A Knowledge-Based IoT Security Checker

Marco Anisetti[1], Rasool Asal[2], Claudio Agostino Ardagna[1], Lorenzo Comi[1], Ernesto Damiani[1,3], and Filippo Gaudenzi[1](✉)

[1] DI – Università degli Studi di Milano, Milan, Italy
{marco.anisetti,claudio.ardagna,lorenzo.comi,ernesto.damiani, filippo.gaudenzi}@unimi.it

[2] British Telecommunications, London, UK
rasool.asal@bt.com

[3] Centre on Cyber-Physical Systems, Khalifa University, Abu Dhabi, UAE
ernesto.damiani@kustar.ac.ae

Abstract. The widespread diffusion of ubiquitous and smart devices is radically changing the environment surrounding the users and brought to the definition of a new ecosystem called Internet of Things (IoT). Users are connected anywhere anytime, and can continuously monitor and interact with the external environment. While devices are becoming more and more powerful and efficient (e.g., using protocols like zigbee, LTE, 5G), their security is still in its infancy. Such devices, as well as the edge network providing connectivity, become the target of security attacks without their owners being aware of the risks they are exposed to. In this paper we present *IoT Security Checker*, a solution for IoT security assessment coping with the most relevant IoT security issues. We also provide some preliminary analysis showing how the IoT Security Checker can be used for verifying the security of an IoT system.

1 Introduction

Internet of Things (IoT) is changing the world where we live and the way in which we interact. Current environment composed of billions of interconnected devices points to scenarios where everything can be a data source or an actuator. According to Gartner, there will be more than 20 Billions devices by 2020 and every sector, from private life to public services, will be influenced and significantly improved by IoT technologies.[1] Baby-monitor, fitness bands, dog-tracker, smart-locker are already common goods with large adoption. Their exponential rate of adoption makes IoT devices and infrastructure the target of new security attacks [4], introducing many concerns about the risks an IoT systems need to face. The heterogeneity, variety, and complexity of IoT systems require the support of high security standards, which conflicts with the intrinsic insecurity of devices that are often under the control of non-expert users. Several studies and articles [4,6,10,13] reported on security threats and flaws affecting an enormous amount of devices, resulting in large-scale attacks and data breach [5].

[1] https://www.gartner.com/newsroom/id/3598917.

G. Mencagli et al. (Eds.): Euro-Par 2018 Workshops, LNCS 11339, pp. 299–311, 2019.
https://doi.org/10.1007/978-3-030-10549-5_24

IoT security does not only concern the application layer, where IoT devices play the role of sensors or actuators, but it also affects lower layers of the stack such as network, hardware, and the center of the architecture (e.g., cloud) [2,3].

In this paper, we present the IoT Security Checker (Sect. 5), a scanner supporting pentesters in carrying out a complete and structured analysis aimed to identify IoT device vulnerabilities. IoT Security Checker identifies IoT devices and collects information driving such analysis. It relies on public information sources such as Shodan [8], Censys (https://censys.io/), and National Vulnerability Database (https://nvd.nist.gov/).

The remaining of the paper is organized as follow. Section 2 describes the main security issues affecting IoT solutions. Section 3 describes an IoT classification used as a reference by the IoT-Security Checker. Section 5 describes the architecture and processes implemented by the IoT Security Checker, while an experimental scenario is reported in Sect. 6. Section 7 presents the related work on IoT security. Finally Sect. 8 draws our conclusions.

2 Security Attack Surfaces

IoT security introduces new requirements and challenges due to: *(i)* lack of control on the production environment, *(ii)* limited resources of the devices, *(iii)* limitations on the connectivity, reachability, power consumption, *(iv)* difficulties in imposing security best practices that consider the entire IoT environment. The goal of providing a secure IoT environment is a complex task that requires to consider both the plurality of devices and the heterogeneity of the IoT infrastructure and edge network. Device hardening requires a security-by-design approach involving the whole development-cycle, from hardware design to software/firmware implementation. This scenario is further complicated by the fact that security features need not to hinder the IoT functioning, especially preserving resource consumptions.

