A. Acien, A. Nieto, G. Fernandez, and J. Lopez, "A comprehensive methodology for deploying IoT honeypots", *15th International Conference on Trust, Privacy and Security in Digital Business (TrustBus 2018)* vol. LNCS 11033, pp. 229243, 2018.
http://doi.org/https://doi.org/10.1007/978-3-319-98385-1_16
NICS Lab. Publications: https://www.nics.uma.es/publications

A comprehensive methodology for deploying IoT honeypots

Antonio Acien, Ana Nieto, Gerardo Fernandez, and Javier Lopez

Network, Information and Computer Security (NICS) Lab
Lenguajes y Ciencias de la Computación
Universidad de Málaga, Spain
{acien,nieto,gerardo,jlm}@lcc.uma.es

Abstract. Recent news have raised concern regarding the security on the IoT field. Vulnerabilities in devices are arising and honeypots are an excellent way to cope with this problem. In this work, current solutions for honeypots in the IoT context, and other solutions adaptable to it are analyzed in order to set the basis for a methodology that allows deployment of IoT honeypot.

Keywords: IoT, honeypot, security, methodology

1 Introduction

The *Internet of Things* (IoT) has initiated a technological revolution that affects both the public and private sector. One major concern is that users are delegating in their personal objects an important part of their daily routine without being really protected against malicious attacks.

In fact, the number of connected devices is expected to exceed twenty billion in 2020 (being prudent) [1]. This is a problem due the obvious density of devices sharing the same spectrum, but it also means that there will be even more platforms and services deployed independently and within a too short time interval. Security solutions need time to be deployed prior this deployment of networks and devices and it will be materially impossible to do that. Furthermore, even considering that security mechanisms and services are deployed in some way, nowadays it is impossible to accurately predict the effect that a targeted attack can have on the current infrastructure network.

In particular, IoT devices are a succulent call for attackers intended to cause the major damage possible, precisely given i) the user's dependence on their devices, and ii) the power of decision that we give to these devices (e.g. relying on an automatic vehicle to drive for us).

It would be unrealistic to think that attacks will not occur, so we must promote measures to detect threats as early as possible, understand them and analyze them in a safe environment, prepared to receive them. In other words, it is necessary to attract attacks against trap nodes or networks in order to obtain malicious code that can be analyzed safely. These attractive hooks for cyberattackers are known as honeypots and honeynets. The simulation and emulation

of these services would also allow the deployment of trap services dynamically as the analysis platform requires. Moreover, these honeypots are useful when it comes to monitoring and logging attacks that usually erase their traces, and are affordable both economically and on processing power, since they allow to use simulated systems instead of real devices.

Although the security on IoT devices was considered back as far as 2011 due to attacks on routers and other embedded devices [2], it was the amount of botnets (networks of bots) that started appearing what raised serious concern. After some botnet attacks (Tsunami, Gafgyt, BrickerBot), the one that caused major disruption was Mirai in 2016, which taking advantage of vulnerabilities in IoT devices, launched a distributed denial of service attack that took down websites such as Amazon, Twitter or GitHub [3]. As the worry about these attacks increases, the amount of malware targeted to IoT platforms does to, as well as the budget destined to IoT security [4].

This article is focused on one of the major concerns in the deployment of IoT honeypots. Specifically, we propose a methodology to deploy *relevant* honeypots in IoT environments, considering aspects as the *ranking popularity* of the chosen devices and the requirement for avoiding the detection of our *trap nodes* as honeypots by some of the most popular IoT scanners. Although the methodology proposed could be adapted to generic honeypots we focus on IoT honeypots due to the concern they represent nowadays.

2 Related work

Although there have been revent and exhaustive surveys about honeypots which include classifications, maintenance and focus [5], to the best of our knowledge, there is no work regarding the deployment of IoT honeypots with a whole methodology behind.

The existing surveys do not analyze the specific requirements that are needed in order to deploy honeypots in IoT environments. This is the main focus of this paper; to properly and clearly identify a set of steps and procedures to successfully deploy honeypots in IoT environments avoiding to be detected by web search engine tools and, at the same time, be attractive to the attackers.

