NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium | 978-1-6654-7716-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/NOMS56928.2023.10154338

# Worst Attack Vulnerability and Fortification for IoT Security Management: An approach and An Illustration for Smart Home IoT

Fathima James[1], Indrajit Ray[2], Deep Medhi[3]

[1]University of Missouri–Kansas City, USA, [2]Colorado State University, USA, [3]National Science Foundation, USA
fjmb7@umsystem.edu, indrajit.ray@colostate.edu, dmedhi@nsf.gov

***Abstract*—In the domain of IoT security management, we consider attack vulnerabilities and how to identify those so as to prevent attacks from spreading. More specifically, inspired by this problem for Smart Home Internet of Things (SHIoT), we take a complex network framework in which an IoT system attack graph can be cast. We then address the problem of assessing the worst vulnerability, that is the one that has the potential to cause maximum damage, in the SHIoT. Due to the non-additive nature of an attack path's attack probability, we show how the problem can be modeled so that a shortest path-based algorithm approach can be used to determine the worst vulnerability. We then illustrate an approach to iteratively fortify the environment to reduce impact from vulnerability. Finally, we show an approach to use Common Vulnerability Scoring System (CVSS) to determine attack probabilities on arcs in the attack graph and present analysis on representative attack graphs for small to large attack graphs.**

***Index Terms*—IoT device management, Smart Home Internet of Things, Finite State Automata based Attack Model, Graph Analysis, Vulnerability Analysis.**

## I. Introduction

IoT security management is a challenging problem. The devices may be compromised for a variety of reasons. In this context, it is useful to cast an IoT environment as a complex network that is intended to support a mission rather than just as a set of connected entities. With this view, the security and survivability of the IoT environment is dependent not only on the security of the underlying infrastructure but also on the ability to ensure that unforeseen circumstances, such as, changes to the mission requirements, zero day attacks, and unpredictable human errors in interactions with the mission is adequately addressed and managed. Additionally, in the worst case, there needs to be provisions for the graceful degradation of mission services by avoiding cascading catastrophic failures, when all defensive measures have failed.

To ensure that such a complex network continues to operate in a survivable manner, it is important to be proactive in understanding and reasoning about evolving threats to the service availability, their potential effects on the mission survivability, and identify ways to best defend against these threats, instead of being reactive. Unfortunately, currently there does not appear to exist a comprehensive algorithmic solution that addresses all aspects of this survivability problem. For example, there are gaps in the research on how to map a qualitative description of a mission's survivability goals into a formal model of survivability. There are gaps in our understanding of how human operators contribute towards the survivability goals of a mission. These are challenging issues and many fall outside the scope of this work. Here, we take the first steps in trying to address some aspects of the survivability problems. We believe that a graph-theoretic analysis of networks have the potential to help with proactive analysis of survivability goals. In this paper, we outline an approach towards doing this using the notion of attack graphs in IoT networks.

Instead of covering the entire spectrum of all types of IoT devices and environments, we illustrate here the problem for Smart Home IoT (SHIoT). An attack on an SHIoT system can take place either by initiating an attack from within the smart environment (that is, an insider or local network attack) or by initiating the attack from an external source (i.e., an outsider or public network attack)[1]. SHIoT devices are more vulnerable to cyber-attacks because they are special purpose internet-connected devices and run tiny operating systems such as INTEGRITY, Contiki, FreeRTOS, and VxWorks, whose security solutions are not entirely robust and once deployed, may not be easily upgradable to ensure security capability against evolving cyber-attacks [2].

In this work, we first elaborate on a graph-based representation and analysis of attack modeling for an off-line assessment. We then address the problem of assessing vulnerability from an attack source to a compromised state by considering the attack graph of an SHIoT system. Towards this, we start with a broader framework for the graph-based approach for attack graphs of an SHIoT system. We represent all possible ways that the SHIoT's mission can be compromised based on our current knowledge of vulnerabilities existing in the system as graph. Nodes in the graph represent states and arcs represent transition from one state to another caused by a vulnerability. Each arc is associated with a probability value that represent the probability of the state transition triggered by an attack. Attack is captured by a path in the graph. Our objective is to determine the easiest way (from an attacker's perspective) to compromise