In this context, Open Web Application Security Project (OWASP) has identified the top ten IoT vulnerabilities (https://www.owasp.org/index.php/Top_IoT_Vulnerabilities), as well as several possible attack surfaces (https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Attack_Surface_Areas) that are summarized in the following.

– *Ecosystem Access Control* refers to access control mechanisms, enrollment and decommissioning procedures.
– *Device Memory* refers to the possibility of having clear-text credentials stored in memory and the management of cipher keys.
– *Device Physical Interfaces* refers to the firmware extraction and updates, and to removal storages and reset operations.
– *Device Web Interface* refers to all features and services offered by the device over the web.
– *Device Firmware* refers to the presence of credentials, sensitive information, keys stored inside the firmware.

- *Device Network Services* refers to all security issues related to connectivity and includes communication channels, UDP services and CLI (command line interface) to interact with the device.
- *Administrative Interface* refers to all possible attacks and threats related to the admin console, which may involve web attacks and restrict access and accounting techniques to improve security.
- *Local Data Storage* refers to the need to encrypt data at rest and to guarantee integrity.
- *Cloud Web Interface* refers to all services that are not offered by the devices, but rather connected to them and, in turn, with the user through an interface.
- *Third-party Backend APIs* refers to all issues related to the possibility of data breach using third parties. It points to the need of encrypted channels and anonymized data.
- *Update Mechanism* refers to techniques that should prevent all attacks during update operations, which may modify or replace device firmware and software.
- *Mobile Application* refers to all applications connected to the devices.
- *Ecosystem Communication* refers to all techniques that permit to monitor the status of an IoT device, including deprovisioning and update notifications.
- *Network Traffic* refers to all security issues that are related to the network and communication choices made during the design (e.g., radio or cabled communications).
- *Hardware (Sensors)* refers to all possible physical tampering and damages that may be applied to sensors and devices.
- *Privacy* refers to those devices leading to personal information, such as location or medical data, leak.
- *Authentication* refers to all authentication mechanisms offered by the IoT: administrative access, user access through the web, cloud applications, mobile applications, peer to peer IoT exchange information, to name but a few.
- *Vendor Backend APIs* refers to all possible attacks and vulnerabilities that may affect the APIs provided by the vendors.

Clearly, IoT attack surfaces are not limited to IoT devices. They also include processes involving devices, cloud or mobile applications enabling interaction with devices, as well as considered environments. Furthermore, the whole IoT stack from physical layer to service layer may be the target of an attack. IoT attack surfaces can be organized in four main categories that we identified in this paper as follow:

- *Third-Party Services.* This category involves all attack surfaces that depend on services and apps that may be used to collect or manage IoT devices (e.g. smarphone apps, data logger, cloud apps).
- *Physical Environment.* This category involves all attack surfaces that are correlated to environmental or physical damages a device may cause or suffer.
- *Device Logic.* This category includes all attack surfaces related to software, interfaces, and services embedded in or provided by IoT devices.
- *Communication Channel.* This category includes all attack surfaces related to the communication channel such as ZigBee and BLE.

3 Devices Classification

Although IoT is today a shared concept, there are different definitions. In this paper, we consider the ETSI definition that is built on the concept of Machine to Machine (M2M). It separates the communication (*M2M communication*) and the devices (*M2M device*), where *M2M devices* are defined as devices running M2M application(s) using M2M communication capabilities. This section analyzes some of the M2M device characteristics and tries to provide a classification over them that will be used in Sect. 5.

In 2017, NSfocus (https://blog.nsfocusglobal.com/categories/exposed-iot-assets-in-china-analysis/) carried out an analysis on public device endpoints in China based on information collected by most famous scan engines (Shodan, NTI, Zoom Eye). The output of this analysis was a list of 12 IoT categories, which we reassembled into six macro-categories used by the IoT Security Checker to classify the discovered IoT devices.

