the legacy vulnerabilities. This leads to legacy vulnerabilities being targeted by the attackers [48].
3. The third challenge for system administrators is to monitor and control the end-to-end security of such large and complex critical industries [37].
4. The proposed solutions for efficient VM should consider the downtime constraints to take care of the various challenges mentioned above.

This short discussion presented above helps to identify the gap in the state-of-the-art leading the research contributions mentioned in the next section.

3 Tools and Techniques for Systematic VM

In this section, we discuss Tools and techniques for each stage of systematic VM in detail.

3.1 Vulnerability Discovery

Discovering security vulnerabilities in software is a demanding task that requires significant human efforts. Vulnerability discovery is often the liability of software testers before release and white-hat hackers using bug bounty programs after the software is released. However, testers typically aim to find bugs related to performance and functionality with little focus on the security bugs due to the lack of

expertise needed to discover security bugs. In [30] observed that only 40% of the tester have formal training in software engineering practices. Apart from that, black-hat hackers also identify vulnerabilities and later exploit them to gain economic or political benefits. The bug-bounty programs offer bounties in terms of money or recognition to vulnerability discoverers [18]. Therefore, vulnerability discovery is a competition between software testers and white-hat hackers vs. black-hat testers. Discovering vulnerabilities before the software release not only save time, money, a company reputation but also provides users protection and concerns regarding the patch deployment, especially in CPS, where the availability of the systems is the primary concern. Software development with the consideration of security reduces the reported vulnerabilities. Over time, vulnerability discovery tools have evolved to discover vulnerabilities automatically. However, human intelligence acts as a supplement to these tools.

A vulnerability discovery process can be divided into five stages: information gathering, program understanding, attack surface exploration and vulnerability recognition, and reporting [52]. In the information gathering state, the major goal is to understand prior efforts and the base technologies for the program. It plays a critical role in deciding whether to expend additional effort or resources or move on to a different target. In the program understanding state, the hackers attempt to learn the program behavior and its interaction with users and the network. After discovering the program's functionality, the hacker tries to identify the attack surface. This step leads to identifying resources that can be manipulated to influence the program execution and identification of critical components of the program. In the vulnerability recognition step, system administrators explore malicious activities and pass malicious input using automated tools to identify the malicious states of software. An iterative process of program understanding, attack surface exploration and vulnerability recognition leads to identifying vulnerabilities in the system. A comprehensive report is generated in the last stage, including the vulnerability reproduction steps, which the developer later uses to generate the patches. The skilled testers perform penetration testing to identify the vulnerabilities in the system. Penetration testing (commonly known as pentesting) is an authorized simulated cyberattack on a computer system to check for exploitable vulnerabilities. The penetration tests can be performed against the system from inside or outside to study all possible attackers' strategies. Each penetration test specifies guidelines and recommendations to address the identified issues. It is generally categorized into three types: black-box, grey-box, and white-box [26]. In the case of black-box testing, no information is available to the attacker. However, in the case of grey-box testing, basic information about the network is available to the attacker. In white-box testing, detailed system information, network architecture is available to the tester [7]. Since attackers access the target system from an outside network, the black-box testing results are the most realistic pentesting technique. Most widely used tools for pentesting, such as Nmap Metasploit, Burp suite Sqlmap, subfinder are freely available on Kali Linux. A thorough penetration testing when implementing IIoT architecture will reduce the reported vulnerabilities after the software is released. In large-scale IIoT networks, manually

testing each system is challenging under resource-constrained scenarios, hence the researchers focus on automated security analysis solutions.

A manual penetration testing approach was proposed by Denis et al. [13] performing individual system penetration testing using the tools within the Kali Linux on smartphones and computers. The attacks performed were traffic sniffing, Man-in-the-Middle attack, hacking phone Bluetooth, remote desktop and open ports, etc. The primary focus of the work is to demonstrate penetration testing in a simplistic way. On the same line of work [51], developed PENTOS, a pentest tool specially designed for IoT devices to increase security awareness. PENTOS is a Graphical User Interface (GUI)-based tool on Kali-Linux, which first gathers the target system wirelessly followed by performing attacks such as web attacks and password attacks to get unauthorized access, followed by a report generation for successful attacks. PENTOS also has security guidelines for Open Web Application Security Project's top 10 vulnerabilities [41] to increase awareness [13, 51] provide practical experience of penetration testing. However, they do not demonstrate how to apply them on heterogeneous IoT nodes. Moreover, both the works are limited to a fixed set of attacks and are not scalable to a large IIoT network. With the increase in the complexity and size of the IIoT network, pentesting each and every system is a very challenging task. Therefore, researchers have focussed on using penetration graphs first to analyze the feasibility of exploitation. It facilitates the testers' analysis of the target network and provides a reference for executing penetration testing. In this direction, [56] proposed an automatic penetration graph generation algorithm combining the penetration graph generation method with the CVSS information. The authors made heuristics for generating the penetration graph that if a vulnerability has a CVSS score in the range [7–10], it will lead to admin privilege. However, they did not evaluate their framework in terms of scalability and IIoT applicability. AlGhazo et al. [1] proposed a framework that enlists a set of all possible sequences in which atomic-level vulnerabilities can be exploited to compromise specific system-level security given the networked system description. The traditional penetration testing systems are targeted to the pentesting of a system individually, which fails to detect MhMs attacks. This highlights an urgent need for new algorithms, tools, and frameworks to secure such resource-constrained devices. Koroniotis et al. [27] proposed a DL-based penetration testing framework using LSTM enabled vulnerability identification to detect the scanning attacks. The authors used Nessus, Zeek and Scapy to collect the training data by performing fuzzing scanning attacks against the network-enabled components of the smart airport-based testbed. This led to the generation of network traffic that was gathered, processed and labeled.