the SHIoT network's mission, and consequently, determine any actions that could be taken to minimize vulnerability. Owing to probabilistic values, the path vulnerabilities do not have additive cost properties. Thus, an important contribution of our work is to show how this problem can be transformed to a problem of determining the shortest path. Furthermore, we address the problem of fortifying the systemic view of the SHIoT from vulnerabilities from a systems management perspective. For this, we build on the graph-theoretic analysis to tackle this problem through an iterative process. In particular, we identify the weakest arcs in the attack graph that can then help systems administrators to take actions to reduce vulnerability in the attack graph, and to then iteratively fortify the SHIoT system. For our study, we use probabilistic values based on Common Vulnerability Scoring System (CVSS) [3] for vulnerabilities to how this fortification can be assessed on representative graphs.

This work builds on our previous work [4] where we developed a new framework for modeling attacks in SHIoT based on finite state automata (FSA), which has a graph-based representation. An advantage of our approach is that depending on the types of cybersecurity based attacks (such as confidentiality-based or authentication-based) and for the type of network environment, we can generate appropriate FSA-based attack graphs. The finite state automata is intended to help facilitate the attack execution flow by grouping attack vulnerabilities. The FSA based attack model takes the attack behavior as the associated process, classifying the attack entity, then studying the state transfer under the attack behavior, and finally being able to identify attack vulnerabilities.

The rest of the paper is organized as follows. In Section II, we present the overall framework for graph-based representation and analysis for attacking modeling for IoT. In Section III, we show how the worst vulnerability in attack graphs can be determined by using a transformation of the problem and through a shortest-path based method; this is followed by an illustration of the fortification process in Section IV and analysis in Section V. Related works are discussed in Section VI. Finally, we present a summary along with limitations of our work in Section VII. A brief appendix on attack modeling using graph-based finite state automata for SHIoT, which inspired this work, is included; for details on that work, see [4].

## II. Graph-Based Representation and Analysis for Attack Modeling

We cast an IoT environment as a complex network where the intent is to support the mission rather than just as a set of connected entities. We envision the steps to be as follows:

1) Formulate a graph model to capture all possible mission dependencies that are relevant as well as their relationships to the broader mission objectives, which will allow mission survivability related analysis.
2) Identify *graph metrics* and *graph analysis* techniques to answer queries about the resiliency of individual components to security threats, as well as answer questions about the overall mission survivability.

We consider the graph model to help us perform two types of analyses: (i) graph-theoretic and (ii) quantitative. The graph-theoretic analysis is geared more towards what-if queries on survivability. It helps us to identify critical components of the mission, their relationships with each other and to the overall mission objectives. It might also help us identify potential but yet undiscovered attacks on the mission continuity. It also drives the quantitative analysis that may help answer questions about the overall robustness of the mission. Thus, we focus on developing a graph-theoretic analysis framework for the mission based on the above philosophy. We (i) propose graph metrics to evaluate roles played by various arcs in component graphs and identify mission critical arcs, (ii) develop graph metrics that can be used to answer what-if queries on the mission continuity, and (iii) develop efficient techniques to re-evaluate core mission graph and propose enhancement.

We outline three different types of analysis on the mission network:

1) Structural analysis – This will determine which individual components are most critical to the continuity of the mission and if attacked can lead to serious (potentially cascading) failures in the mission. The result of these analyses (there can be different types depending on the chosen metric) will provide different ranked sets of critical arcs representing different aspects of mission continuity (what are those aspects of mission continuity), and will serve as the basis for making quantitative analysis.
2) Vulnerability analysis – This analysis will use the ranked set of mission critical arcs and determine which cyber vulnerabilities on these critical arcs can be exploited resulting in the compromise of the mission. The analysis is similar to a traditional attack graph based analysis but will also perform newer analysis that will allow us to fortify the environment to provide enhanced survivability to attacks. For this, we need to perform quantitative analysis on the resulting graph.
3) Fortification analysis – The purpose of this analysis is to study the impact of different component's compromise on the overall mission and how to strategically address vulnerabilities of different arcs in an attack graph. This helps one to determine strategies for mission survivability towards fortification of the system.

The above thought process led us to the work proposed in this paper wherein we employ an iterative technique to propose fortification.