- **IP-Camera** refers to all devices recording or playing video content such as web cams, DVR and streaming devices such as baby monitors.
- **Router** refers to switches, modems, routers and any other network appliances.
- **Defense** refers to all devices that aim to protect a system (e.g., firewalls, IDSs, IPSs).
- **Printer** refers to all printing and fax devices that may be exposed to the Internet to provide a higher interoperability.
- **ICS** refers to all Industrial Control System (ICS) that plays a fundamental role in smart grids and industry 4.0.
- **Generic** refers to all devices that cannot be ranked in any of the above categories, but expose well-known protocols such as XMPP, CoAP, MQTT.

4 Device Mining

Given the classification in Sect. 3, a preliminary knowledge extraction is needed to support further analysis on the IoT devices. The scope of this knowledge extraction is to identify a set of properties that describe every IoT category in terms of *(i)* keywords, *(ii)* manufacturer, *(iii)* ports, and *(iv)* vulnerabilities, thus enhancing the vulnerability assessment carried out with our IoT Security Checker.

The knowledge extraction is based on text mining done on information retrieved by Shodan [8] and composed of 3 main steps as follow:

1. Create a keyword and manufacturer list for each category.
2. Create a port list for each category.
3. Create a vulnerability list for each category.

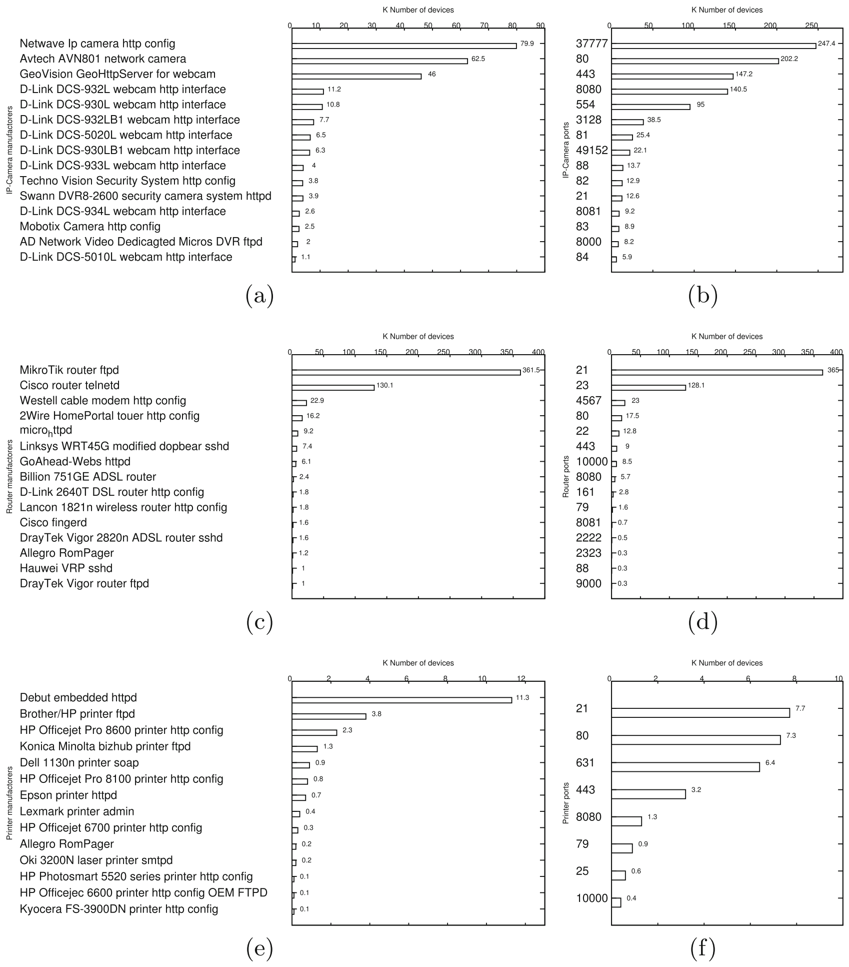

Fig. 1. Distribution of manufacturers and ports over three categories: IP-Cameras (a, b), Routers (c, d), Printers (e, f).