There is some work done in the field of IoT honeypots, with some interesting deployments and architecture, but due to the novelty of the field, the amount is very limited. Some IoT honeypots deployment architectures, such as IoTPOT [6], SIPHON [7] or IoTCandyJar [8] are quite interesting, since they have advantages such as redirecting low-interaction honeypots to high-interaction ones (making the whole system look more real to an attacker without a high cost), connecting the honeypots to cloud servers to make them look distributed all around the world, or using the responses of real IoT devices connected to the internet. Some other works are improved implementations of these ones, such as [9], which is an improved open-source version of IoTPOT.

There are also honeypots that are not focused on architecture, but worth mentioning, such as Wificam or Honeypot Camera [5], which impersonate web-

cams, MTPot [10] and ThingPot [11], which act as a IoT devices with certain vulnerabilities (Mirai and TR-064 respectively), or others that focus on personal area networks (PAN) [12]. Some honeypots specialize in industrial environments, such as Conpot [13]. Both categories, honeypots architectures and standalone ones, are detailed in Table 1.

Table 1. IoT honeypots

Honeypot	Characteristics				Viability of emulation		
	Scope	Type	Protocol	Architecture	HW	Downl.	Maintenance
IoTPOT	Industrial, home, personal	High	Telnet	MIPS,ARM, PPC	No	No	Article: May 2015
SIPHON	Profesional, personal	High	SSH, HTTP	Cloud-based	Yes	No	Article: Jan. 2017
Multi-purpose IoT honeypot	Industrial, home, personal	High	HTTP, SSH, TR-064, Telnet	MIPS,ARM, PPC	No	Yes	Article: May 2017
Conpot	Industrial	Medium	Modbus (TCP), SNMP	ICS, SCADA, BACNet, HVAC	No	Yes	Article: Sep. 2016 [13]
IoTCandyJar	Industrial, home, personal	*Intelligent*	HTTP, SSH, Telnet, TR-064, XMPP, MQTT, UPnP, CoAP, MS-RDP...	MIPS,ARM, PPC	No	No	Article: July 2017
ThingPot	Household	Medium	HTTP, XMPP	Philips Hue lights	No	No	Last commit: Aug. 2017
HoneyThing	Routers	Low	TR-064	-	No	Yes	Last commit: Mar. 2016
ZigBee Honeypot	Personal	Medium	ZigBee	-	No	No	Last commit: June 2017
Honeypot-camera	Profesional, personal	Low	HTTP	-	No	Yes	Last commit: June 2015
MTPot	Mirai	Low	Telnet	-	No	Yes	Last commit: Nov. 2016
Wificam	Profesional, personal	Low	HTTP	-	No	Yes	Last commit: Apr. 2017

3 Methodology

Figure 1 shows the steps in the H-IoT methodology. We separate this methodology in five main blocks: (1) IoT security search, (2) build honeypot, (3) training,

(4) public deployment and (5) visualization and evaluation. Note that this last block can be executed in parallel to the training and the public deployment.