## III. Vulnerability Analysis through an Algorithmic Approach

Our vulnerability analysis for an attack graph is based on determining the worst vulnerable path in an attack graph; see Appendix for an illustration on how an attack graph is generated for an SHIoT system. Since the vulnerability of an arc in the graph is represented as a probabilistic value, the usual shortest

path based approach cannot be used. Thus, we present an approach to tackle the problem.

Consider an attack graph $\mathcal{G}$ of $N$ nodes in which $v_{ij}$ represents the vulnerability probability of an arc associated with attack $S_i \rightarrow S_j$, where $0 < v_{ij} \leq 1$, in this attack graph. If two states $S_i$ and $S_j$ are not connected, then $v_{ij}$ is assumed to be zero.

Given $v_{ij}$, the vulnerability of path $p$ from state $S_i$ to state $S_j$ is given by

$$v_p^{(i,j)} = 1 - \prod_{(i',j') \in \mathcal{P}_{ij}} (1 - v_{i'j'}) \tag{1}$$

where $\mathcal{P}_{ij}$ is the path consisting of the set of arcs $(i', j')$ for path $p$ from state $S_i$ to state $S_j$. Thus, the problem of finding the most vulnerable path in a graph between two states $S_i$ and $S_j$ among the set of paths $\Omega$ may be written as

$$\max_{p \in \Omega} v_p^{(i,j)} \tag{2}$$

Observe that $v_p$ has non-additive properties in terms of arc vulnerability probabilities. Thus we cannot directly apply a shortest path algorithm based on the vulnerability probabilities.

Instead, we bank our approach on another observation. The complement of vulnerability for an arc is reliability where we denote the reliability of $S_i \rightarrow S_j$ for an arc to be $r_{ij} = 1 - v_{ij}$. While there has been work on determining the most reliable path in a graph ([5]), the problem of finding the most vulnerable path has not been explored.

Now, given $r_{ij}$, the vulnerability of path $p$ in (1) can be written as

$$v_p^{(i,j)} = 1 - \prod_{(i',j') \in \mathcal{P}_{ij}} r_{i'j'} \tag{3}$$

Now we introduce the term $w_p$ to be $1 - v_p$, i.e., we can rewrite (3) as

$$w_p^{(i,j)} = \prod_{(i',j') \in \mathcal{P}_{ij}} r_{i'j'} \tag{4}$$

Note that $w_p$ is not the reliability of path $p$.

Based on $w_p$, we can write (2) as the following equivalent problem

$$\min_{p \in \Omega} w_p^{(i,j)} \tag{5}$$

Since (3) has product terms, for the minimization problem (5) we cannot directly apply a shortest path algorithm. On the other hand, taking logarithm of both sides in (4), we can write

$$\log w_p^{(i,j)} = \log \left( \prod_{(i',j') \in \mathcal{P}_{ij}} r_{i'j'} \right) = \sum_{(i',j') \in \mathcal{P}_{ij}} (\log r_{i'j'}) \tag{6}$$

Thus, we can now solve the minimization problem (5) by using the arc weight to be $\log r_{i'j'}$ since we now have additive properties of path in terms of arc cost $\log r_{i'j'}$. We do still have an additional issue to address. Since $0 \leq r_{i'j'} \leq 1$, the term $\log r_{i'j'} < 0$, i.e., in the log-space, the arc weights are always negative. Recall that the attack graph we described has acyclic property, which means that we can apply Bellman-Ford algorithm for arcs with negative weights $\log r_{i'j'}$ [6].

To summarize, our overall approach to determine the most vulnerable path in an attack graph is as follows from the initial state $S_1$ to the compromised state $S_{comp}$:

- Instead of arc vulnerability $v_{ij}$, use the transformed term $\log(1 - v_{ij}) = \log r_{ij}$ for each arc as the abstracted arc weight.
- Instead of solving (2), solve (5) by considering the arc weight as $\log r_{ij}$ using the Bellman-Ford shortest path algorithm on the acyclic attack graph.

This is summarized in Algorithm 1.

