Modeling Malware-driven Honeypots

Gerardo Fernandez(✉), Ana Nieto, and Javier Lopez

Network, Information and Computer Security (NICS) Lab,
Department of Computer Science, University of Malaga, Malaga, Spain
{gerardo,nieto,jlm}@lcc.uma.es
http://www.nics.uma.es

Abstract. In this paper we propose the Hogney architecture for the deployment of *malware-driven honeypots*. This new concept refers to honeypots that have been dynamically configured according to the environment expected by malware. The adaptation mechanism designed here is built on services that offer up-to-date and relevant *intelligence information* on current threats. Thus, the Hogney architecture takes advantage of recent *Indicators Of Compromise* (IOC) and information about suspicious activity currently being studied by analysts. The information gathered from these services is then used to adapt honeypots to fulfill malware requirements, inviting them to unleash their full strength.

Keywords: Honeypot · Malware · Adaptive · Dynamic · Intelligence · IOC

1 Introduction

According to the report issued by Symantec last year, there was an increase of 36% in the collection of unique malware samples compared to the previous year [1]. In 2016, ransomware grew considerably, affecting almost half of businesses worldwide [2]. Infections via e-mail, phishing and botnet nodes remain the most commonly used methods to compromise computers in the business environment. As a consequence, one of the biggest concerns today is how to respond effectively to malware dissemination campaigns.

Honeypot systems are designed to capture attacks by simulating real services and/or applications. They employ deception techniques that try to satisfy the attacker's demands, providing him/her with valid responses to service requests and apparently accepting modifications they want to make on the system. There are two main scenarios commonly used for deploying honeypots that differ depending on the objective pursued:

- *Replicate live services of the production environment*: showing a footprint similar to that of the services offered in the production network.
- *Research environments*: showing a configuration of honeypots that enables attacks to be captured, to later analyze new techniques used.

J. Lopez et al. (Eds.): TrustBus 2017, LNCS 10442, pp. 130–144, 2017.
DOI: 10.1007/978-3-319-64483-7_9

This paper focuses on the second scenario, specifically on the design of a capture system that can respond to attacks performed automatically. The main issue when designing this type of solution is the lack of information prior to the attack. Currently, there are principally two approaches to the problem; studying only specific scenarios (web servers, SSH/Telnet protocols, etc.), or implementing specialized trap systems for a reduced set of malware families (eg. Mirai) [3]. However, new malware attacking these honeypots will not necessarily activate all stages of the attack, due to an unfulfilled requirement.

The main contribution of this paper is the design of the Hogney architecture to capture evidence and acquire knowledge about new malware activities by using malware intelligence services. These services are designed to distribute knowledge about compromised IP addresses (for filtering systems), or serve as a platform for the exchange of information about characteristics and operation of malware. One of Hogney's goals is to integrate this type of service into the dynamic adaptation process of honeypots, designed according to the requirements of the malware.

This article is structured as follows. Section 2 details related work. A brief introduction to malware intelligence services is described in Sect. 3. Section 4 describes the components of the proposed architecture, whose interaction is analyzed in Sect. 5 using a specific attack example. Section 6 discusses the feasibility of implementing and deploying the proposed solution. Finally, the conclusions and future work are presented.

2 Related Work

There have been previous works for adapting services offered in honeypots to attackers' requests [4]. Honeytrap implements a connection interception service that dynamically selects which services to offer as stated by a pre-set configuration file. Honeyweb is able to emulate Apache, Microsoft IIS and even Netscape servers. The decision of which one to serve is taken after analyzing the URL requested and then the HTTP headers are configured according to the needs of the attacker.

Moreover, there are honeypots for the TELNET protocol that can simulate that they are running under up to 8 processor architectures [5]. The decision of which architecture to use depends on the type of command sent by the attacker. Similar work has been done with SSH [6,7] whose adaptability mechanism focuses on the interaction with the attacker through an established SSH session. SIPHON [8] focuses on the construction of honeypots using physical devices interconnected through wormholes that redirect attacks towards a set of trap devices.

With a broader scope of application, the work in [9] shows a dynamic management system of high and low interaction honeypots, deployed in a virtualized way according to the honeybrid decision engine. Decision making in this case is focused on detecting interesting traffic using pre-established rules triggered by intrusion detection engines such as Snort.

However, these approaches do not take into account specific aspects of malware behavior. In some cases it is because the scope is not intended for automatic propagation malware, but rather for manual attacks. In other cases the area of study is focused on specific replication of devices, protocols or services based on prior knowledge of attack vectors.