Based on Shodan public information, an analysis on the main services and ports used by IoT devices has been carried out. Figure 1(a) reports the top 15 IP-Cameras manufacturer found by Shodan using the filter "device:webcam", while Fig. 1(b) shows the distribution of ports used by IP-Cameras. Manufacturer and ports for routers has been identified using the filter "device:switch", device:broadband+router" and "device:load+balancer"; filters "device:print+server" and "device:printer" have been used for category printer. Results from categories router and printer are shown in Fig. 1(c), (d), (e), (f), while the complete results of ports analysis are summarized in Table 1.

Table 1. Matching between categories and ports from Shodan analysis

Category	Ports
IP-Camera	81, 82, 83, 84, 88, 443, 554, 37777, 49152, 143
Router	1900, 21, 80, 8080, 1080, 9000, 8888, 8000, 49152, 81, 8081, 8443, 9090, 8088, 88, 82, 11, 9999, 22, 23, 7547
Printer	80, 631, 21, 443, 23, 8080, 137, 445, 25, 1000
Firewall	8080, 80, 443, 81, 4433, 8888, 4443, 8443
ICS	47808, 20000, 44818, 1911, 4911, 2404, 789, 502, 102
Generic	5222, 5683, 1883, 8883

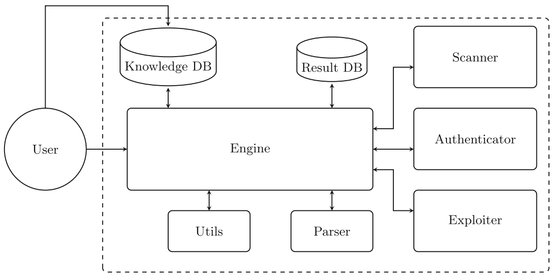

Fig. 2. IoT Security Checker architecture.

5 IoT Security Checker

The IoT Security Checker helps pentesters in identifying vulnerable devices in a given network, using discovery mechanism and known exploits. In the following, we describe the IoT Security Checker architecture, execution flow, and target exploit.

5.1 Architecture

Figure 2 shows the internal modules of IoT Security Checker.

Knowledge DB contains all information and data acquired during the mining phase in a NO-SQL DB. It can be updated in real time as new information is collected.

Scanner manages and starts the host discovery process. The scanning operations are run using masscan (https://github.com/robertdavidgraham/masscan).

Authenticator executes a dictionary attack on the following services: FTP, Telnet, SSH, HTTP basic. The dictionary is built based on the knowledge extraction in Sect. 4.

Exploiter executes a set of exploits of well-know IoT vulnerabilities. As for dictionary, the exploit list derives from the knowledge extraction in Sect. 4.

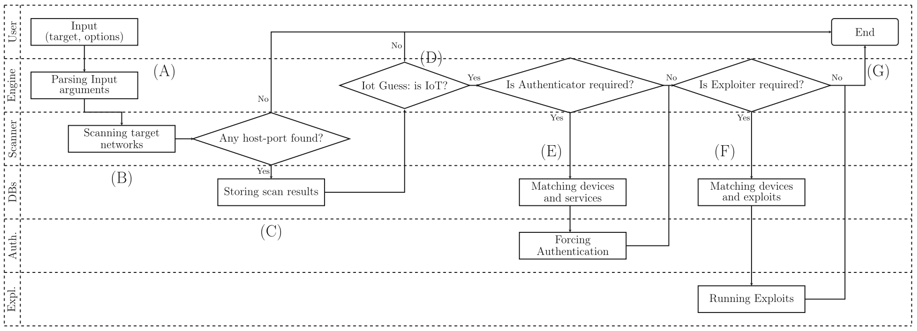

Fig. 3. IoT Security Checker execution flow

Engine manages all operations and exchanges of information through all modules. The user can set up the scan and then the Engine is in charge of starting the scanning, redirect data to the parser and DBs, setting up the execution of the Exploiter and Authenticator based on the results.

