IoT-PEN: A Penetration Testing Framework for IoT

Geeta Yadav
Khosla School of Information Technology
IIT Delhi, India
geeta@cse.iitd.ac.in

Kolin Paul
Department of Computer Science & Engineering
IIT Delhi, India
kolin.paul@cse.iitd.ac.in

Alaa Allakany
Cybersecurity Center, Kyushu University, Japan &
Faculty of Computers and Information, Kafrelsheikh University, Egypt
alaa_83moh@yahoo.com

Koji Okamura
Research Institute for Information Technology
Kyushu University, Japan
oka@ec.kyushu-u.ac.jp

***Abstract*—With the horizon of 5^{th} generation wireless systems (5G), Internet of Things (IoT) is expected to take the major portion of computing. The lack of inbuilt security and security protocols in cheap IoT devices give privilege to an attacker to exploit these device's vulnerabilities and break into the target device. IoT network security was initially perceived from the perspective of a single, or a few attacks surface only. However, attacks like Mirai, Wannacry, Stuxnet, etc. show that a cyber attack often comprises of a series of attacks on vulnerabilities of victim devices to reach the target device. Penetration testing is generally used to identify the vulnerabilities/ possible attacks on traditional systems periodically. A timely fix of these vulnerabilities can avoid future attacks. Traditional penetration testing methods focus on isolated and manual testing of a host that fails to detect attacks involving multi-hosts and multi-stages.**

In this paper, we introduced first-of-its-kind, *IoT-PEN*, a Penetration Testing Framework for IoT. The framework consists of server-client architecture with "a system with resources" as server and all "IoT nodes" as clients. *IoT-PEN* is an end-to-end, scalable, flexible, and automatic penetration testing framework for IoT. *IoT-PEN* seeks to discover all possible ways an attacker can breach the target system using target-graphs. It constructs prerequisite and postconditions for each vulnerability using the National Vulnerability Database (NVD). We also demonstrated that even if an individual system is secure under some threat model, the attacker can use a kill-chain (a sequence of exploitation of multiple vulnerabilities on different hosts) to reach the target system.

***Index Terms*—Internet of Things Security, Penetration Testing**

I. Introduction

5^{th} generation wireless systems (5G) is expected to provide a Internet of things (IoT) specific connectivity interface [1]. IoT provides a platform to connect the physical devices and everyday objects over the internet. Most of the IoT devices are using cheap sensors/ actuators, which are likely to be very insecure. Over time, IoT is creating a completely new and complex set of problems for the security community. Different from enterprises system, securing large scale, resource constraint IoT devices is a challenging task. If we analyze some of the recent attacks on IoT, they involve a series of attacks on the victim nodes e.g. WannaCry, in 2017, was initially carried out using a malicious PDF (malware embedded), that further exploited the vulnerability of SMBv2 protocol that executed dropper code and made a connection to an unregistered IP address. Also, it created a mssecsvc2.0 service and made an entry in the registry. It encrypted local files and asked a ransom note for $300 in Bitcoin [2]. DDoS attack on the Dyn DNS company and French service provider OVH in 2016 by exploiting the default passwords and the outdated TELNET service to get control of millions of web cameras which were manufactured by a specific Chinese company exhibited that an attacker can turn the Internet of Things into the Internet of Vulnerabilities (IoV) [3]. In 2010, a very sophisticated attack Stuxnet was reported on Iranian nuclear centrifuges [4]. It entered the network via a USB stick (air-gapped network) and replicated itself to all connected machines running Microsoft windows jumping across the air-gapped network. It then searched specific version of Siemens Step7 Industrial Control Systems (ICS) and modified the Program Logic Controllers (PLC). The main target for Stuxnet was Siemens Step7 system, and it would sit silently on other systems.

A brief study of Wannacry, DDoS, Stuxnet shows that a series of unrelated attack events that are logically connected can provide an attack-path to the target node, even though there is no direct path to the target node, pointing to the idea of usable security for large scale IoT network.

Definition 1: In Secure System of Systems, even if each system has some vulnerabilities, absence of a attack-path from the compromised node to the target node implies that the target system has met condition of usable security.