The Hogney architecture is intended to highlight the benefits of incorporating existing live information about current malware campaigns, recent indicators of compromise (IOCs), or/and intelligence information available through projects such as *Malware Information Sharing Platform* (MISP) [10] or *Virus Total Intelligence* (VTI) [11], in order to build an environment as close as possible to malware needs, in such a way that its whole load is unleashed and can be analyzed.

3 Malware Intelligence

We use the term *malware intelligence* [12] to refer to *malware behavior and threat information*. Sometimes the terms *threat intelligence* and *cyber threat intelligence* are also used when there is a need to describe how malware spreads, which nodes are been used and who is behind that code. Nevertheless, we think the term *malware intelligence* better represents the kind of information we need for the architecture described in Sect. 4.

Depending on the information requested, different types of malware intelligence services can be used. We classify them in three levels (L1–L3, Table 1):

L1. Services that offer lists of compromised IP addresses belonging to botnets or that are part of any current malware deployment campaign.
L2. Services that allow information about malware files or malicious URLs to be obtained, discovering the malware family, architecture and target operating system, and in many cases information related to the implementation: linked libraries, anti-analysis or anti-virus techniques, processes to which it injects code, etc.
L3. Malware information sharing services. These services will provide the most up-to-date information regarding the dissemination activities of malware. They allow access to published IOCs and to information about incidents currently under investigation, so none of the information collected has been published.

Regarding L1 services, there is a wide range of projects that list IP addresses, URLs or domains used by malware. For instance, a search for the domain *wrcwdxjh.org* produces an output similar to the following:

```
wrcwdxjh.org Intel::DOMAIN
from malwaredomains.com,locky
via intel.criticalstack.com F
```

This reveals that the domain is related to *Locky* ransomware.

L2 services provide detailed information about files and URLs linked to malware. By submitting a file to these services we get an overview of what kind of malicious activities it performs, what processes are launched, what services are used and what traces it leaves behind.

Table 1. Common information obtained from malware intelligence services

Level	Domain	Information provided
L1	IP/Domains/URL	ip-src, ip-dst, port, url, malware name
L2	File	processor, architecture, mail, PE/ELF/MACH-O executables, document specific, traffic generated by sample, ...
L3	Threat intelligence	private/public info about current threats, IOCs, correlation of incidents, ...

A common query is to search the hash of a suspicious file in order to obtain a report that contains, between other things, information about the architecture and operating system needed to run the file, communications with a domain or IP, files read or modified, libraries and methods used, file format and anti-analysis techniques implemented. If the search request does not provide result, the file is sent for a complete analysis that will generate the information needed.

L3 services are useful when there has been no information collected by L1 and L2 services or this information is inconclusive. L3 services provide access to intelligence information, shared by incident response teams or malware analysts, among different organizations. Sometimes this type of service gives access to information about active campaigns of malware not yet published, because they are currently being studied by analysts and are therefore only labelled as *suspicious activities*.

For instance, searching the hash value *64973870ed358afec07b0ebb1b70dd40* of a file produces a response in which that hash is related to a current propagation campaign of Locky ransomware. The code below shows part of the response obtained when searching that hash. In addition to the information related to that file, several IPs belonging to Locky command and control nodes are also present.

```
<Attribute>
    <id>3367</id><org_id>2</org_id>
    <info>Malspam (2016-03-16)</info>
    <value>http://188.127.231.116/main.php</value>
</Attribute>
<Attribute>
    <id>366617</id><type>md5</type>
    <category>Payload delivery</category>
    <to_ids>1</to_ids>
    <uuid>56e6c1a6-3b5c-457e-9443-473402de0b81</uuid>
```

```
    <event_id>3354</event_id>
    <distribution>5</distribution>
    <timestamp>1457963430</timestamp>
    <comment>- Xchecked via VT</comment>
    <sharing_group_id>0</sharing_group_id>
    <deleted>0</deleted>
    <value>64973870ed358afec07b0ebb1b70dd40</value>
    <ShadowAttribute/>
    <RelatedAttribute>
        <Attribute><id>3355</id><org_id>2</org_id>
            <info>Malspam (2016-03-14)</info>
            <value>64973870ed358afec07b0ebb1b70dd40</value>
        </Attribute>
    </RelatedAttribute>
</Attribute>
```

4 The Hogney Architecture

The purpose of the Hogney platform is to create trap environments to capture activity performed by malware, adapting them progressively as new evidence is generated to determine which action will be triggered next. Therefore, it is necessary to design an architecture that allows analysis according to the three stages of malware: (1) exploration, (2) infection and (3) execution of the payload.