**Algorithm 1** Vulnerability Analysis: Determining Worst Vulnerability in an $N$-node Attack Graph from state $S_1$ to the compromised state $S_{comp}$

**Require:** Input: Attack graph $\mathcal{G}$ of $N$ nodes with arcs associated with attack $S_i \rightarrow S_j$ and their vulnerabilities $v_{ij}$

$D_{1,1} \leftarrow 0$

for ($k = 2$ to $k = N - 1$) do

$D_{1,k} \leftarrow \infty$

for ($h = 0$ to $N - 1$) do

$D_{1,comp} \leftarrow \min_{\forall k \neq comp} \left\{ D_{1,k} + \log(1 - v_{k,comp}) \right\}$

Update $p$

endfor

return $F \leftarrow D_{1,comp}$ (cost), $p$ (the most vulnerable path)

## IV. Fortification Process

The Fortification process builds on the vulnerability analysis discussed above. Our approach on fortification is based on first identifying the weakest arc on the most vulnerable path of the attack graph. We assume that once we know this, we can take measures to reduce its weakness through efforts such as any software updates to reduce its vulnerability.

We assume, in this study, that we can do this improvement in a certain boosting value on the weakest arc on the most vulnerable path. We then re-run the vulnerability analysis on the attack graph with this change in the arc weight due to improvement. We continue this process of improvement iteratively until a desirable threshold on fortification is attained. Since in our scenario, no arc can have $v_{ij}$ below 0.0, we added a condition in this iterative process to check for this possibility. This process is captured in Algorithm 2. In practice, it may not be possible to reduce vulnerability on every arc of the attack graph. This variation can be easily captured in our fortification process by marking such arcs as not candidates for boosting.

## V. Analysis on Vulnerability and Fortification

Our fortification process is assessed in representative attack graphs to quantify the number of iterations needed to reach a particular fortification threshold. Note that our process is quite generic and can be used for a wide range of attack graphs for vulnerability assessment beyond the realm of SHIoT.

**Algorithm 2** Fortification Process

**Require:** Input: an attack graph $\mathcal{G}$ with $N$ nodes with arc vulnerabilities $v_{ij}$
**Require:** Input: Fortification threshold: $F_{threshold}$
**Require:** Input: Boost parameter: $B$
Initialize: $F_{now} = 1.0$
$F_{now}, p_{now} \leftarrow$ Algorithm 1
While ($F_{now} \geq F_{threshold}$) do
$(i', j') \leftarrow \text{argmax}_{(i,j) \in p_{now}} v_{ij}$
$v_{i'j'} = \max\{v_{i'j'} - B, 0\}$
$F_{now}, p_{now} \leftarrow$ Algorithm 1
end While
return # of iterations to reach $F_{threshold}$

### A. Determining arc vulnerability

Our approach for attack model vulnerability is based on the probabilistic estimation of arc's vulnerability on the attack graph. To compute the probability of an attack arc, the probability of success needs to be estimated while an attacker exploits a vulnerability exploitation. We use the metrics defined in VCE database Common Vulnerability Scoring System in this paper to evaluate the attack probability. CVSS is the most commonly used vulnerability scoring system and it is supported by the US national vulnerability Library (NVD) [7]. It comprises three distinct groups of metrics such as base, temporal, and environmental. The base metrics measure the intrinsic characteristics of a vulnerability with two subscores: (1) the exploitability score, composed of the access complexity and authentication (AU) occurrences and (2) the impact score, expressing the potential damage on confidentiality(C), integrity, and availability(AC). The temporal metrics measure dynamic aspects of a vulnerability in the environment around the smart home. The environmental metrics measure two aspects of impact that are dependent on the environment surrounding the smart home. More information on CVSS metrics and their scoring computation can be found in the CVSS documentation [3].

In this work, we considered only the base metrics score such as authentication, confidentiality, and access control in the analysis. Since this paper focuses on the vulnerability probability assessment of the smart home network system, in order to simplify the problem, we do not consider the temporal and environment metrics group. The Base Score formula depends on sub-formulas for Impact Sub-Score (ISS) and Exploitability, which are defined below:

$$\text{ISS} = 1 - [(1 - \text{Confidentiality}) \times (1 - \text{Integrity}) \times (1 - \text{Availability})]$$

Given the vulnerability exposure information (CVSS attributes), the probability of vulnerability $v$ of an arc $(i, j)$ is computed from CVSSs Exploitability subscore as the following:

| Metric | Metric Value | Numerical Value |
|---|---|---|
| Attack Vector | local | 0.7 |
| | remote | 1.0 |
| Attack Complexity | high | 0.8 |
| | low | 1.0 |
| Privileges Required | required | 0.6 |
| | not-required | 1.0 |
| User Interaction | none | 0.8 |
| | required | 0.6 |
| Confidentiality | partial | 0.7 |
| | complete | 1.0 |
| Integrity | partial | 0.7 |
| | complete | 1.0 |
| Availability | partial | 0.7 |
| | complete | 1.0 |