These multi-stage attacks exploited different vulnerabilities at different stages, different systems indicating that securing a system randomly in the system of systems can not avoid these attacks in the future. The system administrator needs to secure the end-to-end system. Penetration testing is performed on traditional systems, to identify all possible vulnerabilities on each host manually and in an isolated way, that fails to identify multi-stage and multi-host attacks on the target system.

Target graphs illustrate possible ways in which an adversary can exploit a system's multi-stage multi-host vulnerabilities to

This research was supported by DST-JST project "Security in the IoT space", DST Grant Number RP03321, JST Grant Number JPMJSC16H3.

break into the target. System administrators usually evaluate system security using Target graphs to identify the source and sequence of exploitation. It also recommend the security measures that should be taken to defend the complete system. Each path in a target-graph is a sequence of exploits that leads to an undesirable state, e.g., obtaining administrative access to a critical host.

IoT-PEN uses the NVD database, an updated and comprehensive database, to identify the possible vulnerabilities on a particular version of the software. It maintains examined information about software and hardware vulnerabilities of different domains [5]. It considers product name, product version, and vender. It indexes vulnerabilities to Common Vulnerability Enumeration (CVE) [6] Ids and enables automation of vulnerability management.

State-of-the-art penetration tools, i.e., Netsparker, Acunetix, Probably, BackTrack, Idappcom Metasploit, Nessus, etc. provide the functionality only to the subscribed users. Open-source tools, i.e., Wapiti, ZAP (Zed Attack Proxy), Vega, W3af, etc. have minimal functionality and are targeted to a particular threat model. Almost all of these techniques are manual and perform isolated Pentesting. These solutions fail in case of IoT, where the devices are quite heterogeneous. Performing penetration testing manually on a large number of IoT devices is a challenging task for system administrators. Thus, highlighting the urgent need of new algorithms, tools, frameworks for securing these resource constraints devices.

To overcome these weaknesses, we propose *IoT-PEN*, which is an automatic, flexible, and End-to-End penetration testing framework. The framework consists of a server-client architecture with "a system with resources" as a server and all "IoT nodes" as clients. In *IoT-PEN*, a customized script runs over a system of systems and finally report possible exploitation. Flexible *IoT-PEN* specify plug and play concept for penetration-testing. *IoT-PEN* consists of different modules to consider the heterogeneity of IoT systems, a user can select the required modules, and a customized framework is generated. Apart from this, the main motive of *IoT-PEN* is end-to-end penetration framework. Currently, proposed approaches focus on the individual system or component testing, which fails to detect multi-host, multi-stage vulnerabilities.

A. Contributions

Our major contributions are as follow:

1) *IoT-PEN* a novel flexible, automatic, and E2E Penetration testing framework for the resource constraint IoT devices. It identifies all possible paths from where an attacker can reach to the target node.
2) We propose a novel mechanism to perform the penetration testing particularly for IoT system using client-server architecture without the intervention of the system administrator. Also, it is capable of identifying the insider as well as possible outsider attacks.
3) We evaluated *IoT-PEN* performance and scalability in terms of execution time vs. number of nodes, of number of attack-paths generated vs. the number of nodes in the network, number of attack paths vs. vulnerabilities on the nodes.

In the next section, we discuss the problem setup with real-time reported vulnerabilities on a Philips Hue system. Then we discuss related work done in Section III followed by the design and implementation of *IoT-PEN* in Section IV. *IoT-PEN* performance evaluation is done in Section V.

II. Problem Setup

An IoT system consists of sensors, actuators, bridges, gateways, routers, etc. Each component is itself a complete system. Therefore, we can consider an IoT network equivalent to a system of systems. The proposed approaches in the literature mainly focus on penetration testing of the individual system separately using a fixed list of brute force attacks. These approaches are inefficient in terms of time, cost, and also restricted to the availability of exploits.