During the *exploration* phase the attacker tries to discover which services are running, checking them for vulnerabilities. An attacker succeeds if he finds a vulnerable service for which he has an exploitation mechanism. Hogney tries to deduce the type of service that the attacker is looking for, offering a honeypot that meets his needs.

In the case the exploration is successful, the attacker moves on to the second stage, *infection*, where the infection code is launched against a vulnerable service for a wide range of reasons. At this point it is important to analyze the infection vector to discern what kind of action the attacker wishes to unleash on the honeypot: inject code, upload a file, leave code, manipulate files, send an email, login to the system, etc. As far as possible it should be made clear to the attacker that such action has been successfully carried out.

Thus, we reach the third stage: *execution of the payload*. This is where, depending on the type of activity, the execution of the code is simulated, the downloaded file is executed in a controlled environment, or the e-mail is sent. However it is important to note that all the modifications executed in the victim's environment have to be logged.

These three phases are managed by the control components shown in Fig. 1: the interception module (IM), dynamic configuration module (DCM) and monitoring of the generated evidence (EM). These three modules are fed with information that allows the next step that malicious code intends to carry out to be predicted.

A key element of the Hogney architecture is the use of malware intelligence services. There are a multitude of services that can be used to obtain information about malware activity, either through searching IP addresses belonging to malware campaigns, querying known infection vectors, signatures or search patterns, or by the explicit execution of malware samples and observation of the actions and changes made. Hogney orchestrates this information, adapting honeypots to the three aforementioned stages. To this end, there are two different trap environments: (i) honeypots specializing in certain protocols and/or services, and (ii) highly interactive environments in which to execute files generated by malware.

The relationship between the components is detailed in the following subsections.

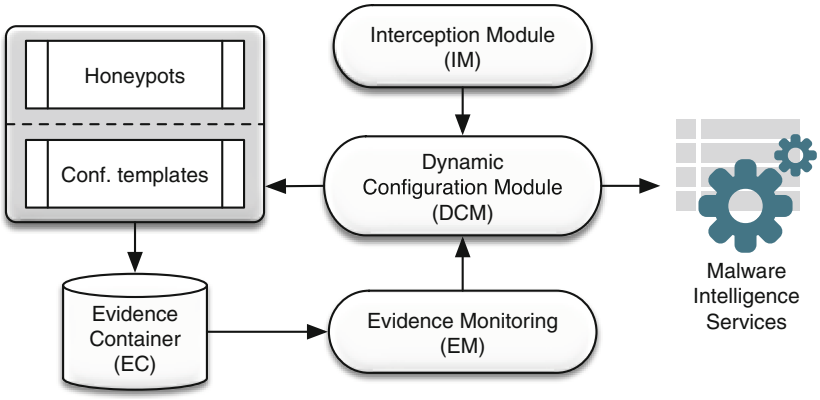

Fig. 1. Architecture diagram

4.1 Interception of Connections

A typical honeypot is configured to listen to a series of predefined ports. When a connection is established with one of them, it responds as described in its configuration. Some low-interaction honeypots only receive requests and store them (e.g. honeyd) while medium interaction honeypots (Inetsim [13]) are able to operate at the service level, responding to sender requests in accordance with the behaviour configured by the operator.

The component for the *interception of connections* (IM) will (i) listen to all ports studied, (ii) receive and accept connections and (iii) send service requests to the DCM component for the configuration of honeypots (Fig. 1). Such requests should include all the information that may have been collected at the time of establishing the connection (IP, destination/source ports, protocol headers, etc.) so that the DCM can more accurately estimate the honeypot with the highest probability of success for this connection.

4.2 Configuring Trap Services

This service is called the *Dynamic Configuration Module* (DCM) and is able to dynamically discern which honeypot is the most suitable for the type of malware involved. To achieve this objetive it is be necessary to have a repository of pre-configured honeypots, set up in such a way that it is easy to switch from one to another depending on the malware's requirements, or to modify its configuration.

For instance, the SMTP protocol is frequently used by malware to send e-mails with attachments containing some kind of malicious code. If a request is received at port 25 it will be redirected to a honeypot capable of handling the reception of the e-mail. However a request to the HTTP/HTTPS port can have very different objectives: it could be an attack on the Apache web server (Windows, Linux, ...), on IIS, etc. It could additionally be an attack on a particular version of WordPress, Joomla, etc. This diversity complicates the provisioning of a honeypot that will successfully fit the attack.