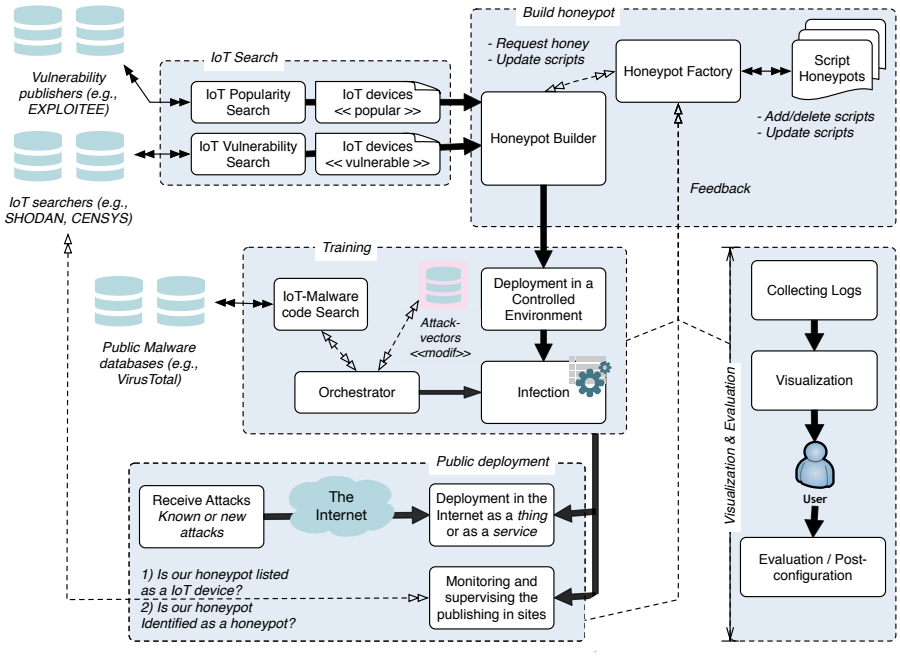

Fig. 1. Methodology

3.1 IoT Search

The two first challenges to be faced are (i) making the honeypot attractive for attackers and (ii) making the set of deployed honeypots representative of the general IoT context. This last one is crucial, due to the wide variety of devices which can be attacked. The usefulness of the results will depend on how rich the selected sample is. The related works are too specific or can become obsolete too soon if they don't satisfy (i) and (ii).

The first phase of the analysis aims to face both challenges. In this phase, IoT devices which can be used as honeypots are analyzed first. This, the searches are performed keeping in mind two basic attributes: popularity and vulnerability. These are not mutually exclusive, since popular devices may not be vulnerable to attacks, and vulnerable devices could be not widely used.

Next, the particularities of each search criterion are described. Some devices, such as those matching both criteria, or those in charge of a highly critical infrastructure, can be considered as high risk. Therefore, there must be a balance

between the relevance of these attributes, which will depend on the context of the devices.

Popularity-based search In order to carry out a search by popultarity, search engines specialized in IoT were consulted, searching for tags using manufacturers, models and types of devices. The search engines where the results were obtained from are the following:

- **SHODAN**: Search engine for IoT devices directly connected to the internet. Features filtering, banners, location and several other details. Also features HoneyScore, a parameter indicating the probability of a host being a honeypot.
- **Censys**: Searches the IPv4 namespace with ZMap. It can filter by protocol or words in the banner. Uses semantic search.
- **Reposify**: Thought with the goal of improving security, detecting bad settings (default credentials, open ports, obsolete firmware...).
- **Thingful**: Geographical search engine which categorizes the different IoT devices that it finds.
- **Wigle**: Searches networks by their SSID, which allows the user to find some devices that set up networks with a default name.

Moreover, general and specialized sellers were also used. To this respect, Table 2 shows a preliminar list of results of popular devices following both aforementioned criteria.

Table 2: Popular IoT devices (data retrieved on 16/02/2018)

Device type	**Model**	**Selling ranking**	**Search engine ranking**
Routers	TP-Link TL-WR841N	1 (Amazon Spain, category: routers)	3 (Shodan, tags:router) – 47K results
	TP-Link TL-WR740N		2 (Shodan, tags: router) – 78K results
	TP-Link TL-WR741ND		5 (Shodan, tags: router) – 11K results
	Linksys E2500		2 (Shodan, tags: linksys) – 3K results
	Netgear WNR1000v3		3 (Shodan, tags: netgear) – 3K results
	Linksys E1500	12 (Amazon USA, category: routers)	
	T-Mobile/ASUS AC1900	1 (Amazon USA, category: routers)	
	Netgear R6700	3 (Amazon USA, category: routers)	
	TP-Link N540	5 (Amazon USA, category: routers)	
	Linksys WRT54GL	2 (Amazon USA, category: routers)	

	Linksys E2500	4 (Amazon USA, category: routers)	2 (Shodan, tags: linksys) –1.8K results
	Linksys E4200	8 (Amazon USA, category: routers)	4 (Shodan, tags: linksys) – 500 results
IP cameras	DLink DCS-932L	17 (Amazon Spain, category: security cameras)	
	DLink DCS-5300		Censys, tags: "dcs-5300" – 73 results
	Sony SNC-RZ25		Censys, tags: "snc-rz25" – 798 results
	Axis M1054		Censys, tags: "axis m1054" – 401 results
	Axis 2100		Censys, tags: "axis 2100" – 350 results
DVRs and NVRs	Sony NSR-500	5 (Network Webcams)	Censys, tags: "nsr-500" – 3 resultados
	Axis Companion Recorder	1 (Network Webcams)	Censys, tags: "axis companion" – 10 results
	Hikvision DS 7604/7608/7616	3 (Network Webcams)	
	UniFi NVR		2 (Shodan, tags: nvr – 357 results)
	Zmodo NVR (ZMD-DT-SCN8)		5 (Shodan, tags: dvr – 450 results)
Antennas	Ubiquiti AirGrid M AG-HP-5G27		4 (Shodan, tags: router – 13K results)
Smart TVs	TiVo Series2 Firmware		2 (Shodan, tags: dvr – 3'6K results)
Industrial devices	Schneider Electric BMX P34 2020		1 (Shodan, tags: Schneider electric – 608 results)
	Schneider Electric BMX NOE 0100		2 (Shodan, tags: Schneider electric – 289 results)
	Schneider Electric SAS TSXETY4103		3 (Shodan, tags: Schneider electric – 169 results)
	Schneider Electric TM221CE40T		4 (Shodan, tags: Schneider electric – 117 results)
	Schneider Electric TM221CE40R		5 (Shodan, tags: Schneider electric – 60 results)
	Omron CJ2M		Censys, tags: "omron cj2m" – 4256 results, Shodan, tags: "omron cj2m" – 521 results
Drones and UAVs	DJI Phantom 3	9 (Amazon USA, category: hobby RC)	
	UDI U818A	8 (Amazon USA, category: hobby RC)	
	Holy Stone S160	1 (Amazon USA, category: hobby RC)	