A. IoT Example

Here, we consider an example of real-time small IoT system, e.g., Phillips Hue bulbs system which consists of a smart bulb, Hue bridge, IoT gateway, server, mobile application, and communication links. As per the Definition 1, even if each node has different vulnerabilities, the absence of any target-path to the target node (bulb) is sufficient to define the target as a secure system. In Fig. 1, we demonstrated an example of a Phillips Hue IoT system, where each node has at least one vulnerability. In the current scenario, we consider that the attacker wants to get control of the smart bulb. The attacker can get the cloud API authentication key by exploiting vulnerability CVE-ID CVE-2014-0220 on the mobile phone on which the Philips Hue app is running. The attacker login to cloud as an authenticated user exploiting vulnerability CVE-ID CVE-2015-2883. Now, she can weave cloud web service for connected device/application settings followed by access of execution log exploiting vulnerability CVE-ID CVE-2019-4047. Further, it searches for connected devices and logs in using default user login-passwords and able to get user-sensitive data exploiting CVE-ID CVE-2018-18394 and able to get root access exploiting with CVE-ID CVE-2018-18392. Further, she modifies the data being transmitted to bulb exploiting vulnerability with CVE-ID CVE-2017-14797. It specifies that even though an attacker is unable to attack the bulb control directly, but it can successfully take control of the bulb by exploiting multi-host, multi-stage vulnerabilities. It is not possible to identify these categories of attacks using individual system penetration testing only.

III. Related Work

There have been few kinds of research work proposed for penetration testing. A manual penetration approach proposed by Denis et al. [7] performs individual system penetration testing using the Kali Linux for smartphones, Bluetooth, etc. PENTOS [8] is a pentest tool specially designed for IoT devices. It is not applicable to heterogeneous IoT nodes. Moreover, the framework test the communication network for a few

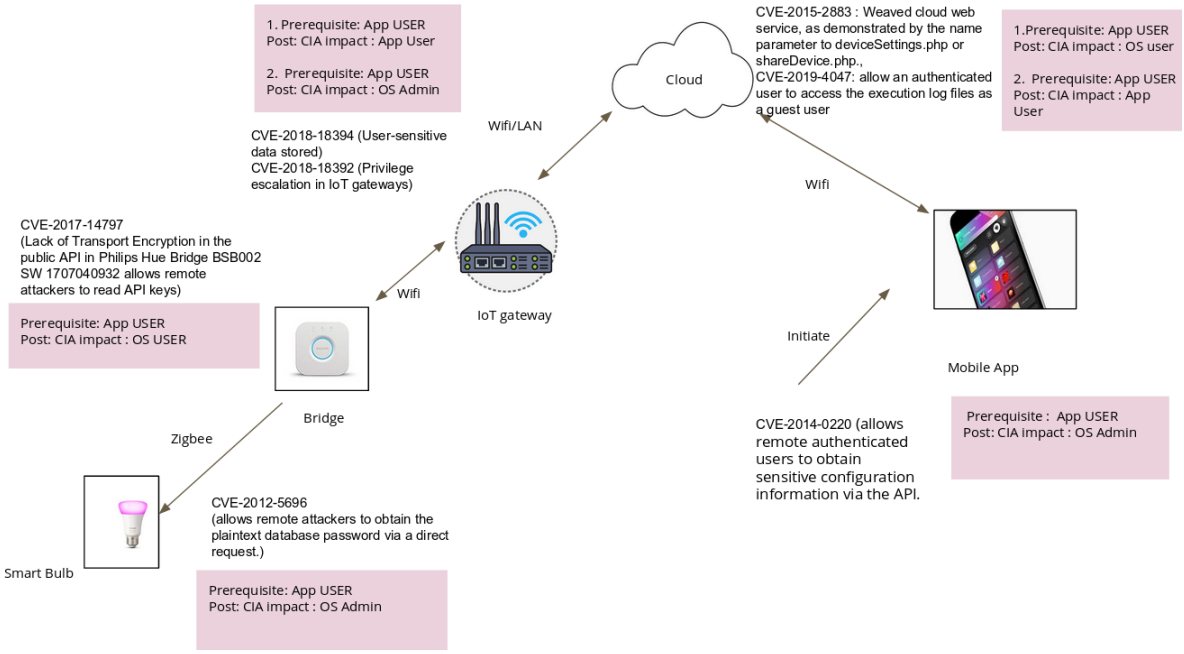

Fig. 1. Multi-host, multi-stage vulnerability exploitation example with pre and post condition

specific attacks. Xueqiu et al. in [9] propose an automatic generation algorithm combining the penetration graph generation method with the CVSS information together. However, they did not evaluate their framework in terms of scalability and IoT applicability. Aksu et al. in [10] focus on the prerequisite and post-condition rules of vulnerabilities generation by rule-based techniques and using machine learning. AlGhazo et al. in [11], proposed a framework which enlists set of all possible sequences in which atomic-level vulnerabilities can be exploited to compromise certain system-level security given the networked system description. Ou et al. in [12] presented a Prolog programming language based network security analyzer MulVal. Almost all the works, focus on isolated penetration testing of each system and are manual and not scalable for complex IoT networks.