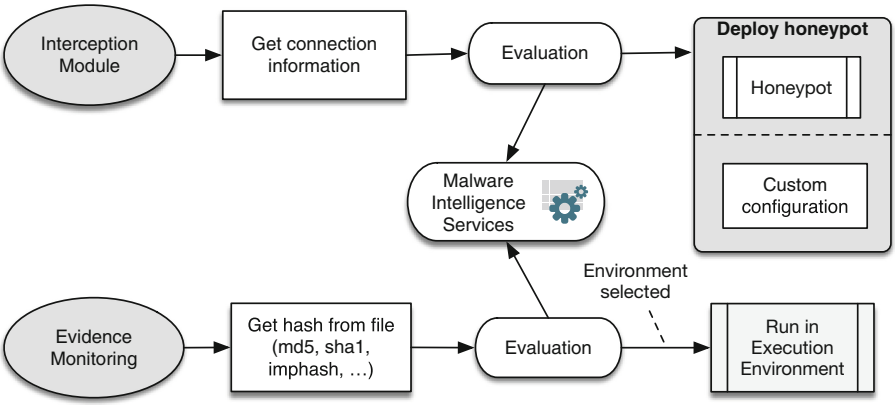

Fig. 2. Two different execution flows of the DCM module

Consequently, this component must process the connection requests in order to choose a honeypot, to initiate or adapt, for the malware in question. This decision is based on the data gathered about the characteristics of the connection:

- **Source IP:** check whether this IP address belongs to a malware campaign currently in progress. In this case, information must be retrieved about the malware family and what services and applications it affects.
- **Destination IP:** if a file has been executed and it is trying to establish an external connection, firewall rules must be adapted to allow this traffic. At the same time, all traffic generated must be recorded for further analysis.
- **Protocol headers:** first packets of many service level protocols contain information about the type of service expected. This must be monitored in order to decide which honeypot, offering that service, could be set up and run.

- **Service information:** destination IP addresses, file and folders, running processes, DNS entries, etc. This information facilitates the configuration of suitable honeypots.
- **Related files:** downloaded files contain useful information about the target operating system, required libraries, resources needed, etc., which can later be used to select a suitable execution environment.

To illustrate, Fig. 2 shows a graph of the execution flow of this component in two specific cases: (i) a service request is received from the interception module, and (ii) the request is received from the evidence monitoring component upon detection of a file created in the evidence container.

In the first case, the information about the received connection (source/destination IP address, protocol, service data, destination files/folders, etc.) must be analyzed. This information will then be used to try to find out which malware is behind that connection. Hence, queries to *external intelligence services* are launched to look for any evidence of malware based on the information collected. As a result, the information obtained from these services is used to adapt a honeypot, already pre-configured, so that it is as close as possible to the scenario that the attacker expects to find. The inquiry process is shown in Fig. 3, where the diagram shows the process which determines whether the IP is linked to malware activities. L1 services are used to determine whether or not the IP is part of any current malware campaign. If there are not results, a query to L2 and L3 services is launched.

The second case reflected in Fig. 2 corresponds to a scenario in which the attacker has managed to download some type of file, either within a honeypot or when running in an execution environment configured by DCM. The monitoring process (EM) will detect the existence of any new evidence, in the form of a new file stored in EC, and will ask DCM to deploy an execution environment for it. Again, this will initiate another request to malware intelligence services to obtain information about that file, in order to gather information about how to build a suitable environment for it. This process is reflected in Fig. 4.

4.3 Evidence Monitoring

The architecture designed includes a container of evidence to store any type of content generated during the attack, regardless of whether it is an executable, interpreted code, binary code, images, documents, etc. The objective of this container is twofold: to gather as much information as possible about the actions carried out by malware, as well as to facilitate the continuity of the attack process, by activating the different stages implemented in the malware.

The *evidence monitoring* (EM) component is continuously monitoring the creation of new evidence. When a new piece is detected, a request is sent to the DCM containing the characteristics of the evidence (file type, operating system, etc.). Then, a new execution environment is set up to analyze this evidence.