Vulnerability-based search The second criterion to select devices as honeypots is based on known vulnerabilities. Furthermore, it is very important to have an updated database with the most relevant vulnerabilities classified by the type of device and context, and the estimated impact of the vulnerability.

Websites specialized in device vulnerabilities were used for this research. The most used are listed here, altough some papers and journals were also used. These are cited when needed.

- **Exploitee.rs**: Website listing vulnerable devices and their exploits.
- **CVE Details**: Database containing all the CVE (Common Vulnerabilites and Exposures) entries, detailing which brands, models, versions of firmware are affected, and links to the exploits if available.
- **Exploit database (exploit.db)**: Website which contains usable exploits and proofs of concepts about some vulnerabilites.

Table 3: Vulnerable IoT devices

Device type	**Model**	**Known vulnerability**
Routers	TP-Link TL-WR841N	Directory traversal Cross-site scripting
	TP-Link TL-WR740N	Denial of service through httpd crash
	TP-Link TL-WR741ND	Malicious code injection in SSID field
	Linksys E2500	Code injection in URLs
	Netgear WNR1000v3	Password recovery credential disclosure
	Linksys E1500	Code injection in URLs
	Zyxel AMG1302 and P-660HN	TR-064 vulnerability [14]
	Sagecom Livebox	Denial of service filling the IPv6 routing table
	T-Mobile/ASUS AC1900	Remote code injection
	Netgear R6700	Remote code injection
	Linksys WRT54GL	Code injection Denial of service Buffer overflow Authentication bypassing
	Linksys E4200	Remote password stealing
IP cameras	DLink DCS-932L	Cross-site request forgery Remote password stealing
	DLink DCS-930L	Cross-site request forgery
	Sony SNC-RZ25	Heap-based buffer overflow
Industrial devices	Schneider Electric BMX P34 2020	Stack-based buffer overflow
	Schneider Electric Modicon M340	Buffer overflow through web server login
	Schneider Electric Homelynk (LSS100100)	Cross-site scripting

	Schneider Electric TSEXTG3000	Stack-based buffer overflow
	Omron CJ2M	Passwords transmitted in clear [15]
Drones and UAVs	DJI Phantom 3	GPS spoofing GPS jamming [16]
	Parrot AR 2.0	GPS spoofing [17]
	UDI U818A	Remote unauthenticated controlling [18]
Home appliances	iKettle	Password stored in clear [19]
	Samsung RF28HMELBSR Fridge	Man-in-the-middle attack through SSL vulnerability [20]
	FitBit Aria Scale	Password stored in clear [21]
	Miele PG 8528 Dishwasher	Directory traversal vulnerability
	Philips Hue Lights	Vulnerability in firmware verification [22]
Miscellanea	CloudPets	Unencrypted cloud database [23]

3.2 Honyepot builder

This step in the methodology focuses in building honeypots considering both inputs from the previous phase, which are i) IoT devices with known vulnerabilities and ii) popular IoT devices. Sometimes both aspects may coincide. During this step, the viability to design and implement each honeypot is carefully analyzed in order to establish implementation priorities.