IV. Design and Implementation

In *IoT-PEN*, we follow the single responsibility principle, as we break the framework into independent micro-services, capable of running on their own.

A. Target Graph Structure

We use the following nomenclature along with the corresponding definitions.
Locality of the attacker: {Network, adjacent, local, physical}
Authentication = { None, Required}
Privilege required = Privilege gained = {none, user, root}
CPE: {Application, Operating System, Hardware}.

Definition 2: Target graph is a digraph $G_A = (P, Q)$, where P denotes the set of nodes and Q denotes a set of edges. $P := \{v = d : d \epsilon D\}$, where D is a list of devices. An edge (p_i, p_j, vul_k),<Source Node, Target Node >denote a set of implication that p_i can be exploited by attacker at p_j vulnerability $vul_k \epsilon Vul$. $Vul := \{vul_1, vul_2, ..., vul_m\}$ is a set of all vulnerabilities from NVD dataset.

Definition 3: A vulnerability is considered as a CVE entry, and represented by tuple with three elements $<CVE_{ID}$, prerequisite , Postcondition $>$.

Definition 4: Prerequisite of a vulnerability $vul_k \epsilon Vul$, where $Vul := \{vul_1, vul_2, ..., vul_m\}$, is a function of {Vulnerability Description, Attack Vector (Locality), Authentication, Priviledge, CPE }. Prerequisite denotes the required attacker priviledge to enter into the system.

Definition 5: Post-condition of a vulnerability $vul_k \epsilon Vul$ where $Vul := \{vul_1, vul_2, ..., vul_m\}$, is a function of {Vulnerability Description, Impact of vulnerability, Priviledge, CPE }. Post-condition denotes the acquired attacker priviledge.

Definition 6: An edge (p_i, p_j, vul_k), denotes that vulnerability vul_k of node p_i can be exploited by node p_j in the target graph. For this, p_j should satisfy the locality required to exploit the vulnerability and post-condition of node d_j should be sufficient to exploit vul_k on node d_i.

B. Implementation

The framework has the following parts:
Input : The input given to the framework is network topology. Network topology contains the unique id of all devices(in our case, we are taking Internet Protocol (IP) addresses) and the direct reachability in adjacency matrix form.
S1 : Pentesting setup installation : IoT-PEN follows a server-client architecture. All the client nodes and server are prepared for penetration testing. A patch (installation of the

vulnerability scanner tools, set up to listen to server command and act accordingly) is applied to all the nodes in the topology.

S2 : Get current state information of each node : As most of the IoT devices use MQTT protocol for communication to the server, *IoT-PEN* also uses MQTT for server-client communication. Each IoT node state information in XML format (generated by the vulnerability scanner tools) is collected by the server node for further processing automatically.

S3 : Extract CPE from .xml file generated by Nmap : In this step, it parses the XML files received at the server to extract the current state of nodes in terms of operating system and applications/ services version and vendor information followed by generating CPE. Concurrently, It locally sync the NVD dataset for fast processing; then the NVD dataset containing CVE-CPE mapping is changed to dataset with CPE-CVE mapping for improving the time complexity.

S4: Pre-Post condition generation for the reported vulnerabilities & Target-graph generation : In S3, we have collected all the possible vulnerabilities on each system. Then, we find out the prerequisites and post-conditions for each reported vulnerability by using NVD database considering Definition 4 and Definition 5. We followed a Modified approach of rule-based prerequisite and post-condition generation proposed in [10]. The modified approach for prerequisite and post-conditions for these vulnerabilities are based on rules specified in Table I, Table II and Table III. The prerequisites and post-conditions for IoT example in Section II is shown in Fig. 1.Then we generated target paths using Algorithm 1. The attack-path generated w.r.t. IoT example in Section II is shown in Fig. 2.