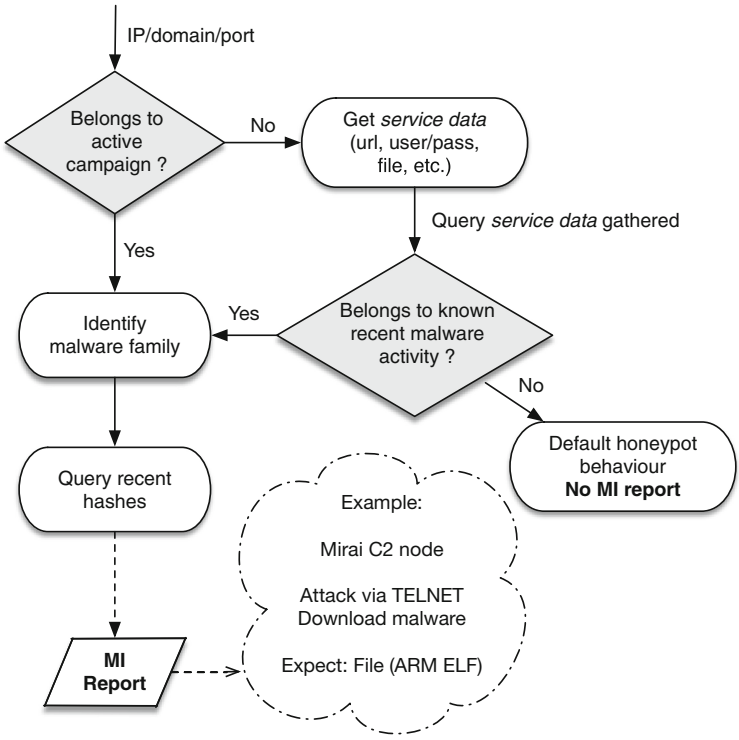

Fig. 3. Requesting information about an IP address

4.4 Provisioning of Honeypots

Thus far, each element of the architecture described corresponds to a controlling or monitoring process. Hogney also needs a set of preconfigured honeypots for common scenarios susceptible to attack. Fortunately, there are a multitude of honeypots specialized in certain environments [4] (Cowrie for ssh, glastopf for HTTP, conpot for PLCs, jackpot for SMTP, elastichoney for elasticsearch, etc.) and others of more general scope (inetsim). Hogney includes them as the basis for our current set of preconfigured honeypots.

However, for a honeypot to be used by DCM, it needs to fulfill some requirements:

- Easily configurable by modifying text files.
- Provide options for configuring banners, service folders, responses to protocol commands, etc.
- Allow configuration of the listening network interface.
- Include capabilities for recording activities performed by attackers.

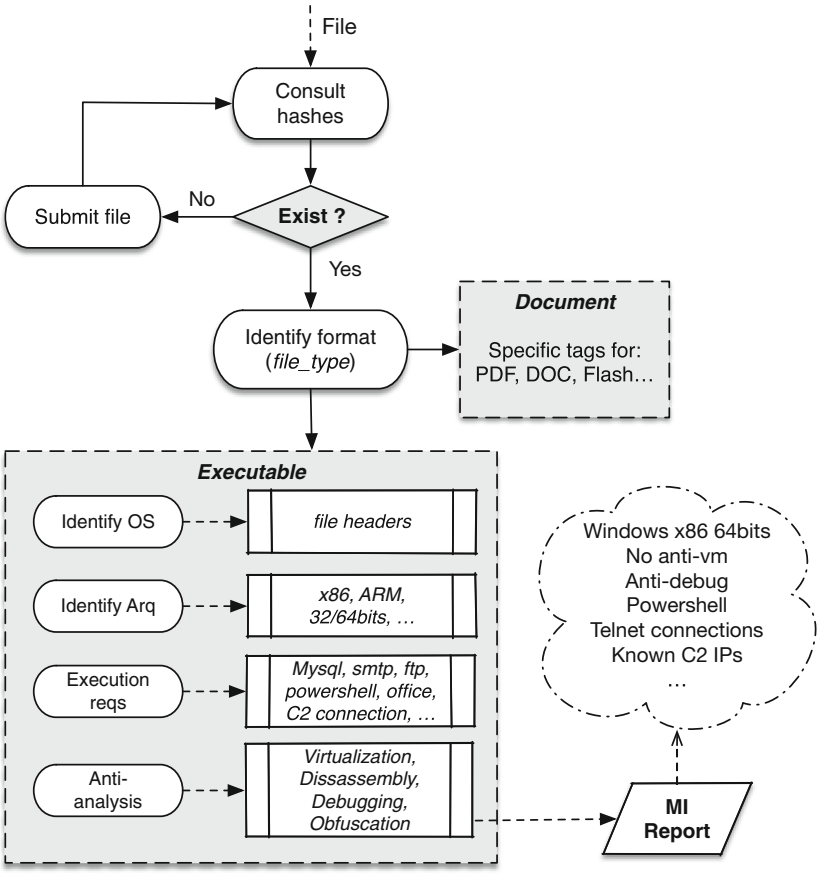

Fig. 4. Requesting information about a downloaded file

Compliance with these requirements will allow DCM to modify the configuration files according to the parameters received in a request for service (which folders should be available, which applications, which protocol banners are expected to be found, etc.).