Although the implementation of the honeypots can depend of many factors (e.g., type of honeypot to be deployed, their dependency on hardware, etc.), in order to improve the efficiency of the system, it is very important to maintain a repository of solutions previously implemented. This would allow the deployment of honeypots which are similar to the already used ones in a fast and efficient manner, reusing some aspects such as parameters and settings. This is a likely scenario, since the output of the later phases will be used as feedback for this one, which will allow fine-tuning the deployed honeypots.

To classify and catalog these solutions based on the context, during this phase it is recommended to use a Honeypot Factory. This additional component will be used to control the access and modifications of a rich set of scripts, which describe the configuration of honeypots for the different application domains.

In addition, the deployment of honeypots will be much more efficient if these can be generated on-demand. This may happen, for example, if during the attack the honeypot gets corrupted, stuck, disabled or *broken*.

In the proof of concept presented in this paper, the infection of the virtual machine is manual, so this phase does not have as much weight as it would in a further developed deployment. However, future works are directed to develop specific IoT honeypots following this methodology.

3.3 Testing

During the *training* phase, the deployment of the honeypots is carried on in a controlled environment in order to test it. In order to do that, the honeypot is infected using, for example, known malware code downloaded from the Internet. In some cases the infection of a device is not trivial at all. One of the purposes of this step in the methodology is also to evaluate the viability of a honeypot for being infected. Note that, in some cases, it will be desirable that the honeypot will be the most vulnerable possible (e.g., to record lazy, opportunistic and automated attacks) but in other cases the honeypot should represent a challenge for a persistent attacker that might suspect of the victim if it has not a minimum security level.

3.4 Public deployment

The most useful feedback to the system implementing this methodology will be obtained from the phase of *public deployment*. During this step the selected honeypot is finally deployed with direct access to the Internet.

Paradoxically, although it is known that connecting IoT devices directly to the Internet exposes them to several threats, this is precisely what motivates the work. Due to the high density of devices, our honeypots can go unnoticed to the attackers, or even works, be identified as honeypots (alerting other attackers about it). The measures taken in the previous phases regarding the viability and interest of the honeypots try to prevent this.

Moreover, some factors, such as *marketing* will be needed, in order to make our devices listed by the relevant IoT search engines, and the places where the attacker get their targets from. Second, websites where devices are identified as honeypots must be checked to know if ours have been detected. Thus, a continuous monitoring is mandatory.

Last but not least, the deployment of the honeypot must be realistic. The technical complexity might depend for example on the dependencies of the solution with other theoretically *real* components in a productive environment. For example, PLCs in a SCADA system might require to interchange command controls with a controller. To reproduce all this behavior is very complex, furthermore considering that in some cases the protocols are proprietary.

3.5 Evaluation and validation

The phase of *visualization and evaluation* is parallel to both the *training* and *public deployment* phases. However, the results achieved from both phases must be clearly separated and classified. This will contribute to know if the expected results prior the public deployment correspond with the real results obtained after the deployment.

In particular, this phase is directed to make the solution usable to an administrator or investigator, depending on the user profile of the resultant system where the methodology is being implemented.

We must emphasize that logs can be collected in different formats and thus may represent a problem during the visualization step. Some tools for the interpretation of logs expect to receive the logs in a specific format. Therefore how to translate these logs to extract useful information must be considered during the implementation.

This methodology also considers the feedback provided by the user to improve/configure the platform of honeypots. This is a feature that is needed in order to allow the improvement of the solution and enlarge its maintenance as far as possible.