Future directions: In Table 1, we compare state-of-the-art vulnerability discovery approaches. We observed that most vulnerability discovery approaches focus on isolated system testing with a little focus on user-friendly GUI. These approaches will not detect the possible attacks exploiting MhMs vulnerabilities. Moreover, the penetration report only mentions the vulnerabilities reported, without further analysis, which are the critical vulnerabilities, which systems are critical in the network and

Table 1 Vulnerability discovery: summary of the related work

Research work	Tools used	Vulnerability databases used	Attacks performed	MhMs attack-paths	Critical path, node, vulnerability selection	GUI-based
Denis et al. [13]	Tools within the Kali Linux suite particularly Metasploit, Wireshark, Ettercap	✗	Traffic sniffing, man-in-the-middle attack, hacking phone bluetooth and remote desktop and open ports	✗	✗	✗
Visoottiviseth et al. [51]	Tools within the Kali Linux suite	✗	Password attack, web attack and wireless attack	✗	✗	✓
Xueqiu et al. [56]	Attack graph	Severity score provided by CVSS	✗	✓	✗	✗
Al Ghazo et al. [1]	Attack graph	✗	Remote code execution, unquoted servicepaths, user credentials construction, cross-site scripting, authentication token/cookie	✗	✗	✗
Koroniotis et al. [27]	Deep learning & Nessus, Zeek and Scapy to perform fuzzing scanning attacks to gather data for training	✗	Scanning attacks	✗	✗	✗

the most likely exploited attack paths. This analysis helps the system administrators to take proactive measures to secure the network.

3.2 Vulnerability Analysis

After identifying the vulnerabilities in the network using network scanners, penetration testing, etc., the next step of VM is to assess the vulnerabilities. A systematic and strategic assessment of a vulnerability would provide an actual severity and impact leading to an efficient resource allocation strategy. The NVD uses CVSS to analyze and assign a severity score to a vulnerability in the range [0, 10]. The vulnerabilities are analyzed based on their basic characteristics (such as Attack complexity, Attack vector, Privilege needed), temporal characteristics (such as Exploit Code Maturity, Remediation Level, Report Confidence) and environmental characteristics. Weighted Impact Vulnerability Scoring System (WIVSS) [49] is proposed to achieve higher diversity and accuracy of severity scores. WIVSS uses factors similar to CVSS, i.e., attack vector, attack complexity, authentication, confidentiality impact, integrity impact and availability impact. However, it uses different weights for the impact metrics (confidentiality impact, integrity impact and availability Impact) compared to the CVSS.

Phillips et al. [43] proposed a graph-based vulnerability analysis system, where a node represents a stage of attack and edge represents the transitions between the attack stages for network-vulnerability analysis considering internal and external attackers. The analysis system needs a common attack database with respective network configuration and topology configuration is analyzed. The level of effort is calculated by combining the probability of success on the edges. The likelihood of success is proportional to attack-path length. The major limitation of the work lies in the need for atomic steps of attacks. In a practical case, an attacker does not always follow a fixed set of patten. Moreover, the authors only presented a brief idea about the analysis system with no implementation and scalability analysis.

Ammann et al. [5] proposed a scalable vulnerability analysis approach by considering an assumption of monotonicity, i.e., the precondition of an exploit remains the same irrespective the attacker has exploited another vulnerability. The goal is achieved by combining the attacker access privilege, network connectivity and vulnerability in a common attribute, reducing the attack graphs' complexity.

Future directions: CVSS and WIVSS do not consider the domain characteristics while scoring the vulnerabilities. Therefore, directly using CVSS severity score and analysis may not give the exact severity of a vulnerability. Hence, extending the CVSS vulnerability analysis is necessary by considering the environment and network characteristics for deploying further security measures.

3.3 Vulnerability Prioritization

With the expansion of networks due to IIoTization, more and more IIoT devices are connected to the Internet. Hence, there is a drastic increase in the number of vulnerabilities reported on these systems. Currently, NVD contains more than 1.60 lakhs vulnerabilities, out-of-which 18,767 vulnerabilities were reported in 2020 itself. Patching each vulnerability is a very challenging task. However [21], studied the ratio of vulnerability exploited and vulnerability reported for 2009–2018. 76 k vulnerabilities were reported to NVD in the mentioned period, out of which about 12.8% (9.7/76 k) of all vulnerabilities had their published exploit code. A key observation is that only about 5% (4.2/76 k) vulnerabilities were exploited. This shows that not all vulnerabilities are exploited, nor all vulnerabilities can be patched in a resource-constrained scenario. Hence, vulnerability prioritization should be considered.

To efficiently handle these scenarios in a resource-constrained environment, industries prioritize vulnerability patching using crude heuristics based on limited data. Hence, many known vulnerabilities are breached by attackers for which the patch was already available. It raises a few challenges to the system administrators:

1. Suppose we patch all the vulnerabilities of the network. In that case, resources are consumed on the low-severity vulnerability, which has less probability of exploitability and low impact, even if they got exploited.
2. In another scenario, if we patch a few critical-severity vulnerabilities, it may be an economical, efficient strategy but may lead to other high-risk vulnerabilities, including MhMs exploitation.

In brief, vulnerability prioritization is a practice to balance resource availability and exploitation impacts with a large amount of discovered vulnerabilities. The vulnerability prioritization should be strategic and efficient.

Game theory has been used widely in capturing the strategic interactions between the intelligent agents, i.e., the attacker and the defender, where the payoff of each depends not only on their own action but also on other players' actions. Apart from game theory, graph theory is also used to find an optimized strategy. The expert analysis also helps to understand the severity of a vulnerability. Next, we discuss related work in each category, i.e., expert analysis based, graph theory-based and game theory-based approaches in detail.