In addition to specific honeypots of low, medium or high interaction it is necessary to have execution environments where the evidence obtained can be processed. These environments, also considered as high-interaction honeypots, comprise both virtualized and physical machines managed by an orchestrator process. *Cuckoo Sandbox* [14] has been selected for the deployment of the execution environments, largely because it covers our main needs:

- It provides an API that can be used to send files to preconfigured analysis environments.
- It issues an activity report after the execution of files, which is easy to process automatically.

- Good escalation capabilities, offering different mechanisms of adaptation where an increasing number of analysis environments is needed.
- The analysis performed is good enough for the automation needed in our platform.
- Freely available.

5 Hogney Behavior Under the Mirai Attack

This section details the relationship between the components of the Hogney architecture and its behavior when exposed to a well known malware propagation attack. We have chosen Mirai because it produces a rich relationship between the different components of the architecture. However, we want to remark that we have designed Hogney independently of any malware family, and for the sake of brevity we decided to only present the Mirai case.

Mirai is a botnet that principally attacks typical embedded devices in IoT. In 2016 it became famous for causing a DDoS attack on a DNS service provider named DYN that led to the disconnection of services like GitHub, Twitter, Reddit, Netflix, Airbnb among many others. It mainly attacked TELNET services through dictionary attacks, to turn compromised devices into new botnet nodes to be used in subsequent DDoS attacks [15].

Under this scenario (Fig. 5), the IM receives a request for connection to port 23. Next, it consults intelligence services regarding the source IP of the connection to determine whether the telnet honeypot needs to be adapted in some way. The information obtained reveals that the originating node belongs to a Mirai botnet.

Mirai makes telnet connections for two reasons: (i) from a bot to detect that this service accepts known credentials and (ii) from a *loader* to cause the download of a malware file. Since Mirai has malware versions for different architectures, the default configuration of the honeypot can be modified by the DCM to show one of the architectures determined by the intelligence service (ARM in this case).

In the use case modeled in Fig. 5 we depicted the second case, where Hogney is receiving a connection from a Mirai loader. Here, the honeypot deployed records the commands to be executed and even the files that the *loader* has specified to be installed. There are several honeypot implementations of the telnet services (such as Cowrie) that are able to correctly interpret regular file download commands like *curl* or *wget*. The files downloaded are stored in the evidence container.

Once the creation of the downloaded file has been detected, the monitoring process sends a request to the DCM to prepare a honeypot for it. It uses the intelligence obtained from the analysis of the file (e.g. malware for ARM 32-bit architectures in the Linux environment), to create an emulation environment to execute the file (like QEMU [16]).

After executing the file in an environment deployed by the DCM, a connection is made to an external IP address that, after contrasting it with the intelligence available, could reveal a new C2 node of Mirai, or confirm that this is already