Utils provides a set of functionalities for input/output validation.

Parser includes all parsers that translate and filter outputs from one module and give them as input to another module. It also includes a human-readable translation parser for the final output.

Result DB stores all the information about the found services, hosts, and devices. It is used as the target list by Authenticator and Exploiter, or to store the final evaluation of the target devices. The host table storing the scanner results has the following structure:

– *Timestamp.* It is the timestamp when the scanner returned the result.
– *IP.* It is the IP of the found services; we note that there might be several rows with the same IP since the primary key is composed of the pair (IP, Port) representing the service.
– *Port.* It is the port where the service is listening.
– *Service.* It is the type of found service (e.g., SSH, HTTP)
– *Banner.* It is the discovered banner for the related service.
– *Info.* It contains extra information found during the scanning of the given service.
– *Error.* It contains any possible errors during the scanning operations.

5.2 System Flow

Figure 3 annotates the architecture in Fig. 2 with the flow of a single execution of the IoT Security Checker. Before starting the flow execution, the IoT Security Checker must be initialized by loading the knowledge. The IoT Security Checker can then be configured by the user by simply specifying the target.

Once the input parameters[2] are set (A), based on the parameters and the knowledge, the Engine launches the appropriate scanning (B). If no services/hosts are found, the execution ends; otherwise, they are stored in the result DB (C). The Engine analyzes all results based on the knowledge to identify whether they are IoT devices or not (D); this process is called the *IoT guess*. In case IoT devices are found, the Authenticator runs the appropriate vocabulary attacks based on the available services (E). The Authenticator stores the attack results in the Result DB and then the Exploiter runs the available exploits only on those appropriate services (F-G). The results are stored in the Result DB and finally shown to the user (H-I).

5.3 Target Exploit

The IoT Security Checker implements a set of exploits as follows.[3]

- *Cisco-PVC-2300*: the web camera Cisco PVC-2300 is affected by several vulnerabilities that may allow an unauthenticated user to login and access to multiple functionalities. The developed exploit tries to login and download the device configuration to read username and password.
- *Dlink*: a set of Dlink webcams are affected by different vulnerabilities that mainly permits OS command injection. The developed exploit tests each of these vulnerabilities.
- *h264-dvr-RCE*: a set of devices identified by the caption "Cross Web Server", which have been used by several companies, may suffer Remote Command Injection. This vulnerability allows an attacker to execute any commands on the vulnerable device. The exploit verifies the vulnerabilities attempting to create a file on the target device.
- *Humax-HG100R*: the Humax Wifi Router is vulnerable to Authentication Bypass attack by sending specific crafted request to the management console. If the console is publicly exposed, an attacker can exploit it and may get access to confidential information.
- *Rom-0*: a set of network appliances from companies such as ZTE, TP-Link, ZynOS, and Huawei are vulnerable to Authentication Bypass attacks. An attacker can access confidential data sending a crafted HTTP request to the /rom=o resource.
- *TV-IP410wn*: Trendnet TV-IP410WN webcams are vulnerable to Remote Command Execution attacks. The developed exploit verifies these vulnerabilities by executing the *ls* command on the target device.

[2] All input parameters are described in details at https://github.com/c0mix/IoT-SecurityChecker.