Expert analysis based vulnerability prioritization approaches: The CVSS is an indicator of true vulnerability severity. CVSS is used by nexpose [44] vulnerability management tool to rank the vulnerabilities. However, the severity score provided by CVSS is static and has not changed over time. These scores are standard for all systems and can be improved by considering temporal and environmental metrics with base metrics [17]. WIVSS [49] is proposed to achieve higher diversity and accuracy of severity scores. WIVSS uses factors similar to CVSS, i.e., attack vector, attack complexity, authentication, confidentiality impact, integrity impact and availability impact. However, it uses different weights for the impact metrics (confidentiality impact, integrity impact and availability impact) compared to the CVSS.

Graph-based vulnerability prioritization approaches: Graph-based vulnerability prioritizing approaches like SecureRank [35], Risk-Rank [3] and VULCON [16] provide a static ranking of patching order and they do not consider the behavior of an attacker. SecureRank defines a security metric based on the percentage of time a random attacker would spend endeavoring to exploit a vulnerability successfully. It takes network topology and vulnerability severity as inputs and returns defense probability for each subsystem. Defense probability denotes the probability of selecting a vulnerability on a particular subsystem for patching to reach the optimal state. Our framework in stage 3 establishes that it reaches a Nash equilibrium. The authors compared SecureRank with density, source and type-based prioritization and observed that SecureRank provides an effective and efficient patch prioritization approach. It prioritizes vulnerabilities based on a balance between immediate risk and the risk due to system interdependencies' cascading. The Risk-Rank algorithm captures the risk diffusion by using complex interaction over time. Risk-Rank is verified by using a case study based on the organization's conceptual structure, business units' risk dependencies and vulnerabilities. VULCON is a patch prioritization framework proposed for network security management. It is based on fundamental performance metrics, i.e., "time-to-vulnerability remediation" and "total vulnerability exposure". The proposed algorithm uses a mixed-integer multi-objective optimization algorithm to prioritize vulnerabilities for patching subject to the given resource constraints. However, the graph theory-based approaches fail to incorporate the attacker behavior, which plays a vital role in analyzing the possible impact of exploiting a vulnerability.

Game theory-based vulnerability prioritization approaches: Game theory-based approaches for patch prioritization [4, 10, 24, 47] incorporate attackers' behavior to better estimate the prioritization strategy.

Alshawish and Risk de Meer [47] proposed a game-theoretical model to optimize the security strategy of electricity distribution networks with vulnerable Distributed Energy Resource (DER) nodes. The authors consider an adversarial model for false data injection attacks to compromise vulnerable nodes. The impact of this attack in a smart grid on a defender includes the loss of voltage regulation and the cost of induced load control under supply-demand mismatch between the generator and distributor. The proposed greedy approach is formulated in a three-stage defender-attacker-defender game, (i) the defender first chooses a strategy to secure DER nodes (ii) the attacker will try to compromise the DER nodes (iii) the defender chooses the security investments strategy by controlling the loads and non-compromised nodes. The authors use a greedy approach to compute attacker-defender strategies and recommend optimal financial investments to secure the systems. Kamdem et al. [24] proposed a two-player zero-sum Markov game to identify the optimal strategy to disconnect vulnerable services to slow down the attack.

Alshawish and Risk de Meer [4] proposed an integrated risk-based methodology for prioritizing possible vulnerability remediation activities by leveraging Time-To-Compromise (TTC) security metric. This model employs the network topology, attackers' capability and published vulnerability and exploit information. TTC is calculated by taking into account the total number of disclosed vulnerabilities, the

number of high severity vulnerabilities, the number of low severity vulnerabilities, the total number of existing exploits, the expected time taken for identifying the zero-day vulnerability, the expected time taken for calculating the exploit and adversarial skill set. The authors provide a game-theoretic approach considering the stochastic nature of risk assessments across an electric power organization. The authors acknowledged that TTC-based models could convey misleading results due to the aggregation of anticipated features of a vulnerability. Chen et al. [10] proposed a bi-level optimization model under a game-theoretic framework to incorporate the interactions of a system administrator and an adversary. The interactions among cyber-physical elements are considered to determine cascading failure under potential attacks. The approach leads to optimal resource allocation by the system defender to maintain system reliability. However, the game theory-based approaches proposed earlier for patch prioritization consider only the single attacker-defender scenario, which is not pragmatic in all cases.

Apart from the above approaches [2], proposed an ML-based exploit prediction model leveraging vulnerability information from different databases, i.e., NVD, ExploitDB, ZDI and Dark Web (DW). The attacker behavior is integrated by considering the blogs/ posts for respective vulnerabilities on DW. However, the learning-based detection approaches may be deceived due to intentionally discussing the random vulnerabilities on DW by adversaries.

Future directions: In Table 2, we compare state-of-the-art vulnerability prioritization approaches based on architectural feature, vulnerability feature, patch dependencies, attacker feature and approach category. We observed that most approaches do not consider resource constraints, functional dependencies, patch dependencies and multiple defender-attackers practical scenarios. Incorporating an attacker's behavior plays a vital role in proper resource allocation and failure to consider the patch dependencies will lead to patch breaks while deployed. In this direction [57, 58], proposed a prioritization framework leveraging a game-theoretical model. However, the approach can be extended by considering different attacker strategies and network characteristics.