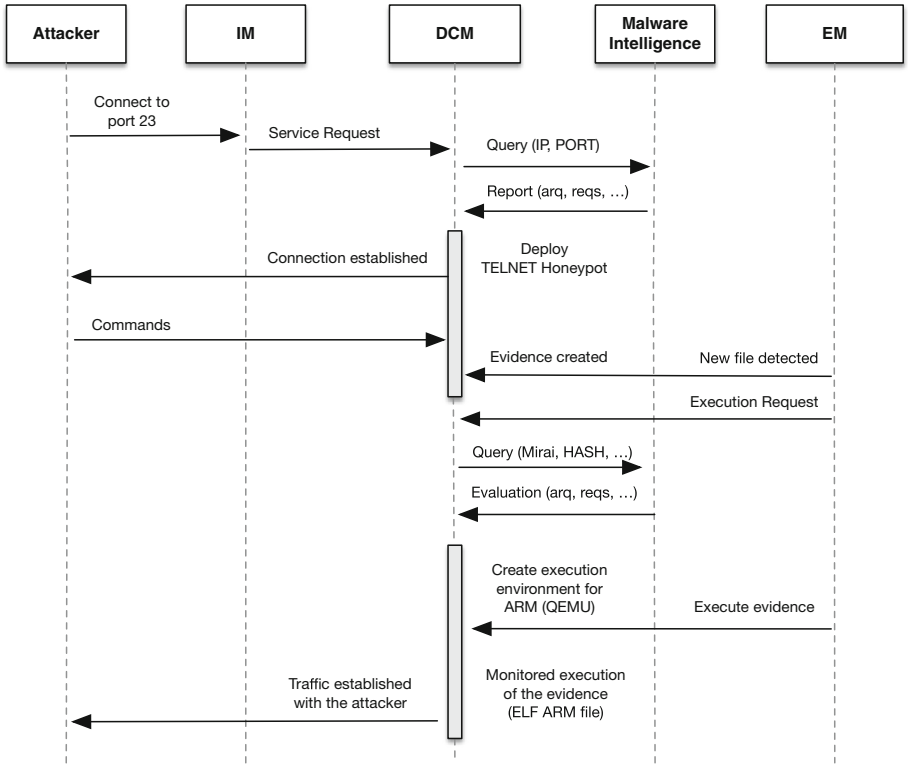

Fig. 5. Hogney components interaction under the Mirai attack

known. Keeping a honeypot acting as a bot node of Mirai allows access to the information about how the C2 node operates, such as the execution of DDoS attacks (and the IP addresses affected).

6 Implementation Discussion

The proposed architecture will eventually employ a multitude of available honeypots, some for generic use and others focused on certain protocols, all together offering a wide range of solutions for the capturing of evidence. However, the construction of a malware capture and analysis platform, like the one designed in this paper, involves overcoming a number of obstacles, some related to design criteria and others regarding implementation. In this section, the implementation of the main components (IM, DCM and EM) of the Hogney architecture will be discussed.

With respect to the *interception module* (IM) there are currently applications, such as *honeytrap*, which work in a similar way as that described for IM in Hogney. Honeytrap is a process that remains listening on all ports, waiting for

a connection to be requested. When this happens the connection is redirected to the predetermined honeypot to attend the request. It also has the ability to apply reverse proxy and *mirroring* techniques to address incoming connections. The only drawback of this application is that the honeypot that is served is defined according to the port accessed by the attacker. However, it could serve as a basis for building the IM component that the architecture needs.

The *evidence monitoring* (EM) component must correlate the different events generated, so that each piece of evidence obtained must be linked to the initial connection that triggered its creation. Consequently, it is necessary to register the execution of the three stages of the attack under the same case in a way that facilitates correlation and generation of reports a posteriori.

So far, we have detected three different engines in Hogney that generate evidence:

- The IM component, recording connections that are established with honeypots.
- Honeypots of low and medium interaction, which generate files and/or commands.
- High-interaction honeypots, which create files and connections as a result of running malware on them.

In this sense, the selection of Cuckoo Sandbox [14] for the management of high interaction honeypots will benefit the organization of the evidence executed in virtualized and physical machines. This is because Cuckoo creates a different structure of folders and files for each file launched on an analysis machine. This structure contains all the collected evidence provoked by the execution of the file.

The logic behind the choice of which trap environment to launch is one fundamental aspect to be developed for the DCM component. It must decide, depending on the information gathered, which is the most appropriate environment to serve to the attacker. For this purpose, it will use information available through malware information services, grouped into the three levels described in the Sect. 3.

With regard to L1 services, there is a wide range of projects that offer specific malware information: indicators of compromise (IOCs), types of devices affected, sources of IP used, etc. Perhaps the most convenient way to maximize results when searching, is to use platforms that allow the user to launch queries using several combined sources. For instance, to date the *Critical Stack Intel* [17] includes 118 feeds and offers an application to query them externally.

There are several alternatives for L2 services, although we have selected two solutions that, a priori, can cover our the needs with respect to file/URL analysis:

- *Virus Total Intelligence* (VTI) [11]: this service allows searches with a multitude of parameters. In addition to the hash of a file, it is possible to consult IPs or URLs linked to malware, as well as search for characteristics of the file/URL itself (operating system, architecture, resources contained in the file, system resources used, etc.). Although Virus Total offers free access to its core service for antivirus evaluation, VTI is a paid service.

– *Hybrid Analysis de PAYLOAD Security* [18]: similar to VTI but with fewer options for searching and a smaller base sample base. Nevertheless, the analysis performed on the files adds some interesting features not provided by VTI, such as the anti-analysis techniques implemented. This service, unlike the previous one, offers free access to its intelligence database by limiting the number of queries a user can make.

Hogney will use the MISP platform [10] to access malware intelligence information (L3 services). MISP offers different ways of sharing information, providing an API for querying and obtaining events in different formats (MISP XML, MISP JSON, STIX, STIX JSON, CSV, etc.). Hogney will use a custom installation of MISP connected to an external community for accessing shared data. We will add some custom attributes to our own community created in MISP to improve the interoperability between Hogney and MISP for configuring honeypots.