TABLE I
RULES FOR GENERATING PREREQUISITE USING LOCALITY, AUTHENTICATION AND PRIVILEGE

Locality of attacker	User interaction	Priviledge	CPE	prerequisite
-	-	None	-	None
Local	-	High	OS	Admin OS
Local	None	High	App	Admin OS
Network	Required	High	OS	Admin OS
Local	-	Low	OS	User OS
Local	None	Low	APP	User OS
Network	Required	Low	OS	User OS
Local	Required	High	App	Admin App
Network	Required	High	App	Admin App
Local	Required	Low	App	User App
Network	Required	Low	App	User App

0

S5 : Analysis of attack-paths & Recommendations The last stage of IoT-PEN is responsible for generating recommendations in terms of “All the possible paths through which an attacker can attack the target system.”, “Optimization techniques for the Target-paths”.

V. PERFORMANCE & SCALABILITY

We measured IoT-PEN performance on Ubuntu Linux machine (Version - 16.04.1 LTS (Xenial Xerus)), 16GB RAM. For experiments, we randomly created nodes and network topology. We measured time-taken by individual steps of

TABLE II
RULES FOR GENERATING PREREQUISITE USING DESCRIPTION FIELD (A-APPLICATION)

Description	CPE	Locality of attacker	Privilege	prerequisite
“allows local administrators” or “allow local administrators” or “allows the local administrator”	-	local	High	Admin OS
“allows local users” or “allowing local users” or “allow local users“ or “allows the local user”	-	local	High	User OS
“remote authenticated admin” or“remote authenticated users with administrative privileges”	A	Network	Low	Admin App
“remote authenticated user” &! “remote authenticated users with administrative privileges”	A	Network	Low	User App

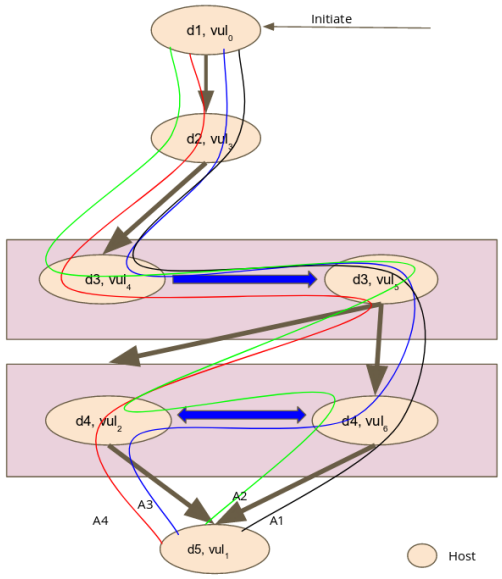

Fig. 2. Generted target graph (A1, A2, A3, A4 are possible attack path. d1, d2, d3, d4, d5 are the devices in IoT example Fig. 1)

IoT-PEN (i.e., Information gathering (Node state), Database Sync, CPE CVE mapping generation, prerequisite and post-conditions generation and finally graph generation) to verify the applicability of IoT-PEN to complex IoT network. The time taken by each step is shown in Table IV.

We repeated each experiment five times and then took average to reduce error in the experimental values. We measured the time taken for information gathering on ten devices, which takes 20.41 seconds on an average. In parallell, we can execute step 2 that locally creates a database for NVD CVEs taking 299.23 seconds the first time, for later steps, we can use a script to sync only the updated data. For generating CPE-CVE mapping and generation of prerequisite and post-condition for each node takes around 53 and 15 seconds respectively. Therefore, for the initial setup, 367.23 seconds are spent for data-synching and converting data to the desired form.