3.4 Vulnerability Remediation

After the system administrators have analyzed and prioritized the vulnerabilities, the next phase is to deploy the patches. It is the most challenging stage of patch management due to the complexities of software arising from network inter-connectivity and inter-dependencies. Hence, a patch deployment may affect other dependent applications potentially. Apart from software dependencies, the patch dependencies hierarchy needs to be considered, i.e., if patch A depends upon patch B then before deploying patch A, the administrators need to deploy patch B. Another concern is the limited time between the patch availability and the exploit release, leading to the high probability of successful exploitation. This raises concern for the deployment of

Table 2 Vulnerability prioritization: summary of the related work

Research work	Architectural feature	Vulnerability feature	Attacker feature	Functional dependency	Multiple-attacker-defender	Patch dependencies
Rapid7-community [44]	✗	Attack vector, attack complexity, scope, impact (confidentiality, integrity, availability)	✗	✗	✗	✗
Spanos et al. [49]	✗	Weighted characteristics (Attack vector, attack complexity, scope, impact (confidentiality, integrity, availability)	✗	✗	✗	✗
Miura-Ko and Bambos [35]	Network topology	% of time a random attacker would spend trying to exploit	Random attacker	✗	✗	✗
Alpcan and Bambos [3]	Bipartite graph	✗	✗	✗	✗	✗
Farris et al. [16]	Mission criticality score	Total vulnerability exposure, Time-to-vulnerability remediation	✗	✗	✗	✗
Shelar and Amin [47]	Network model of radial electric distribution systems	Impact of attack in terms of loss of voltage regulation and cost of induced load control under supply-demand mismatch	Three-stage defender-attacker-defender game	✗	✗	✗
Alshawish and Meer [4]	Network topology	Time-to-compromise feature	All possible attack-paths to the target node	✗	✗	✗
Yadav and Paul [59]	Inter dependencies	Cost of defend, cost of attack and Impact of attack (NVD Database, CVSS score)	Single attacker	✗	✗	✗
Kamdem et al. [24]	Nework topology	CVSS severity score	Single attacker	✗	✗	✗
Chen et al. [10]	Node-link model	Cascading failure under malicious attacks	Single attacker	✗	✗	✗

the patches as soon as they are available. A security patch should be well tested before deployment. It may sometimes break the service rather than repairing faulty patches that introduce issues like backward compatibility, interoperability issue, patch break and introduction of a new vulnerability. The presence of faulty patches increases the cost of patch deployment and service downtime. Hence, many system administrators often delay installing patches and keep using outdated software, leaving known vulnerabilities readily exploitable. However, in ICS systems, the patching is scheduled with consideration of the requirement of the system availability, pre-deployment testing and post-deployment testing. In brief, the difficulty in dealing with patch dependencies and the significant amount of human effort required for configuring a test environment to simulate a production-identical environment hinders automated patch deployment. Therefore, before deploying a patch, a deep analysis of the impact of patch deployment should be done. A sophisticated live patching technique has been proposed to reduce the service downtime or maintenance window [33]. However, their applicability in practice is minimal. Commonly available virtualization capabilities allow system administrators to perform a majority of the patchwork outside of the maintenance window by capturing the disk activities and replaying them during the actual maintenance window.

Future directions: A patch management policy that tests and applies suitable patches to all affected areas in an efficient and timely manner is crucial. A trustworthy remediation solution helps developers, security and devOps teams by keeping them in sync so that the entire vulnerability management process runs smoothly. With the decrease in system downtime and the need to keep systems updated highlights the need for advanced techniques for live patching.

3.5 Vulnerability Verification and Monitoring

Once the vulnerabilities have been patched, the next stage of VM is to verify or test the deployed patches on these systems. The patch deployments are verified by monitoring the systems for unexpected service interruptions. Manual patch deployment verification approaches are challenging, error-prone and time-consuming in complex networks. There is a lack of automated tools to overview the state of the system post-patch deployment.

Moreover, due to system availability requirements, few critical vulnerabilities can not be patched in ICS systems. In those cases, system administrators need to monitor these systems to timely detect ongoing exploitations. If an ongoing attack is detected in the network, which may lead a path to the critical asset, suitable actions to stop the attack or reduce the attack's impact should be taken. There has been active research for more than a decade for using system logs to detect the anomalous behavior of a system using either rule-based strategies or ML-based techniques on a single system. However, only a few approaches focus on correlating the attack scenario on different systems to find the indications of compromise, which leads to the detection of an

attack before it reaches its target. We study the related work into three categories as discussed below:

Techniques for anomaly detection on a system using system logs: Rule-based anomaly detection approaches [40, 64] are limited to detect specific scenarios with high accuracy, requiring domain expertise. [64] represented the Syslog behaviors using a combination of hidden Markov models followed by learning the model using a discounting learning algorithm. Oprea et al. [40] proposed a graph-theoretic framework based on belief propagation to detect advanced persistence threats infection. The ML-based anomaly detection approaches can be categorized as supervised and unsupervised learning-based approaches. Supervised learning-based approaches derive a model from the labeled training data, which generally label data either normal or anomalous. Chen et al. [11] presented a decision tree-based approach to diagnose failures on Internet sites. First, they trained the decision trees on the request traces. The training request traces data also included the request failure scenarios visible and labeled by the user. When tested on real-failure data from eBay (an eCommerce website) request traces, the proposed approach successfully identified 13 out of 14 failure cases. Liang et al. [31] applied Support Vector Machine (SVM) to predict failures in IBM BlueGene/L event logs[2]. The supervised-learning-based approaches need a large amount of labeled data to train the model. In an unsupervised-learning algorithm clustering approach, LogCluster utilizes the base idea to check if a particular log sequence has occurred or not [31]. Apart from these, program invariants were used to detect abnormal events. Initially, program invariants are being identified to learn the linear relationships between system events during the program execution. A log sequence that does not follow the program invariants is labeled as anomalous. The above ML approaches made a close-world assumption that the log set is finite and data will be stable over the period. However, in practice, log data may encounter previously unseen log sequences, decreasing the accuracy of the anomalous log detection. In this direction, Deeplog [15] is an online anomalous log detection approach using the LSTM model. The approach consists of three key modules, i.e., key anomaly detection, parameter anomaly detection and workflow construction. Deeplog is trained on normal data only and can adapt new log patterns on false positive detection. Zhang et al. [65] proposed an anomaly detection approach by utilizing an attention-based Bi-LSTM model. Meng et al. [34] highlighted to use of the semantics of the log messages rather than the indexes, which is generally used for anomaly detection to reduce the false positive. LSTMs have proved to be a promising solution to sequence and time-series related problems.