7 Conclusions and Future Works

The ability of malware intelligence services to provide early recognition of malware traces is noteworthy. It is no surprise therefore, that this is why they are widely used in many defense systems such as IDS and firewalls. Even services like proxies and DNS use the information they provide to avoid leading the user to malicious sites.

Until now, these services have not been used for the dynamic deployment of honeypots. The Hogney architecture, proposed in this paper, shows the versatility that they can provide to configure honeypots in the initial stages of an attack. This functional architecture provides a set of components for the automatic deployment of honeypots according to the intelligence information obtained.

The next steps will be taken towards analyzing the convenience of adopting machine learning techniques for the core of the DCM component. There has been some progress made in creating a machine learning dataset [19] implementing the MIST representation [20] of malware behavior. As stated by MIST's authors, "*the representation is not restricted to a particular monitoring tool and thus can also be used as a meta language to unify behavior reports of different sources*". We could integrate the information gathered from malware intelligence services to quickly create an up-to-date dataset for the DCM component.

Acknowledgments. This work has been funded by Junta de Andalucia through the project FISICCO (TIC-07223), and by the Spanish Ministry of Economy and Competitiveness through the project IoTest (TIN2015-72634-EXP/AEI).

References

1. Internet security threat report: vol. 21, Symantec, Technical report, 2016, April 2016

2. SentinelOne: Sentinelone ransomware research data summary (2017). https://go.sentinelone.com/rs/327-MNM-087/images/Data%20Summary%20-%20English.pdf
3. Cymmetria: Mirai open source iot honeypot (2016). http://blog.cymmetria.com/mirai-open-source-iot-honeypot-new-cymmetria-research-release
4. Nawrocki, M., Wählisch, M., Schmidt, T.C.: A Survey on Honeypot Software and Data Analysis. arXiv.org, vol. 10, pp. 63–75 (2016)
5. Pa, Y.M.P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., Rossow, C.: IoTPOT - a novel honeypot for revealing current IoT threats. JIP **24**(3), 522–533 (2016)
6. Pauna, A., Patriciu, V.V.: CASSHH – case adaptive SSH honeypot. In: Martínez Pérez, G., Thampi, S.M., Ko, R., Shu, L. (eds.) SNDS 2014. CCIS, vol. 420, pp. 322–333. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54525-2_29
7. Wagener, G., State, R., Engel, T.: Adaptive and self-configurable honeypots. In: Integrated Network Management (IM) (2011)
8. Guarnizo, J., Tambe, A.. Bhunia, S.S., Ochoa, M., Tippenhauer, N.O., Shabtai, A., Elovici, Y.: SIPHON - Towards Scalable High-Interaction Physical Honeypots. CoRR, vol. cs.CR (2017)
9. Fan, W., Fernández, D., Du, Z.: Adaptive and flexible virtual honeynet. In: Boumerdassi, S., Bouzefrane, S., Renault, É. (eds.) MSPN 2015. LNCS, vol. 9395, pp. 1–17. Springer, Cham (2015). doi:10.1007/978-3-319-25744-0_1
10. Wagner, C., Dulaunoy, A., Wagener, G., Iklody, A.: Misp: the design and implementation of a collaborative threat intelligence sharing platform. In: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, pp. 49–56. ACM (2016)
11. G. Inc.: Virus total intelligence (2017). https://www.virustotal.com
12. Porcello, J.: Navigating and Visualizing the Malware Intelligence Space, pp. 1–7, November 2012
13. Hungenberg, T., Eckert, M.: Internet services simulation suite (2014). http://www.inetsim.org
14. Guarnieri, C., Tanasi, A., Bremer, J., Schloesser, M.: The cuckoo sandbox (2012)
15. Angrishi, K.: Turning internet of things (IoT) into internet of vulnerabilities (IoV): Iot botnets, February 2017
16. Bellard, F.: Qemu, a fast and portable dynamic translator. In: USENIX Annual Technical Conference, FREENIX Track, pp. 41–46 (2005)
17. Critical Stack Inc.: Critical stack intel // feed (2017). https://intel.criticalstack.com
18. Payload Security.: Free automated malware analysis service (2017). https://www.hybrid-analysis.com
19. Ramilli, M.: A machine learning dataset for everyone (2016). http://marcoramilli.blogspot.com.es/2016/12/malware-training-sets-machine-learning.html
20. Trinius, P., Willems, C., Holz, T., Rieck, K.: A Malware Instruction Set for Behavior-Based Analysis. Sicherheit (2010)