TABLE I: Base metric elements and values of the base metric group based on the CVSS [3]

$$v = \text{ISS} \times \text{AttackVector} \times \text{AttackComplexity} \times \text{PrivilegesRequired} \times \text{UserInteraction}$$

All metrics are determined under the assumption that the attacker has already located and identified the vulnerability. Thus, the analyst need not consider how the vulnerability was identified. Additionally, many different sectors' individuals will be scoring vulnerabilities, such as software vendors, vulnerability bulletin analysts, and security product vendors. However, vulnerability scoring is expected to be skeptical of the individual and their organization. For example, the privilege required metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability, and this metric value can be categorized as none (1.0), and required (0.6)[3].

### B. Analysis

For our what-if analysis, we consider the 9-node attack graph for SHIoIT shown in Figure 4 in the Appendix. In addition, we used a 15-node attack graph (Figure 1) from [8], and generated 26-node and 50-node attack graphs.

We applied our fortification process on these graphs. For the boost parameter, we started with 10% improvement and conducted the study till 25% for an arc. For the fortification threshold, we used values 0.4 to 0.7. The results for 9-node, 15-node, 26-node and 50-node attack graphs are presented in Tables II, III, IV, and V, respectively. As can be seen from the results, a higher fortification threshold and larger attack graphs require more iterations to reach the goal, while small boost steps also require a higher number of iterations. A higher number of iterations reflects that more efforts are needed for fortification.

This type of what-if analysis is helpful in systems management for system administrators as they can strategically allocate resources towards fortification of an SHIoT system. Secondly, a real resource can be associated with each iteration to determine the overall cost of such fortification.

| Increment \Threshold | P(0.4) | P(0.5) | P(0.6) | P(0.7) |
|---|---|---|---|---|
| 10% | 152 | 179 | 197 | 216 |
| 15% | 104 | 111 | 126 | 150 |
| 20% | 73 | 89 | 93 | 102 |
| 25% | 60 | 66 | 73 | 80 |

TABLE II: Iterations required for attack graph with 9 nodes shown in Figure 4

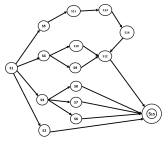

Fig. 1: An attack graph with 15 nodes

| Increment \Threshold | P(0.4) | P(0.5) | P(0.6) | P(0.7) |
|---|---|---|---|---|
| 10% | 246 | 274 | 312 | 356 |
| 15% | 186 | 205 | 278 | 303 |
| 20% | 154 | 178 | 211 | 284 |
| 25% | 112 | 136 | 187 | 223 |

TABLE III: Iterations required for an attack graph with 15 nodes shown in Figure 1

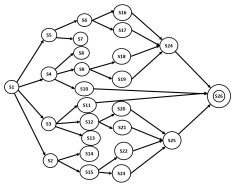

Fig. 2: An attack graph with 26 nodes

| Weight Increment | P(0.4) | P(0.5) | P(0.6) | P(0.7) |
|---|---|---|---|---|
| 10% | 468 | 494 | 531 | 588 |
| 15% | 411 | 432 | 467 | 498 |
| 20% | 392 | 421 | 458 | 487 |
| 25% | 368 | 394 | 421 | 464 |

TABLE IV: Iterations required for an attack graph with 26 nodes shown in Figure 2