[2] The event logs are the events from OSs, applications or devices and are stored in a single cluster by the operating system. Events logged by the operating system are also called system logs.

Techniques for ongoing attack detection on single system logs: In this direction [8], proposed a rule-based model to detect targeted port scans, detection of Cross-Site Scripting (XSS) and SQL Injection (SQLI) attacks using access logs of Apache HTTP Server. Moh et al. [36] leveraged the features of both rule-based and learning-based approaches to detect the SQLI attacks using web server logs. A collaborative approach by combining intrusion detection at different layers, i.e., network, kernel and application, can increase the accuracy of attack detection as compared to individual detectors, without much degradation in performance [55]. An attack-story reconstruction approach proposed by Pei et al. [42] correlates the log graph utilizing logs from different levels on a single host. However, these approaches are limited to attack detection on a single system with knowledge of how they can be used for correlated attacks.

Approaches for temporal and spatial correlation of attacks using logs: In this direction [11], proposed a process query system based on control and estimation methods to correlate the distributed network events. Attack graph has been used for correlating attacks on MhMs attacks. However, manual construction of the attack graphs is challenging and error-prone. Few automatic attack-graph generation have been proposed in literature e.g. [1, 20, 22, 62, 63]. The attack-graph generation approaches either use their model checker or use a knowledge database of vulnerabilities and exploits, e.g., NVD, ExploitDb etc., to generate the pre-requisites and post-conditions related to exploitation steps. The pre-requisite and post-conditions of a vulnerability denote the conditions needed to exploit a vulnerability and the capability gained by exploiting it. However, these approaches [1, 20, 22, 62, 63] limit themselves to generate the attack paths only, i.e., they will not detect any ongoing attacks. Therefore, there is a need for an effective methodology to detect ongoing MhMs attacks timely.

Future directions: In Table 3, we compare state-of-the-art vulnerability monitoring approaches based on 'Data used', 'Technique used', 'Attacks detected', 'Detect attacks on single system' and 'Detect MhMs attacks'. We observe that except [11], the approaches are targeted to detect vulnerabilities exploitation on a single system. [11] approach lack the practical implementation and feasibility analysis.

Table 3 Vulnerability monitoring: summary of the related work

Research work	Data used	Techniques used	Attacks detected	Detect attacks on single system	Detect MhMs attacks
Yen et al. [64]	Log data	✗	Variation from the normal behaiour	✗	✗
Oprea et al. [40]	Web proxy logs	Belief propagation inspired from graph theory	APT infection attack	✗	✗
Chen et al. [11]	Request traces	Decision trees	Causes of failures	✗	✗
Liang et al. [31]	System logs	SVM and nearest neighbor method	Variation from the normal behaiour	✗	✗
Zhang et al. [65]	System logs	An attention-based Bi-LSTM model	Variation from the normal behaiour	✗	✗
Meng et al. [34]	System logs	Extracting semantic information using Template2Vec	Sequential and quantitative anomaly detection	✗	✗
Du et al. [15]	System Logs	LSTM-based deep learning	Denial of service attack, port scan, socially engineered attack	✓	✗
Wu et al. [55]	Logs from IDS Snort, Libsafe and sysmon	Graph-based and a Bayesian network based aggregation method	Buffer overflow, flooding and script-based attacks	✓	✗
Pei et al. [42]	DNS logs, Auditd logs, Firefox logs, Syslog	Graph analytics	A phishing email, watering-hole attack, trojan software, an unofficial patch containing malicious payloads	✓	✗
Jiang and Cybenko [11]	System events	Control and estimation methods		✓	✓
Ba et al. [8]	Access log files of Apache web servers	Rule-set based approach	Web scan detections, SQL injection and XSS attacks	✓	✗
Moh et al. [36]	Web server logs	Hybrid approach (combine rule and larning based approach)	SQL Injection	✓	✗

4 Reseach Directions

The researchers should aim at building techniques for an efficient VM.

1. The researcher should analyze the reported vulnerabilities specific to IIoT-based SCADA systems to understand better the type of attacks, the vulnerable components, and the vulnerabilities' impact.
2. A focus on developing frameworks to analyze and find a series of vulnerabilities in different systems that are needed to exploit to reach the target system. The framework should recommend a consolidated report of the vulnerable state of the system and all possible target paths to the critical node. The framework should be scalable to the IIoT network.
3. Not all vulnerabilities are always exploited by the attackers, and not all vulnerabilities can be patched due to the resource constraints such as people, infrastructure, tools and time available to patch every vulnerability. Also, ICSs such as SCADA have strict system uptime and availability requirements. These constraints place significant importance on the patch prioritization of networks and devices, which needs to be strategic and efficient.
   There is a need to develop a patch prioritization framework that is applicable to ICSs. The prioritization order should consider the architectural characteristics to understand the domain knowledge of the target system, vulnerability characteristics to embed the vulnerability severity, patch dependencies to avoid the patch break on deployment and attacker behavior to reflect a practical scenario. The framework should recommend a strategy for patching, which is optimal and effective considering resource constraints.
4. Moreover, the researcher should focus on designing and developing frameworks that correlate the evidence of an incident spread temporarily and spatially in the network. The framework should detect the ongoing exploitation of MhMs vulnerabilities on a system. In this direction, GloM has been presented to monitor MhMs attack [61].