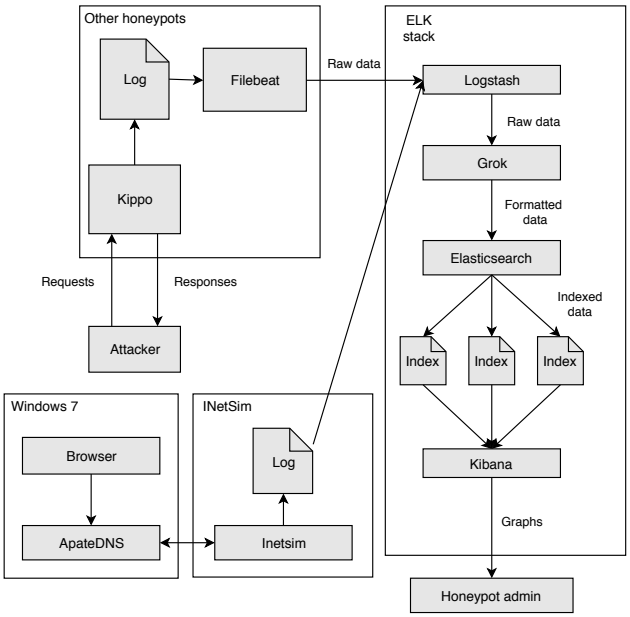

Fig. 2. Relationships between infected machines and ELK

4 Preliminary results

In October of 2016, a malware specifically targeted to IoT devices called Mirai, caused a denial of service attack on the DNS provider Dyn. The magnitude of this attack was unprecedented, taking down services such as Spotify, Twitter, Netflix and Amazon. It was performed through thousands of vulnerable IoT devices, which were made part of a botnet, sending requests to the target. Most of these devices were simply left with the default credentials by the manufacturer, which allowed an automated attack to log and modify their behavior.

Recently, the malware has jumped to Windows platform, behaving in a slightly different way. Previously, only the IoT devices directly exposed to the

Internet were vulnerable. Now, if a Windows computer is infected with its Mirai version, it will scan the local network, searching for vulnerable devices to make them part of the botnet. This makes devices in private networks or located behind a firewall vulnerable [24].

Due to the novelty, repercussion, strength, and challenging aspects of this particular attack, it has been used in the proof of concept presented in this paper, which displays an infection with Mirai in a Windows computer inside of a controlled testing environment. Some other honeypots have been deployed in the very same environment, in order to run some tests that are beyond of the scope of this paper (environment preparation and checking, for example).

The malware samples have been downloaded from VirusTotal and Malwr, and the infection procedure has been carried out as detailed in a Securelist guide.

4.1 Environment

The environment where the proof of concept is run is vCloud, which allows virtualization, network definition, and snapshot restoration, which are useful tools. Moreover, it also provides isolation, in order to prevent the deployed attacks from spreading outside of the environment.

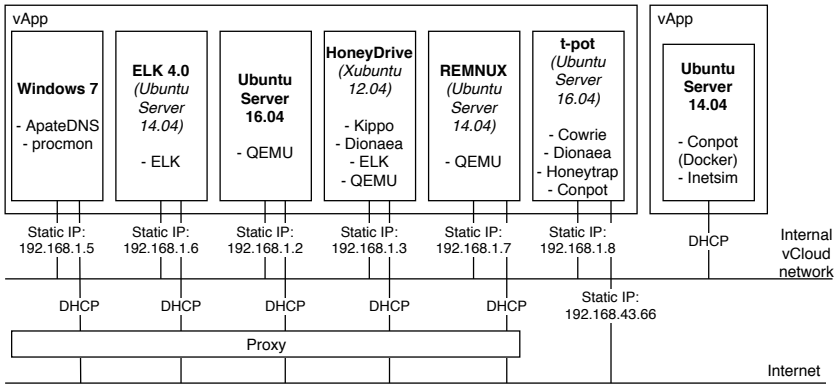

Fig. 3. Deployment in a Controlled Environment

The honeypots deployed and their connections are shown in Fig. 3. It can be seen how they are connected to an internal network which does not have access to the Internet. In this configuration, a machine is dedicated to centralizing the logs for the evaluation and validation phase. This part of the methodology was implemented using the ELK stack. This stack is a solution which combines three services: ElasticSearch, Logstash and Kibana. The reasons why these technologies were chosen are their comprehensibility and the wide amount of plugins available, which simplify many tasks. The interactions between the are pretty simple: Logstash imports the logs from other machines, ElasticSearch indexes

them, and Kibana retrieves the results from the indexes, showing them in simple and easy to understand graphs. Some of the plugins used are Filebeat, to send the logs from the honeypots to the stack, and Grok, to parse them into intelligible fields for the users.

Once the honeypots are correctly connected to the ELK stack, in order to save the logs and obtain useful information from them, and the environment is properly set to prevent the infection from spreading, the infection can be carried out.

4.2 Infection

Since the first step of the infection is to download a malicious executable file, and the machines do not have an Internet connection, a fake HTTP server has been set up inside the Windows machine with INetSim. The files needed for the infection are put there, and the requests to the URLs where Mirai is stored are redirected to the fake server through ApateDNS. Thus, the Windows machine downloads the files as if it were connected to the Internet.

