HAW HAMBURG

Bachelorarbeit

Miguel Dias

Automated Identification of Attacking Tools in a Honeypot

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Miguel Dias

Automated Identification of Attacking Tools in a Honeypot

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr. Martin Becke

Eingereicht am: 23 January 2019

Miguel Dias

Thema der Arbeit

Automated Identification of Attacking Tools in a Honeypot

Stichworte

honeypot, malware, automated, attacking tools, indicators of compromise

Kurzzusammenfassung

Die Untersuchung von Malware ist ein wiederkehrendes Thema in der Sicherheitsumgebung. Das für diesen Job geeignete Werkzeug ist Honeypot. Als stark überwachte Netzwerk-Lockvogel kann man sich mit einer hohen Anzahl von Treue-Angriffen gegen ihn identifizieren. Diese Zuverlässigkeit wird verwendet, um die Angriffe zu untersuchen und zu untersuchen. Sie können mit dem Ziel verwendet werden, Signaturen dieser Angriffe zu erstellen. Wenn der Hacker ein Programm zur Ausführung des Angriffs verwendet, ist es daher einfach, ein Aktivitätsmuster im Netzwerk zu erstellen. Es ist möglich, die Aktivität des Angriffswerkzeugs rückzuentwickeln und ein Muster von Indikatoren aufzubauen, das sie in einer Gruppe von Indikatoren gruppiert, die diesen Akteur identifizieren.

Miguel Dias

Title of Thesis

Automated Identification of Attacking Tools in a Honeypot

Keywords

honeypot, malware, automated, attacking tools, indicators of compromise

Abstract

The study of malware is a recurring topic in the security environment and the tool right for that job is Honeypot. Being a heavily monitored network decoy is possible to identify with a high rate of fidelity attacks performed against it. This reliability is used to research and study about the attacks it captures. They can be used with the goal of

creating signatures of those attacks. Therefore if the hacker used a program to execute the attack it's easy to create a pattern of activity in the network. It's possible to reverse engineer the activity of attacking tool and build a pattern of indicators that it throws clustering them in a group of indicators identifying that actor.

Contents

List of Figures

List of Tables

1 Introduction

The world is so much depending on Network/ the Internet, as the Internet of Things (IoT) is taking over the world in the area of communication, data warehousing and device communication. Due to the amount of data and information available over the network, data theft has become part of network activities, as network intrusion occurs on daily basis.[1] And the growing interest in taking over systems lead to a spread development in ways of attacking and the creation of tools to do so, some don't even demand much knowledge on the topic just a few clicks and you are inside your target machine.

All this different ways of compromising systems left most cyber defenders one step behind when trying to patch systems or configure protection mechanisms against known attacks and existing breaches. Notwithstanding the fact, there were also a evolution on ways to identify and fight against the attacks created, one of those is Honeypot.

On the contrary of most traditional security systems, honeypots are a security resource whose value lies in being probed, attacked or compromised. The idea might seem very counter-intuitive but creating a system meant to be hacked brings a lot of advantages and creates excitement within cyber security world.

Alongside other criminal activities, little has been revealed on how attackers operate what tools they use, how they learn to hack, and what motivates them to hack systems. Honeypots give us an opportunity to have a glance into this world by monitoring attackers break into and control a honeypot, to learn how these individuals operate and why. Honeypots give us another advantage which is the ability to take the offensive.[16]

Traditionally, the attacker has always had the initiative. They control whom they attack when, and how. The defender mindset of the security community is building security measures to prevent the bad guy from getting in, and then detect whenever those preventive measures fail. But as any good chess strategist will tell you, the secret to a good defense is a good offense. Honeypots give advantages by giving the control when is the hacker allowed to attack.

Granting this control and inside view on the attack makes Honeypots great to collect data and investigate the origin, behaviour and reasons of an attack. So they can be used to create predictive information increasing defenders chances of protecting their systems against future threats.

Although people are often prone to making mistakes during analyses or, possibly, when trying to establish relationships between multiple features. This makes it difficult for them to find solutions to certain problems. Machine learning can often be successfully applied to these problems, improving the efficiency of systems and the designs of machines.

1.1 Objective

The main objective of the research is to to design and develop a high accuracy automated system that can identify hacking tools. Other objectives of the research are listed below:

- To design and implement a honeypot system to detect attack/intrusion.
- To analyze intrusion data and extract IOC from analyze data.
- To extract and develop IOC rules for identifying hacking tools.

1.2 Structure of thesis

This papers is divided in 4 main chapters. After a main introduction , in the chapter 2 is covered the current research that exists about honeypot and how they stand against other security assets together with systems to create Indicators of compromise.

The third chapter talks about the Architecture used for the extractions of attacking data and the methods used.

The fourth joins the data gathered of the attack and tries to cluster them in a way to create write assumtions about the hacking tool used behind it.