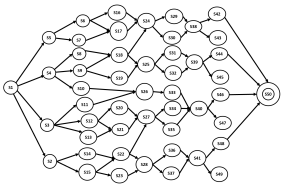

Fig. 3: An attack graph with 50 nodes

## VI. Related Work

In order to understand the IoT security landscape, a general IoT threat model is needed [9]. Several studies have focused on modeling attacks and intrusions with the objective of evaluating various security metrics. Michael and Ghosh [10] employed a finite state machine (FSM) model constructed using system call traces. Costa et al. [11] presented a practical method supported by open source tools that can identify high risk vulnerabilities present in smart home IoT devices. Wang et al. [7] focused on vulnerability assessment of industrial internet of things and proposed a vulnerability graph model based on attack graph and a vulnerability algorithm based on maximum loss stream. Chen et al. [12] combined an analysis of data on security vulnerabilities and a focused source-code examination to develop a finite state machine (FSM) model to describe and reason about security vulnerabilities. Zhang et al. [13] presented an attack modeling method based on system states aggregation. This work combines finite automaton with the changes of system state caused by the attack entity, building the attack model of finite automaton, making an analysis of the model algorithm, and making a quantitative evaluation on attack cost, the success rate, exposure rate and evaluating severity of attack on cyberspace. Davis et al. [14] mentioned that the vulnerability studies of IoT devices to date are not all inclusive and, in some cases, target well-known vendors or devices. For a formal model for survivability that uses probabilistic model checking, see [15] for wireless sensor networks.

As we can see, what is missing is how to analyze attack graphs for worst-case survivability and how to fortify for IoT security management.

| Weight Increment | P(0.4) | P(0.5) | P(0.6) | P(0.7) |
|---|---|---|---|---|
| 10% | 578 | 597 | 628 | 661 |
| 15% | 523 | 564 | 592 | 628 |
| 20% | 481 | 512 | 568 | 594 |
| 25% | 452 | 489 | 529 | 576 |

TABLE V: Iterations required for an attack graph with 50 nodes shown in Figure 3

## VII. Summary and Limitations

Inspired by investigating the vulnerability of the threat and attacker motive in an SHIoT environment, we presented a graph-based framework for attacks in IoT security. In this framework, an attack graph is first represented through Finite-state automata for which we present vulnerability analysis, followed by a fortification process to enhance the overall system. In particular, we showed how vulnerability analysis can be done using a Bellman-Ford algorithm with modified arc weights, and how a fortification process can use this vulnerability analysis through an iterative process. It may be noted that our fortification process can be used for any acyclic attack graphs, not just limited to SHIoT. We then studied our approach on representative attack graphs.

Despite presenting a new way to look at IoT security management through worst attack vulnerability, there are a number of limitations of our work. Our approach may not work for all types of IoT security management. In particular, the worst attack vulnerability may not be the most important issue for certain IoT security management. For instance, instead of

| States Description | Vulnerability | Impact | CVE# |
|---|---|---|---|
| $S_1$ (adversary) - trying to access user's phone | Malware, Phishing | Take control of device | CVE-2021-27612 |
| $S_2$ (User's phone) - trying to connect to a public wi-fi medium | Malware, Synchronization, Buffer Overflows, Phishing | Monitor user's online activities, take control of device | CVE-2021-23977 |
| $S_3$ (Public Wi-Fi) - Accessing the IoT application | possibility of joining a fake or rogue Wi-Fi hotspot | allows cyber attackers to monitor users online traffic | CVE-2018-11477 |
| $S_4$ (Dongle/Portal router) - Accessing IoT application | It becomes discoverable to malicious attacker seeking to exploit connection | allows attackers to sniff on network traffic and inject malicious scripts | CVE-2019-13053 |
| $S_5$ (IoT application) - Accessing the web services | Infect associated smart application with malware | User credentials and private data could be stolen | CVE-2019-1698 |
| $S_6$ (Web services) - Accessing the home Gateway Router | SQL Injection, Cross Site Scripting | user data can be modified (Insert/Update/ Delete) | CVE-2021-3340 |
| $S_7$ (Home Gateway Router) - trying to compromise the IoT device | Uses UPnP to modify firewall settings, to reconfigure routers, and opens ports to IoT devices | Botnet creation as part of larger attacks such as DDoS | CVE-2009-2257 |
| $S_8$ (Compromised IoT device) | Add fake/Sybil nodes to network and spread malware | Affect the whole network system, Increases the power consumption of sensor nodes | CVE-2019-1957 |

TABLE VI: Network Transition State Vulnerabilities from [4] (CVE from [16])

probability, the cost to fortify an arc may be important or a combination of both probability and cost with different priorities may be important. Furthermore, while we used CVSS to estimate the probability of an arc in our illustration, in practice, this could be very difficult to determine. Also, estimates of the probability could be erroneous, and thus, a straight-forward application of our worst attack vulnerability approach may lead to error propagation. These issues would be important to consider in future research.