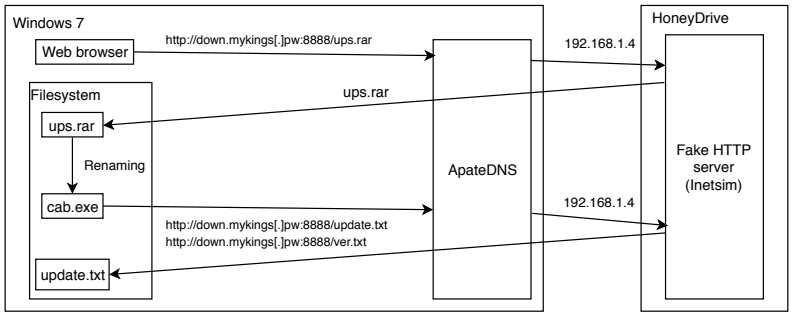

Fig. 4. Mirai infection process

Once the first file is obtained, it is renamed an executed, just as the infection guide details. If this execution is made with administrator privileges, it changes the DNS addresses of the machine (and does nothing, otherwise) to 114.114.114.114 and 8.8.8.8.

The process tries then to connect again to URLs from the same domain where the executable was located, in order to download *ver.txt* and *update.txt*. The first is available, but the second is not to be found on the malware sample databases that were accessible, so the console output is *DNS set ok. ver different web:1.0.0.7 local:, needs update.*. Since the *update.txt* file is missing, the execution aborts.

If the executable tries to download the files without ApateDNS running and without Internet connection, the message shown is *get file list failed, exit*, just before it aborts as well.

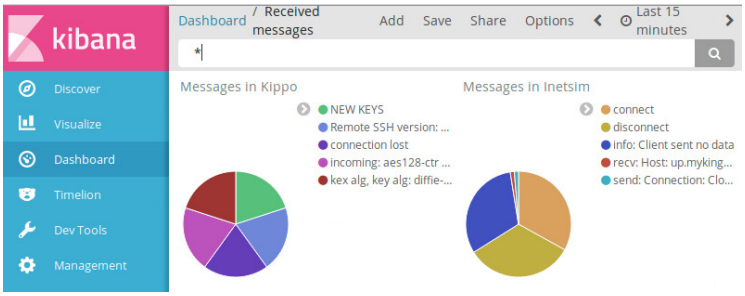

Fig. 5. Visualization of results in Kibana

In order for the executions to be logged, the events were sent to the ELK stack. The messages were parsed with the Grok plugin, split into understandable fields and indexed, used and then visualized. The messages were separated from the rest of honeypots set with Kibana indexes, in order to differentiate Mirai from other attacks.

Also, the traffic generated in the request and response of files by the Mirai executable was captured and analyzed with Wireshark, where the IP redirection and the delivery of the files can be seen.

5 Discussion and future work

The defined methodology covers more cases than those shown in the proof of concept. A fundamental step would be the definition of specific honeypots for IoT following the criterion discussed in this work. The decision on deploying a honeypot for Mirai was based on its recent impact, but defining a set of IoT honeypots in order to capture unknown attacks is a goal to be achieved in future work. Although this is already being worked on, these details cannot be shared until obtaining substantial results from attacks, since the results could be compromised if specific details of the implementation are known.

The training phase is essential before considering the deployment of honeypots, which is why it has been the focus of this work. Nonetheless, some other points not considered in the present paper are very interesting, such as the following.

- **Infection from a Mirai botnet:** Instead of manually downloading the firmware and performing the spread, it would be interesting to cause the machine to be copromised from an already existing botnet. This approack would allow seeing how the honyepot would be indexed by the monitorizing mechanisms of the Mirai-infected devices. This is not easy, since it would mean to expose the honeypot directly to attacks, and maybe participating in its spread.
- **Infection to IoT devices in a network through Windows:** Creating a honeynet for this specifica scenario, in which the Windows machine is

connected to IoT devices in the same local network, and see how they are infected and how they act.
- **Connection with ELK to make an intelligent system:** This consists on detecting events in the logs received at the ELK stack as signs of a potential infection happening based on their parameters (messages, timing, requests...) and taking the appropriate security measures.
- **Hajime infection:** Hajime is an anti-Mirai botnet, which is based on the same principle [25]. It logs on to vulnerable devices with default credentials, but blocks the access to the ports that Mirai usually checks.