Lastly, in the chapter 5 will have a general overall look over the paper and conclusion of the research.

2 Background and current research

In this chapter, will be discussed the reason why Honeypots were created in the first place, how they compare with other existing security assets and how they are the best tool to gather data which can be transformed using machine learning alghorims to create clusters of Indicators of compromise isolating the activity of a specific hacking tool.

2.1 Honeypots

The first public available honeypot was Fred Cohen's Deception ToolKit, in 1998, which was intended to make it appear to attackers as if the system running DTK that had a large number of widely known vulnerabilities.[4]

Over the years, this security asset evolved and now it's possible to define it , according to Spitzner [18], as security resource whose value lies in being probed, attacked, or compromised thus activities on honeypots can be considered suspicious by definition, as there is no point for benign users to interact with these systems. [8]

In practice, honeypots are computers which masquerade as unprotected. The honeypot records all actions and interactions with users. Since honeypots don't provide any legitimate services, all activity is unauthorized and possibly malicious. It's goal is to collect high value data on attacks and attackers by monitoring the state of a real operating system or services.[11]

But this system has a weakness by nature if no one attacks or tries to compromise it, doesn't collect or help to protect against any attack. Regardless, is still considered by many a valuable security tool with some active nature. Other security tools such as firewall or IDS are completely passive, for that their task is to only prevent or detect attacks. Honeypots in other hand also gives way information about new intrusions and new attacks not documented. And that makes honeypot outstanding to aid other security tools so it's also possible to complement it's definition with a role of security tool whose value lies in actively luring attacks to attain intrusion information and improving performance of other security tools such as IDS; and also as technology whose value lies in being an alternative methods for network security.[20]

Inside a honeypot there is two key elements: the **decoys** and the **security programs**, and both combined are used to deliberately allow unauthorized access through fake vulnerabilities to then capture and identify information for security investigation. [5]

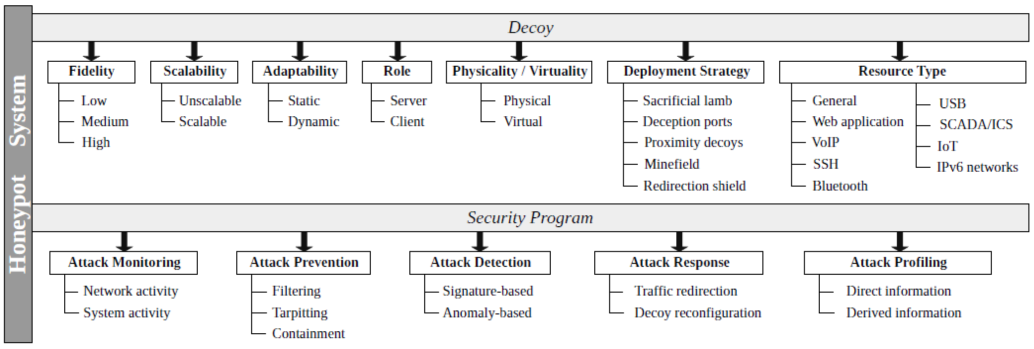


Figure 2.1: Honeypot System [5]

2.1.1 Decoy

The **decoy**, or also referred as **honeytoken** [17], can be any kind of information resource or fake digital entity that it's placed visible and no one should interact with it. Any interaction with a honeytoken most likely represents unauthorized or malicious activity, aiming to capture data. One of the most important decoy's characteristics is it's fidelity, which is a criterion that denotes the degree of exactness of an information system resource that the decoy provides to the attack. It classifies the interaction into three levels: low, medium or high.[5]

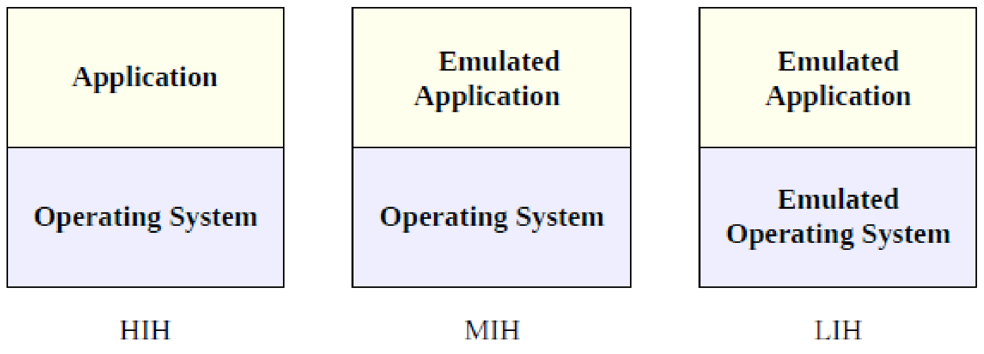


Figure 2.2: Types of Honeypots [5]

Low Interaction Honeypots(LIH)

Emulated services with a limited subset of the functionality they expected from a server, with the intent of detecting sources of unauthorized activity.[12] For example, the HTTP service on a low-interaction honeypot would only support the commands needed to identify that a known exploit is being attempted.