[3] We took inspiration from the following articles: https://media.blackhat.com/us-13/US-13-Heffner-Exploiting-Network-Surveillance-Cameras-Like-A-Hollywood-Hacker-WP.pdf, http://www.kerneronsec.com/2016/02/remote-code-execution-in-cctv-dvrs-of.html, https://www.exploit-db.com/exploits/42732/, https://rootatnasro.wordpress.com/2014/01/11/how-i-saved-your-a-from-the-zynos-rom-0-attack-full-disclosure/, https://medium.com/@lorenzo.comi93/break-into-2k-ip-camera-cb65bbac9e8c.

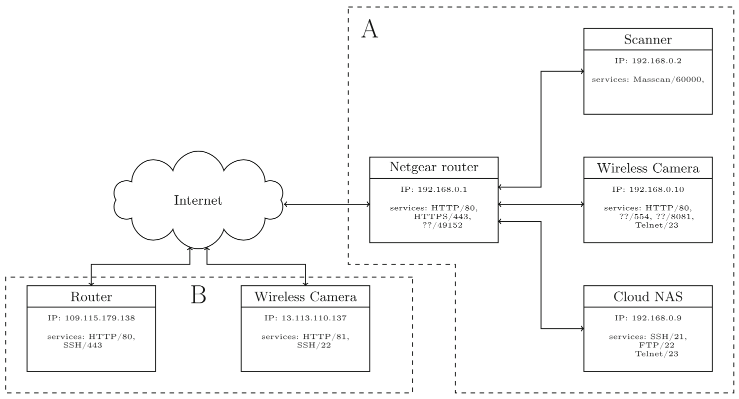

Fig. 4. IoT Security Checker experimental scenario

A zero-day vulnerability affecting some webcams and baby monitors (*IPcamera* vulnerability in the following) has been discovered during the development of the IoT Security Checker and now classified as CVE-2017-17101. It is based on Credential Injection through the camera web app. An attacker can obtain full admin access by using a crafted HTTP request, for instance, accessing video streams and changing credentials.

6 Experimental Scenario

We run the IoT Security Checker in a simulated environment that have been built specifically to test the tool functionalities. Figure 4 shows the networks with all nodes. The experimental environment is composed of a private network (A) containing the *scanner* node, an IPcamera wireless cam and a Cloud NAS. All nodes access internet through a Netgear router. A Wireless cam and a router (B) are added to the scenario and reachable from the private network through Internet.

The IoT Security Checker was executed from the scanner node with the following command,

```
sudo python3 IoT-SecurityChecker.py target.txt -m 300
-w 15 -E ALL -B ALL -T 2 -o result.csv
```

where:
target.txt contains three different targets: *(i)* the private network (192.168.0.0/24) and the two public IPs (109.115.179.138, 13.113.110.137).
-m 300 -w 15 instructs masscan how to run the scan. These parameters require to use no more than 300 packages per second and wait 15 s once the scan is done to get the results.

Table 2. Pairs (host, port) found after network scan (1); hosts, ports and information after process IoT guess (2)

host	port
192.168.0.1	80
192.168.0.1	443
192.168.0.10	8081
13.113.110.137	81
192.168.0.10	23
109.115.179.138	443
192.168.0.9	21
192.168.0.9	23
192.168.0.10	554
192.168.0.9	22
13.113.110.137	22
109.115.179.138	80
192.168.0.10	80
192.168.0.1	49152

(a)

host	port	keywork	classification
109.115.179.138	443	SSH	
13.113.110.137	22		Routers
13.113.110.137	81		IP-camera
13.113.110.137	22	SSH	
192.168.0.1	49152		IP-camera
192.168.0.1	49152		Routers
192.168.0.10	23		Routers
192.168.0.10	554		IP-camera
192.168.0.10	8081		Routers
192.168.0.9	21		Routers
192.168.0.9	22		Routers
192.168.0.9	23		Routers
192.168.0.9	22	SSH	

(b)

Table 3. Final results from IoT Security Checker.