Observing Fig. 3, Fig. 4, we can see that time taken in finding the target-paths to the target node with varying number of nodes in former and the number of vulnerabilities are

TABLE III
RULES FOR GENERATING POST-CONDITIONS USING NVD DATASET. CPE - O (OPERATING SYSTEM), A (APPLICATION) , IMPACT SCORE (5.9 (ALL CIA IMPACT VALUE), [3.4 - 5.9))) (PARTIAL CIA IMPACT), <3.4 (SOME CIA IMPACT VALUE NONE))

Description	Impact Score	CPE	Post-Condition
"gain root" or "gain unrestricted, root shell access" or "obtain root" or "gain privilege" or "gain host OS privilege" or "gain admin" or "obtain local admin" or"obtain admin" or "gain unauthorized access" or "to root" or "to the root" or"elevate the privilege" or "elevate privilege" or "root privileges via buffer overfow"or"unspecified vulnerability" or "unspecified other impact" or "unspecified impact" or"other impacts" or "buffer overflow" or "command injection" or "write arbitrary,file" or "command execution" or"execute command" or "execute root command" or "execute commands as root" or"execute arbitrary" or "execute dangerous" or "execute php" or "execute script" or "execute local" or"execution of arbitrary" or "execution of command" or "remote execution" or "execute code" &! "execute arbitrary SQL"	5.9	ANY	Admin OS
"obtain password" or "obtain credential" or "sniff ... credentials" or"sniff ... passwords" or"steal ... credentials" or "steal ... passwords"or "cleartext credential" or "cleartext password" or "obtain plaintext" or "obtain cleartext" or"discover cleartext" or "read network traffic" or "un-encrypted" or "unencrypted" or"intercept transmission" or "intercept communication" or "obtain and decrypt passwords" or"conduct offline password guessing" or "bypass authentication"	5.9	O	Admin OS
"gain privilege" or "gain unauthorized access" or"unspecified vulnerability" or "unspecified other impact" or "unspecified impact" or"other impacts" or obtain password" or "obtain credential" or "sniff ... credentials" or"sniff ... passwords" or"steal ... credentials" or "steal ... passwords or "cleartext credential" or "cleartext password" or "obtain plaintext' or "obtain cleartext" or"discover cleartext" or "read network traffic" or "un-encrypted" or "unencrypted" or"intercept transmission" or "intercept communication" or "obtain and decrypt passwords" or"conduct offline password guessing" or "bypass authentication" or	3.4 - 5.9	O	User OS
"gain admin" or "obtain admin or obtain password" or "obtain credential" or "sniff ... credentials" or"sniff ... passwords" or"steal ... credentials" or "steal ... passwords" or "cleartext credential" or "cleartext password" or "obtain plaintext" or "obtain cleartext" or"discover cleartext" or "read network traffic" or "un-encrypted" or "unencrypted" or"intercept transmission" or "intercept communication" or "obtain and decrypt passwords" or"conduct offline password guessing" or "bypass authentication" or "SQL injection"	3.4 - 5.9	A	Admin App
"hijack the authentication of users" or "hijack the authentication of arbitrary users" or"hijack the authentication of unspecified victims"	-	-	User App
-	<3.4	-	None

Algorithm 1 IoT-PEN Algorithm

Input:
$Node_{List}$ ▷ List of all the nodes in the network to be pentested.
$Net_{topology}$ ▷ Network topology provided by the system administrator.

procedure PENTEST_NODE($Net_{topology}$)
1) Apply patch to IoT nodes, where patch contains open-source tools installation to get each node state.
2) Server publishes instructions to IoT nodes to scan their current state of operating system and services.
3) IoT nodes perform scanning and share the output file (.xml) to the server = Scan_reports[node]

return Scan_reports[node]
end procedure

procedure E2E_PREMODULE (Scan_reports, $Node_{List}$, $Net_{topology}$)
1) Parse Scan_reports[node] received at server to get CPE for operating system and services for each node = CPE_{list}
2) NVD data have a mapping of CVE to CPE list. We converted it to CPE to CVE list. Also based on the rules specified in Table I, Table II and Table III, for each vulnerability, prerequisite Vul_{Pre} and post-condition Vul_{post} are generated .
3) $vulnerability\_list$[node] = Extract CVE ids w.r.t. CPE list in the CPE_{list} for each node.
4) For each vulnerability j on node i (in $nodevul_{list}$), find prerequisite (PR_{ij}) and post-condition (PC_{ij}) as generated in Step 2.
5) For each node in $Net_{topology}$ find :
   a) $minPriv[node]$ = Minimum privilege needed to exploit any vulnerability on the node.
   b) $maxPriv[node]$ = Maximum privilege gained after exploiting multi-stage vulnerability.
   c) $privgainedmax_{pre}[node]$ = prerequisite corresponding to maximum privilege gained.
   d) $maxfrommin[node]$ = Can maximum privilege achieved by exploitation of vul-chain.