6 Conclusions

Throughout the paper, it has been detailed how the IoT landscape is new, challenging and real. The average Internet daily user has one or several IoT devices, and will interact with more of them in a single day, often even without being aware of it. These devices have not been designed with enough security measures, and they are performing important and sensible roles in our lifes, regarding our privacy and security. The resulting scenario is one with vulnerable devices carrying out sensible tasks.

The difficulty of getting malware samples of IoT attacks, the possibility of simulating and emulating platforms without having real devices, the isolation layer honeypots provide, their logging of events (since IoT attacks usually erase their traces) and other particularities of IoT attacks, make honeypots the perfect tool for this work.

A methodology for this deployment is needed, because although the tools are available, it must be studied which honeypots should be deployed, how they can attract attackers, and how they can be improved based on the information obtained. This is where the proposed methodology comes into play. Every phase is thoroughly detailed, and has justified relevance in all the process. This is demonstrated by carrying out a proof of concept where one of the most worrying IoT attacks is deployed in a controlled environment, and the data obtained from this test is shown, detailing how it was carried out following the steps of the methodology.

7 Acknowledgement

This work has been financed by Ministerio de Economía y Competitividad through the projects IoTest (TIN2015-72634-EXP) and SMOG (TIN2016-79095-C2-1-R). The second author has been financed by INCIBE through the grant program for excellency in advanced cybersecurity research teams.

References

1. T. Danova, "Morgan stanley: 75 billion devices will be connected to the internet of things by 2020," *Business Insider*, vol. 2, 2013.

2. R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.
3. G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *Military Communications Conference (MILCOM), MILCOM 2017-2017 IEEE*. IEEE, 2017, pp. 267–272.
4. B. Insider, "This one chart explains why cybersecurity is so important," *Retrieved August*, vol. 16, p. 2016, 2016.
5. M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *arXiv preprint arXiv:1608.06249*, 2016.
6. Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: analysing the rise of iot compromises," *EMU*, vol. 9, p. 1, 2015.
7. J. D. Guarnizo, A. Tambe, S. S. Bhunia, M. Ochoa, N. O. Tippenhauer, A. Shabtai, and Y. Elovici, "Siphon: Towards scalable high-interaction physical honeypots," in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. ACM, 2017, pp. 57–68.
8. T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, "Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices," *Black Hat*, 2017.
9. P. Krishnaprasad, "Capturing attacks on iot devices with a multi-purpose iot honeypot," Ph.D. dissertation, PhD thesis, INDIAN INSTITUTE OF TECHNOLOGY KANPUR, 2017.
10. A. Radice, "Playing with a mirai honeypot: Mtpot," 2017.
11. M. Wang, J. Santillan, and F. Kuipers, "Thingpot: an interactive internet-of-things honeypot," 2017.
12. S. Dowling, M. Schukat, and H. Melvin, "A zigbee honeypot to assess iot cyberattack behaviour," in *Signals and Systems Conference (ISSC), 2017 28th Irish*. IEEE, 2017, pp. 1–6.
13. A. Jicha, M. Patton, and H. Chen, "Scada honeypots: An in-depth analysis of conpot," in *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*. IEEE, 2016, pp. 196–198.
14. J. P. Singh and A. Chauhan, "Detection and prevention of non-pc botnets."
15. H. Wardak, S. Zhioua, and A. Almulhem, "Plc access control: a security analysis," in *Industrial Control Systems Security (WCICSS), 2016 World Congress on*. IEEE, 2016, pp. 1–6.
16. G. B. R. R. F. Trujano, B. Chan, G. Beams, and R. Rivera, "Security analysis of dji phantom 3 standard," *Massachusetts Institute of Technology, May*, 2016.
17. M. Szabo, "Drone hacking," 2017.
18. T. Fox-Brewster, "Watch a very vulnerable usd140 quadcopter drone get hacked out of the sky," 2017.
19. M. Hughes, "Why the ikettle hack should worry you (even if you don't own one," 2015.
20. PenTestPartners, "Hacking defcon 23's iot village samsung fridge," 2015.
21. K. Munro, "Extracting your wpa psk from bathroom scales," 2015.
22. E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 195–212.
23. A. Hern, "Cloudpets stuffed toys leak details of half a million users," 2017.
24. K. Lab, "A windows-based spreader for mirai malware has been discovered," 2017.
25. S. Edwards and I. Profetis, "Hajime: Analysis of a decentralized internet worm for iot devices," *Rapidity Networks*, vol. 16, 2016.