192.168.0.10	23	Telnet	TelnetAuthenticator	Telnet Access found username: adm password:
192.168.0.9	23	Telnet	TelnetAuthenticator	Telnet Access found username: adm password:
192.168.0.9	22	SSH	SSHAuthenticator	SSH Access found username: test password: admin
192.168.0.9	21	FTP	FTPAuthenticator	FTP Access found username: anonymous password:
192.168.0.9	21	FTP	FTPAuthenticator	FTP Access found username: user password: test
13.113.110.137	81	HTTP	HttpAuthenticator	Http Access found username: test password: test
109.115.179.138	80	HTTP	Rom-0	http://109.115.179.138:80/rom-0
192.168.0.10	80	HTTP	IPcamera	http://192.168.0.10:80 new credentials are admin:hacked

-B ALL -E ALL -T 2 instructs Authenticator (-B) and Exploiter (-E) to run all possible authentications and exploits, but using a maximum of two threads. **-o results.csv** requires to store the final results in file result.csv.

The first phase of the analysis scans the network based on what specified in *target.txt*. The scan returns a set of (host, port) as described in Table 2(a). Each pair is then analyzed to identify possible IoT devices (process *IoT guess*). Table 2(b) reports the results with service and IoT device classification for each part after the IoT guess.

Following the IoT Security Checker flow described in Fig. 3, using the result from IoT guess, the *authenticator* module attempts to authenticate to all available services. Authentication attacks were successful over telnet protocol on hosts 192.168.0.9 and 192.168.0.10: they have been accessed with username "adm" and empty password. Attacks on SSH were tried on hosts 192.168.0.9, 13.113.110.137, 109.115.179.138; access was granted only on 192.168.0.9. Attacks on HTTP basic authentication were tried on hosts 192.168.0.1, 109.115.179.138, 13.113.110.137, 192.168.0.10; a single attack was successful on 13.113.110.137 (access was granted on port 81 with credentials "test:test").

After vocabulary attacks, the IoT Security Checker attempted to attack the devices using the exploits described in Sect. 5.3 based on HTTP.

No host was vulnerable to *h264-dvr-RCE*, *Cisco-PVC-2300*, *TV-IP410wn*, *Humax-HG100R* and Dlink. Host 109.115.179.138 was vulnerable to *Rom-0*, while host 192.168.0.10 to *IPcamera*. Table 3 reports the final results returned by the IoT Security Checker; each row of the table shows a security issue.[4]

7 Related Work

IoT security and vulnerability scanning are hot research topics. Kumar et al. [6] and Zhao et al. [13] presented an overview of the main IoT security issues focusing of the importance of a holistic view over the three-layer system structure. A real use case is described by Seralathan et al. [10], where the authors analyze the security of a general webcam taking into consideration the camera itself, as well as its mobile and cloud applications and communication channels. Shodan [8], a public information source on IoT devices, is widely used in IoT security research [1,7,9,12]. Markowsky et al. [7] analyzed the router status in the Indian Autonomous System Number (ASN) space, identifying misconfiguration or Rom-0 vulnerability. Williams et al. [12] defined a pattern to analyze webcams, smart-tv, and printers. First these devices are identified using Shodan and then a vulnerability scan is used to assess possible vulnerabilities. A similar approach is used by Samtani et al. [9], where Shodan is adopted to identify SCADA system and then Nessus is run to find potential vulnerabilities. The authors however introduced a text-mining approach that filters the results from Shodan to enhance the SCADA recognition process. Al-Alami et al. [1] presented an overall view of IoT devices in Jordan with a specific focus on security using Shodan. Solutions based on Shodan support a fast and wide analysis, but limited to public-accessible devices. Visoottiviseth et al. [11] presented an assessment tool based on Kali Linux called PENTOS. Pentos permits to scan a private network and subsequently assess the found services and hosts. PENTOS is completely manual; indeed the pentesters set the scanning at the beginning and then choose the assessment to run. The IoT Security Checker, being based on a knowledge, automatically identifies IoT devices and sets the appropriate assessment based on their characteristics.