return (Vul_{Pre}, Vul_{Post}, $minPriv$, $maxPriv$, $privgainedmax_{pre}$, $maxfrommin$, $vulnerability\_list$, PC, PR)
end procedure

procedure E2E_MODULE ($Vul_{Pre}, Vul_{Post}, Net_{topology}, minPriv, maxPriv, maxfrommin$, $Target\_node$, $vulnerability\_list$, PC,PR)
1) $current\_node$ = $Target\_node.pop()$, $Target\_paths = []$
2) **while** $current\_node != null$ **do**
3) vul_{list} = $vulnerability\_list[current\_node]$
4) Extract $minPriv[current\_node], maxPriv[current\_node], privgainedmax_{pre}[current\_node], maxfrommin[current\_node]$.
5) Calculate privilege gained after multi-stage vulnerability exploitation .
6) $neighbour_{node}$ = Extract nodes reachable from current_node using $Net_{topology}$
7) **for** each node in $neighbour_{node}$ **do**
8) vul_list = vulnerability[node]
9) **for** each vul in vul_list **do**
10) **if** maxPossiblefrommin[node] is true and vul.locality == current_node.locality **then**
11) **if** $minPriv[current\_node] > PC_{node,vul}$ or $minPriv[current\_node]$ == 0 **then** Target_paths.add(node, current_node, vul)
12) **end if**
13) **end if**
14) **end for**
15) **end for**
16) **end while**

return $Target_{paths}$
end procedure

varying in the later one. Time taken to find target-paths in (1 node, ten vulnerabilities) and (10 nodes, 1 vulnerability) is approximately the same. This is due to IoT-PEN is checking multi-host multi-stage vulnerabilities exploitations. The sudden increase in the number of paths in Fig. 5 may be due to more probability of successful hoping of the attacker from the

TABLE IV
TIME ELAPSED IN VARIOUS STAGES

Step	Stage	Time elapsed
1	Information gathering (Node state)	20.41 sec (For one node)
2	Database Sync	299.23 sec
3	CPE-CVE mapping generation	53 sec
4	Pre-Post conditions generation	15sec

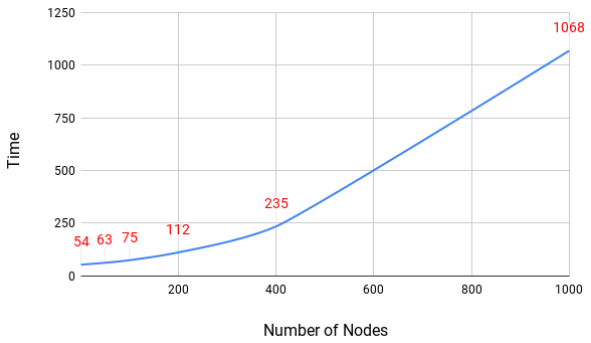

Fig. 3. Running Time vs. Number of nodes (Number of vulnerabilities on each node = 5)

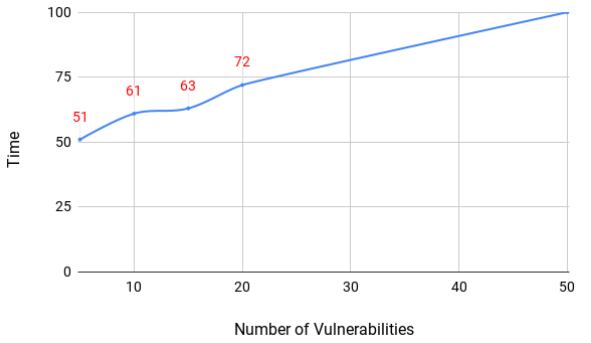

Fig. 4. Running time of IoT-PEN vs. number of vulnerability (Number of nodes = 10)

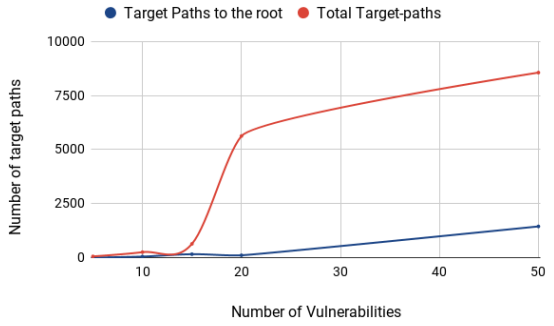

Fig. 5. Number of Target-paths vs. number of vulnerabilities (Number of nodes = 10)

end nodes to the target node. However, if we see, target paths to the target (root node) only then, it varies lineraly with the number of vulnerabilities on the target node.