## REFERENCES


[1] J. Pacheco and S. Hariri, "Iot security framework for smart cyber infrastructures," in *Foundations and Applications of Self* Systems, IEEE International Workshops on*. IEEE, 2016, pp. 242–247.

[2] "Smart home security: Security and vulnerabilities." [Online]. Available: https://www.wevolver.com/article/smart-home-security-security-and-vulnerabilities

[3] "Common vulnerability scoring system version 3.1: Specification document." [Online]. Available: https://www.first.org/cvss/

[4] F. James, I. Ray, and D. Medhi, "Situational awareness for smart home iot security via finite state automata based attack modeling," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2021, pp. 61–69.

[5] M. Roosta, "Routing through a network with maximum reliability," *J. of Math. Analysis & Apps.*, vol. 88, no. 2, pp. 341–347, 1982.

[6] J. Fakcharoenphol and S. Rao, "Planar graphs, negative weight edges, shortest paths, and near linear time," *J. Comp. & Sys. Sci*, vol. 72, pp. 868–889, 2006.

[7] H. Wang, Z. Chen, J. Zhao, X. Di, and D. Liu, "A vulnerability assessment method in industrial internet of things based on attack graph and maximum flow," *IEEE Access*, 2018.

[8] J. Zeng, S. Wu, Y. Chen, R. Zeng, and C. Wu, "Survey of attack graph analysis methods from the perspective of data and knowledge processing," *Security and Communication Networks*, 2019.

[9] D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska, and W. Xu, "Automated Security Test Generation with Formal Threat Models," *IEEE Transactions on Dependable and Secure Computing*, 2012.

[10] C. C. Michael and A. Ghosh, "Simple, State-Based Approaches to Program-Based Anomaly Detection," *ACM Trans. Inf. Syst. Secur.*, 2002.

[11] L. Costa, J. Barros, and M. Tavares, "Vulnerabilities in iot devices for smart home environment," in *Proceedings of 5th ICISSP*, 2019.

[12] S. Chen, Z. Kalbarczyk, J. Xu, and R. Iyer, "A Data-Driven Finite State Machine Model for Analyzing Security Vulnerabilities," in *Proc. 2003 International Conference on Dependable Systems and Networks*, 2003.

[13] Z.-W. Zhang and Y. Yun-Tian, "Research of attack model based on finite automaton," in *2012 National Conference on IT and CS*, 2012.

[14] B. D. Davis, J. C. Mason, and M. Anwar, "Vulnerability studies and security postures of iot devices: A smart home case study," *IEEE Internet of Things Journal*, 2020.

[15] S. Petridou, S. Basagiannis, and M. Roumeliotis, "Survivability analysis using probabilistic model checking: A study on wireless sensor networks," *IEEE systems journal*, vol. 7, no. 1, pp. 4–12, 2012.

[16] "Common vulnerabilities and exposures (CVE)," https://www.cve.org/.


## APPENDIX: ATTACK MODELING USING GRAPH-BASED FINITE STATE AUTOMATA FOR SHIOT: A BRIEF REVIEW

Our graph-based FSA approach for SHIoT, previously reported in [4], was the impetus for the work in this paper for use of the framework in Section II and our approach. Briefly, a finite state automata for SHIoT attack (FSAA) consists of the tuple $(\mathcal{S}, \Sigma, \delta, S_0, \mathcal{F})$. where $\mathcal{S}$ is a non-empty finite set of states representing various states of interest in modeling the attack, while $S_0$ represents the initial state when no attack had been launched. The finite set of input symbols is given by $\Sigma$ and a transition is denoted by $\delta$. The set of terminal states, which can be one of the potential attack success states or attack failure states, is denoted by $\mathcal{F}$ ($\subseteq \mathcal{S}$).

A confidentiality based attack from [4] is shown in Figure 4 with the set of states vulnerabilities along with a Common Vulnerabilities and Exposures (CVE) number [16] in Table VI. See [4] for additional details.

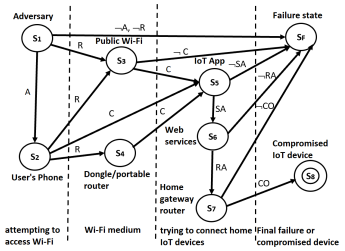


Fig. 4: Confidentiality based 9-node attack graph