5 Conclusion

In this chapter, we first discussed the correlation between SCADA systems and IIoT-based systems, followed by the need of VM for securing these systems. We discuss what is VM? why we need VM? how to perform VM? Afterward, we discuss the issues with state-of-the-art vulnerability management approaches. We observed that vulnerability discovery approaches focus on isolated system testing with a little focus on user-friendly GUI. These approaches will not detect the possible attacks exploiting MhMs vulnerabilities. Moreover, the penetration report only mentions the vulnerabilities reported, without further analysis, which are the critical vulnerabilities, which systems are critical in the network and the most likely exploited attack paths. This analysis helps the system administrators to take proactive measures to

secure the network. We observed that most vulnerability prioritization approaches do not consider resource constraints, functional dependencies, patch dependencies and multiple defender-attackers practical scenarios. Incorporating an attacker's behavior plays a vital role in proper resource allocation and failure to consider the patch dependencies will lead to patch breaks while deployed. We also observed that the vulnerability monitoring approaches are targeted to detect the exploitation of vulnerabilities on a single system only. Hence fail to detect the ongoing MhMs attacks timely.

References

1. A.T. Al Ghazo, M. Ibrahim, H. Ren, R. Kumar, A2G2V: automated attack graph generator and visualizer. in *Mobile IoT SSP'18*, vol. 3 (ACM, Los Angeles, CA, USA, 2018), pp. 1–6. https://doi.org/10.1145/3215466.3215468
2. M. Almukaynizi, E. Nunes, K. Dharaiya, M. Senguttuvan, J. Shakarian, P. Shakarian, Patch before exploited: an approach to identify targeted software vulnerabilities, in *AI in Cybersecurity*, ed. by F.S. Leslie (Springer International Publishing, Cham, 2019), pp. 81–113. https://doi.org/10.1007/978-3-319-98842-9_4
3. T. Alpcan, N. Bambos, Modeling dependencies in security risk management, in *2009 Fourth International Conference on Risks and Security of Internet and Systems (CRiSIS 2009)* (2009), pp. 113–116
4. A. Alshawish, H. Risk de Meer, Risk mitigation in electric power systems: where to start? Energy Inform. **2**(1), 34 (2019)
5. P. Ammann, D. Wijesekera, S. Kaushik, Scalable, graph-based network vulnerability analysis, in *Proceedings of the 9th ACM Conference on Computer and Communications Security. CCS '02* (Association for Computing Machinery, Washington, DC, USA, 2002), pp. 217–224. https://doi.org/10.1145/586110.586140
6. A. Andreu, Operational technology security—A data perspective. Netw. Secur. **1**, 8–13 (2020). https://doi.org/10.1016/S1353-4858(20)30008-8
7. R. Ankele, S. Marksteiner, K. Nahrgang, H. Vallant, Requirements and recommendations for IoT/IIoT models to automate security assurance through threat modelling, security analysis and penetration testing, in *Proceedings of the 14th International Conference on Availability, Reliability and Security. ARES '19* (Association for Computing Machinery, Canterbury, CA, United Kingdom, 2019). https://doi.org/10.1145/3339252.3341482
8. S.M. Ba, F.O. Catak, E. Gül, Detection of attack-targeted scans from the apache HTTP server access logs. Appl. Comput. Inf. **14**(1), 28–36. https://doi.org/10.1016/j.aci.2017.04.002
9. H. Boyes, B. Hallaq, J. Cunningham, T. Watson, The industrial internet of things (IIoT): an analysis framework. Comput. Ind. **101**, 1–12 (2018). https://doi.org/10.1016/j.compind.2018.04.015
10. K. Chen, W. Fushuan, C.-L. Tseng, M. Chen, Z. Yang, H. Zhao, H. Shang, A game theory-based approach for vulnerability analysis of a cyber-physical power system. Energies **12**(15), 3002 (2019). https://doi.org/10.3390/en12153002
11. M. Chen, A.X. Zheng, J. Lloyd, M.I. Jordan, E. Brewer, *Failure Diagnosis Using Decision Trees* (2004), pp. 36–43
12. Y. Cherdantseva, P. Burnap, A. Blyth, P. Eden, K. Jones, H. Soulsby, K. Stoddart, A review of cyber security risk assessment methods for SCADA systems. Comput. Secur. **56**, 1–27 (2016). https://doi.org/10.1016/j.cose.2015.09.009