VI. CONCLUSION

IoT-PEN is first-of-its-kind automatic E2E penetration framework for IoT, a large and complex network. It works well for resource constraint IoT devices, as most of the computation is done on cloud (servers) and possibly on the edge rather than IoT devices. The framework uses the open-source tool for scanning, vulnerability mapping to CVE-ID. The evaluation of IoT-PEN verifies the applicability of IoT-PEN to IoT network. We first pentest individual system and find the vulnerabilities present in those systems. These vulnerabilities are mapped with the NVD database. Prerequisite and Post-condition w.r.t. those vulnerabilities are constructed from the NVD database.

The future work includes replacing CVSS analysis to an in-system module for vulnerability analysis to check zero-day vulnerabilities. A lot of vulnerabilities are reported for individual software; not all are attacked. Therefore, not all vulnerabilities in each target-path need to be patched. So, we will extend the framework for an optimized patch management system.

REFERENCES

[1] S. Li, L. D. Xu, and S. Zhao, "5g internet of things: A survey," *Journal of Industrial Information Integration*, vol. 10, pp. 1 – 9, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2452414X18300037

[2] A. Zimba, Z. Wang, and H. Chen, "Multi-stage crypto ransomware attacks: A new emerging cyber threat to critical infrastructure and industrial control systems," *ICT Express*, vol. 4, no. 1, pp. 14 – 18, 2018, sI: CI & Smart Grid Cyber Security. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405959517303302

[3] K. Angrishi, "Turning internet of things(iot) into internet of vulnerabilities (iov) : Iot botnets," *CoRR*, vol. abs/1702.03681, 2017. [Online]. Available: http://arxiv.org/abs/1702.03681

[4] N. Falliere, L. O Murchu, and E. Chien, "W32.stuxnet dossier," 2010, last accessed 13-11-2018. [Online]. Available: https://www.wired.com/images_blogs/threatlevel/2010/11/w32_stuxnet_dossier.pdf

[5] R. A. M. Steve Christey, "Vulnerability type distributions in cve," 2007. [Online]. Available: https://cwe.mitre.org/documents/vuln-trends/index.html

[6] NIST, "Common vulnerabilites and exposures." [Online]. Available: https://cve.mitre.org

[7] M. Denis, C. Zena, and T. Hayajneh, "Penetration testing: Concepts, attack methods, and defense strategies," in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, April 2016, pp. 1–6.

[8] V. Visoottiviseth, P. Akarasiriwong, S. Chaiyasart, and S. Chotivatunyu, "Pentos: Penetration testing tool for internet of thing devices," in *TENCON 2017 - 2017 IEEE Region 10 Conference*, Nov 2017, pp. 2279–2284.

[9] Xueqiu, Q. Jia, S. Wang, C. Xia, and L. Lv, "Automatic generation algorithm of penetration graph in penetration testing," in *3PGCIC*, Nov 2014, pp. 531–537.

[10] M. U. Aksu, K. Bicakci, M. H. Dilek, A. M. Ozbayoglu, and E. i. Tatli, "Automated generation of attack graphs using nvd," in *CODASPY*. New York, NY, USA: ACM, 2018, pp. 135–142. [Online]. Available: http://doi.acm.org/10.1145/3176258.3176339

[11] A. T. Al Ghazo, M. Ibrahim, H. Ren, and R. Kumar, "A2g2v: Automated attack graph generator and visualizer," in *Proceedings of the 1st ACM MobiHoc Workshop on Mobile IoT Sensing, Security, and Privacy*, ser. Mobile IoT SSP'18. New York, NY, USA: ACM, 2018, pp. 3:1–3:6. [Online]. Available: http://doi.acm.org/10.1145/3215466.3215468

[12] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 336–345. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180446