[4] The logs of this experiment are available at https://github.com/c0mix/IoT-SecurityChecker.

8 Conclusions

This paper presented IoT Security Checker, a vulnerability scanner for IoT devices. The tool, using a knowledge built on public information, can help pentesters in providing an IoT security assessment. The modularity of the tool permits to easily extend the knowledge, and the available vocabulary and exploits. Future work will consider the development of an intelligent knowledge that can be automatically built and updated, driving a more effective assessment. Furthermore, IoT Security Checker will be extended towards assurance verification and monitoring of IoT devices and infrastructures.

Acknowledgements. This project was partly supported by the program "piano sostegno alla ricerca 2015-17" funded by Università degli Studi di Milano.

References

1. Al-Alami, H., Hadi, A., Al-Bahadili, H.: Vulnerability scanning of IoT devices in Jordan using Shodan. In: Proceedings of IT-DREPS 2017, pp. 1–6 (2017). https://doi.org/10.1109/IT-DREPS.2017.8277814
2. Anisetti, M., Ardagna, C.A., Damiani, E., Gaudenzi, F., Veca, R.: Toward security and performance certification of open stack. In: Proceedings of IEEE CLOUD 2015, June 2015. https://doi.org/10.1109/CLOUD.2015.81
3. Anisetti, M., Ardagna, C., Damiani, E., Gaudenzi, F.: A semi-automatic and trustworthy scheme for continuous cloud service certification. IEEE TSC (2017)
4. Ardagna, C.A., Damiani, E., Schütte, J., Stephanow, P.: A case for IoT security assurance. In: Di Martino, B., Li, K.-C., Yang, L.T., Esposito, A. (eds.) Internet of Everything. IT, pp. 175–192. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-5861-5_8
5. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: DDoS in the IoT: Mirai and other botnets. Computer **50**(7), 80–84 (2017). https://doi.org/10.1109/MC.2017.201
6. Kumar, N., Madhuri, J., ChanneGowda, M.: Review on security and privacy concerns in Internet of Things. In: Proceedings of ICIOT 2017, pp. 1–5 (2017). https://doi.org/10.1109/ICIOTA.2017.8073640
7. Markowsky, L., Markowsky, G.: Scanning for vulnerable devices in the Internet of Things. In: Proceedings of IEEE IDAAC 2015, vol. 1, pp. 463–467, September 2015. https://doi.org/10.1109/IDAACS.2015.7340779
8. Matherly, J.: The Complete Guide to Shodan: Collect. Analyze. Visualize. Kindle Publisher (2016)
9. Samtani, S., Yu, S., Zhu, H., Patton, M., Matherly, J., Chen, H.: Identifying supervisory control and data acquisition (SCADA) devices and their vulnerabilities on the Internet of Things (IoT): a text mining approach. IEEE Intell. Syst., 1 (2018). https://doi.org/10.1109/MIS.2018.111145022
10. Seralathan, Y., et al.: IoT security vulnerability: a case study of a web camera. In: Proceedings of ICACT 2018, pp. 172–177, February 2018. https://doi.org/10.23919/ICACT.2018.8323686

11. Visoottiviseth, V., Akarasiriwong, P., Chaiyasart, S., Chotivatunyu, S.: PENTOS: penetration testing tool for Internet of Thing devices. In: Proceedings of IEEE TENCON 2017, pp. 2279–2284 (2017). https://doi.org/10.1109/TENCON.2017.8228241
12. Williams, R., McMahon, E., Samtani, S., Patton, M., Chen, H.: Identifying vulnerabilities of consumer Internet of Things (IoT) devices: a scalable approach. In: Proceedings of IEEE ISI 2017, pp. 179–181 (2017). https://doi.org/10.1109/ISI.2017.8004904
13. Zhao, K., Ge, L.: A survey on the Internet of Things security. In: Proceedings of CIS 2013, pp. 663–667 (2013). https://doi.org/10.1109/CIS.2013.145