13. M. Denis, C. Zena, T. Hayajneh, Penetration testing: concepts, attack methods, and defense strategies, in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* (2016), pp. 1–6. https://doi.org/10.1109/LISAT.2016.7494156
14. L.L. Dhirani, E. Armstrong, T. Newe, Industrial IoT, cyber threats, and standards landscape: evaluation and roadmap. Sensors **21**(11) (2021). https://doi.org/10.3390/s21113901
15. M. Du, F. Li, G. Zheng, V. Srikumar, DeepLog: anomaly detection and diagnosis from system logs through deep learning, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. CCS '17* (Association for Computing Machinery, Dallas, Texas, USA, 2017), pp. 1285–1298. https://doi.org/10.1145/3133956.3134015
16. K.A. Farris, A. Shah, G. Cybenko, R. Ganesan, S. Jajodia, VULCON: a system for vulnerability prioritization, mitigation, and management. ACM Trans. Priv. Secur. **21**(4) (2018). https://doi.org/10.1145/3196884
17. C. Fruhwirth, T. Mannisto, Improving CVSS-based vulnerability prioritization and response with context information, in *2009 3rd International Symposium on Empirical Software Engineering and Measurement* (2009), pp. 535–544. https://doi.org/10.1109/ESEM.2009.5314230
18. R. Hamper, Software bug bounties and legal risks to security researchers. Ph.D. thesis (2019)
19. Idaho-National-Laboratory, History of industrial control system cyber incidents (2018). https://www.osti.gov/servlets/purl/1505628. Accessed 04 May 2020
20. K. Ingols, R. Lippmann, K. Piwowarski, Practical attack graph generation for network defense, in *Proceedings of the 22nd Annual Computer Security Applications Conference. ACSAC '06* (IEEE Computer Society, Washington, DC, USA, 2006), pp. 121–130. https://doi.org/10.1109/ACSAC.2006.39
21. J. Jacobs, S. Romanosky, I. Adjerid, W. Baker, Improving vulnerability remediation through better exploit prediction. J. Cybersecur. **6**(1), tyaa015 (2020). https://doi.org/10.1093/cybsec/tyaa015. https://academic.oup.com/cybersecurity/article-pdf/6/1/tyaa015/33746021/tyaa015.pdf
22. S. Jajodia, S. Noel, B. O'Berry, Topological analysis of network attack vulnerability, in *Managing Cyber Threats: Issues, Approaches, and Challenges. Ed. by Vipin Kumar, Jaideep Srivastava, and Aleksandar Lazarevic* (Springer US, Boston, MA, 2005), pp. 247–266. https://doi.org/10.1007/0-387-24230-9_9
23. G. Jiang, G. Cybenko, Temporal and spatial distributed event correlation for network security, in *Proceedings of the 2004 American Control Conference*, vol. 2 (2004), pp. 996–1001. https://doi.org/10.23919/ACC.2004.1386701
24. G. Kamdem, C. Kamhoua, Y. Lu, S. Shetty, L. Njilla, A Markov game theoritic approach for power grid security, in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (2004), pp. 139–144. https://doi.org/10.1109/ICDCSW.2017.63
25. K. Keshav, S.S. Vijay, D.M. Lourenço, A. Anil Kumar, P. Plapper, Retrofitting of legacy machines in the context of industrial internet of things (IIoT), in *3rd International Conference on Industry 4.0 and Smart Manufacturing on Procedia Computer Science*, vol. 200 (2022), pp. 62–70. https://doi.org/10.1016/j.procs.2022.01.205. https://www.sciencedirect.com/science/article/pii/S1877050922002149
26. M.E. Khan, F. Khan, A comparative study of white box, black box and grey box testing techniques. Int. J. Adv. Comput. Sci. Appl. **3**(6) (2012). https://doi.org/10.14569/IJACSA.2012.030603
27. N. Koroniotis, N. Moustafa, B. Turnbull, F. Schiliro, P. Gauravaram, H. Janicke, A Deep learning-based penetration testing framework for vulnerability identification in internet of things environments (2021). arXiv: 2109.09259 [cs.CR]
28. R.M. Lee, M.J. Assante, T. Conway, German steel mill cyber attack. Ind. Control Syst. 1–15 (2014)
29. M. Lehto, Cyber security in aviation, maritime and automotive. Comput. Big Data Transp. 19–32 (2010)
30. T.C. Lethbridge, J. Diaz-Herrera, R.J. Jr., LeBlanc, J.B. Thompson, Improving software practice through education: challenges and future trends, in *2007 Future of Software Engineering. FOSE '07* (IEEE Computer Society, USA, 2007), pp 12–28. https://doi.org/10.1109/FOSE.2007.13

31. Y. Liang, Y. Zhang, H. Xiong, R. Sahoo, Failure prediction in IBM blueGene/L event logs (2007); In Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, X. Chen, Log clustering based problem identification for online service systems, in *Proceedings of the 38th International Conference on Software Engineering Companion. ICSE '16* (Association for Computing Machinery, Austin, Texas, 2016), pp. 102–111. https://doi.org/10.1145/2889160.2889232
32. Y. Lu, P. Witherell, A. Jones, Standard connections for IIoT empowered smart manufacturing. Manuf. Lett. **26**, 17–20 (2020). https://doi.org/10.1016/j.mfglet.2020.08.006
33. M. Maurer, David Brumley, Tachyon: tandem execution for efficient live patch testing, in *21st USENIX Security Symposium (USENIX Security 12)*. (Bellevue, WA, USENIX Association, 2012), pp. 617–630
34. W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, R. Zhou, LogAnomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs, in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization* (2019), pp. 4739–4745. https://doi.org/10.24963/ijcai.2019/658
35. R.A. Miura-Ko, N. Bambos, SecureRank: a risk-based vulnerability management scheme for computing infrastructures, in *2007 IEEE International Conference on Communications* (2007), pp. 1455–1460. https://doi.org/10.1109/ICC.2007.244
36. M. Moh, S. Pininti, S. Doddapaneni, T.-S. Moh, Detecting web attacks using multi-stage log analysis, in *2016 IEEE 6th International Conference on Advanced Computing (IACC)* (2016), pp. 733–738. https://doi.org/10.1109/IACC.2016.141
37. A. Mosteiro-Sanchez, M. Barcelo, J. Astorga, A. Urbieta, End to end secure data exchange in value chains with dynamic policy updates, in *CoRR* (2022). arXiv: 2201.06335
38. C. Niesler, S. Surminski, L. Davi, Hera: hotpatching of embedded real-time applications, in *28th Network and Distributed System Security Symposium (NDSS)* (2021); NIST, National vulnerability database (2021). https://nvd.nist.gov/
39. Nozomi-Networks, Nozomi-networks (2021)
40. A. Oprea, Z. Li, T.-F. Yen, S.H. Chin, S. Alrwais, Detection of early-stage enterprise infection by mining large-scale log data, in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (2015), pp. 45–56. https://doi.org/10.1109/DSN.2015.14
41. OWASP-community, OWASP top ten (2021). https://owasp.org/www-projecttop-ten/
42. K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, D. Xu, HERCULE: attack story reconstruction via community discovery on correlated log graph, in *Proceedings of the 32nd Annual Conference on Computer Security Applications. ACSAC '16* (Association for Computing Machinery, Los Angeles, California, USA, 2016), pp. 583–595. https://doi.org/10.1145/2991079.2991122
43. C. Phillips, L.P. Swiler, A graph-based system for network-vulnerability analysis, in *Proceedings of the 1998 Workshop on New Security Paradigms. NSPW '98* (Association for Computing Machinery, Charlottesville, Virginia, USA, 1998), pp. 71–79. https://doi.org/10.1145/310889.310919
44. Rapid7-community, Working with vulnerabilities (2021). https://docs.rapid7.com/nexpose/working-with-vulnerabilities/. Accessed 13 June 2021
45. J.R. Reeder, C.T. Hall, Cybersecurity's pearl harbor moment: lessons learned from the colonial pipeline ransomware attack (2021)
46. SANS-ICS, Analysis of the cyber attack on the Ukrainian power grid (2016). https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf. Accessed 03 Jan. 2021
47. D. Shelar, S. Amin, Security assessment of electricity distribution networks under DER node compromises. IEEE Trans. Control of Netw. Syst. **4**(1):23–36 (2017)
48. K. Smith, I. Wilson, The challenges of the internet of things considering industrial control systems, in *Privacy, Security And Forensics in The Internet of Things (IoT)*, ed. by R. Montasari, F. Carroll, I. Mitchell, S. Hara, R. Bolton-King (Springer International Publishing, Cham, 2022), pp. 77–94. https://doi.org/10.1007/978-3-030-91218-5_4
49. G. Spanos, A. Sioziou, L. Angelis, WIVSS: a new methodology for scoring information systems vulnerabilities, in *Proceedings of the 17th Panhellenic Conference on Informatics. PCI '13*

(Association for Computing Machinery, Thessaloniki, Greece, 2013), pp. 83–90. https://doi.org/10.1145/2491845.2491871
50. Tenable-community, Nessus (2021). https://www.tenable.com/products/nessus. Accessed 13 Oct. 2021
51. V. Visoottiviseth, P. Akarasiriwong, S. Chaiyasart, S. Chotivatunyu, PENTOS: penetration testing tool for internet of thing devices, in *TENCON 2017—2017 IEEE Region 10 Conference* (2017), pp. 2279–2284. https://doi.org/10.1109/TENCON.2017.8228241
52. D. Votipka, R. Stevens, E. Redmiles, J. Hu, M. Mazurek, Hackers versus testers: a comparison of software vulnerability discovery processes, in *2018 IEEE Symposium on Security and Privacy (SP)* (2018), pp. 374–391. https://doi.org/10.1109/SP.2018.00003
53. B. Wang, X. Li, L.P. de Aguiar, D.S. Menasche, Z. Shafiq, Characterizing and modeling patching practices of industrial control systems. Proc. ACM Meas. Anal. Comput. Syst. **1**(1). https://doi.org/10.1145/3084455
54. S.A. Weed, US policy response to cyber attack on SCADA systems supporting critical national infrastructure (2017). https://media.defense.gov/2017/Nov/20/2001846609/-1/-1/0/CPP0007_WEED_SCADA.PDF. Accessed 02 Mar. 2022
55. Y.S. Wu, B. Foo, Y. Mei, S. Bagchi, Collaborative intrusion detection system (CIDS): a framework for accurate and efficient IDS, in *Proceedings of the 19th Annual Computer Security Applications Conference. ACSAC '03* (IEEE Computer Society, USA, 2003), p. 234
56. Q. Xueqiu, S.W. Jia, C. Xia, L. Lv, Automatic generation algorithm of penetration graph in penetration testing, in *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (2014), pp. 531–537. https://doi.org/10.1109/3PGCIC.2014.104
57. G. Yadav, P. Gauravaram, A.K. Jindal, SmartPatch: a patch prioritization framework for SCADA chain in smart grid, in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking. MobiCom '20* (Association for Computing Machinery, London, United Kingdom, 2020). https://doi.org/10.1145/3372224.3418162
58. G. Yadav, P. Gauravaram, A.K. Jindal, K. Paul, SmartPatch: a patch prioritization framework. Comput. Ind. **137**, 103595 (2022). https://doi.org/10.1016/j.compind.2021.103595. https://www.sciencedirect.com/science/article/pii/S0166361521002025
59. G. Yadav, K. Paul, PatchRank: ordering updates for SCADA systems, in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (IEEE ETFA)* (2022). https://doi.org/10.1109/ETFA.2019.8869110
60. G. Yadav, K. Paul, Architecture and security of SCADA systems: a review. Int. J. Critic. Infrastr. Protect. **34**, 100433 (2021). https://doi.org/10.1016/j.ijcip.2021.100433. https://www.sciencedirect.com/science/article/pii/S1874548221000251
61. G. Yadav, K. Paul, Global monitor using spatiotemporally correlated local monitors, in *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)* (2021), pp. 1–10. https://doi.org/10.1109/NCA53618.2021.9685330
62. G. Yadav, K. Paul, A. Allakany, K. Okamura, IoT-PEN: a penetration testing framework for IoT, in *2020 International Conference on Information Networking (ICOIN)* (2020a), pp. 196–201. https://doi.org/10.1109/ICOIN48656.2020.9016445
63. G. Yadav, K. Paul, A. Allakany, K. Okamura, IoT-PEN: an E2E penetration testing framework for IoT. J. Inf. Process. **28**, 633–642 (2020b). https://doi.org/10.2197/ipsjjip.28.633.
64. T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, E. Kirda, Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks, in *Proceedings of the 29th Annual Computer Security Applications Conference. ACSAC '13* (Association for Computing Machinery, New Orleans, Louisiana, USA, 2013), pp. 199–208
65. X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, D. Zhang, Robust log-based anomaly detection on unstable log data, in *ESEC/FSE 2019. Tallinn, Estonia: Association for Computing Machinery* (2019), pp. 807–817. https://doi.org/10.1145/3338906.3338931