Medium Interaction Honeypots(MIH)

Unlike to LIH, MIH does not implement TCP/IP stacks by themselves. Instead, MIHs, e.g. Dionaea and Cowrie , bind on sockets and leave the operating system do the connection management. In contrast with LIHs that implement network protocols, the simulation algorithm of MIHs is based on emulating logical application responses for incoming requests. Thus, the request arriving to the MIH will be watched and examined, and the fake responses will also be created by the security program of the MIH.[5]

High Interaction Honeypots(HIH)

High-interaction honeypot (HIH) is fully functional system which can be completely compromised by adversaries. Its decoy is often a genuine system, such as Argos and Cuckoo Sandbox. Because the fully functional honeypot can be compromised, the HIH must equip security toolkits for system activity capture and outgoing traffic containment.

In sum, the honeypot's fidelity it's the characteristic that influences the amount of data possible to gather from the attacks the most.

	Work to Install and Configure	Deploy and Maintain	Information Gathering	Level of risk
LIH	Easy	Easy	Limited	Low
MIH	Medium	Medium	Variable	Medium
HIH	Difficult	Difficult	Extensive	High

Table 2.1: Interaction Levels Differences [16]

So it's possible to conclude that HIH would one of the best to identify and create new signatures for attacking tools because it's the only one that not limits the protocols emulated and because is a real system can analyze fully the interaction from the outside and to the outside. Although is the one that raises the most risks because can attack other machines if the attacker gains control, therefore the MIH it's the safer choice and allows a substancial information gathering.

2.1.2 Security Programs

The security programs will cover the attack monitoring, prevention, detection and responses to the honeypot and of the interactions with the decoys aiming to detect a intrusion of some sort and generate the proper notification. There are two detection approaches: **signature-based** and **anomaly-based**.

Signature-based detection is based on detecting the well-known attacks by recognizing malicious patterns. This approach is often used in production environments to discover unauthorized activity and generate alerts . This type of honeypot is often called **production honeypot** and is aimed to emulate well-known vulnerabilities to lure intrusion so that intruders will be deceived by being forced to waste time to interact with the honeypots. Production honeypots are often LIHs and MIHs that have little or no interaction with the attacker in order to minimize the risk of infecting other production systems in the context.[5]

On the other hand, **anomaly-based detection** means detecting unknown attacks by discovering deviations from normal behaviour patterns. The honeypots using this detection approach are always used in the research environment as research honeypots. A research honeypot is designed to detect anomaly attacks and investigate the unknown signatures.[5]

Thus, research honeypots are often more powerful than production honeypots. HIHs and hybrid honeypot systems are always used as research ones to provide full functional systems. A wider assortment of data can be captured to facilitate further investigation for many purposes.

HONEYPOT

Advantages [12]

- Fewer false positives since no legitimate traffic uses honeypot
- Collect smaller, higher-value, datasets since they only log illegitimate activity
- Do not require known attack signatures, unlike IDS

Disadvantages [12]

- Can be used by attacker to attack other systems
- Only monitor interactions made directly with the honeypot - the honeypot cannot detect attacks against other systems
- Can potentially be detected by the attacker

In sum, Honeypots alone do not solve a specific problem [18] and attacks on other systems will go completely undetected. They are not designed specifically to detect blackhats or prevent attacks actively. Instead, they are a highly flexible and simple tool with the purpose of divert attacks from real systems, and gather information about that attack.

2.1.3 IDS

The IDS basically monitors network traffic for activity that falls within the banned activity in the network. The IDS main job is gives alert to network admins for allow them to take corrective action, blocking access to vulnerable ports, denying access to specific IP address or shutting down services used to allow attacks. This is nothing but front line weapon in the network admins war against hackers. This information is then compared with predefined blueprints of known attacks and vulnerabilities.[2]

Host intrusion detection They monitor the behavior of applications on the host by observing the interaction of those applications with the underlying operating system. In practice, security relevant interactions typically take the form of system calls, and so their scheme works by examining the trace of system calls performed by each application.[19]

Network intrusion detection actually deals with information passing on the wire between hosts. Typically referred to as "packet-sniffers," network intrusion detection devices intercept packets travelling in and out in network along various communication mediums and protocols, usually TCP/IP. Once captured, the packets are analyzed in a number of different ways. Some IDS devices will simply compare the packet to a signature database consisting of known attacks and malicious packet "fingerprints", while others will look for anomalous packet activity that might indicate malicious behaviour. [2]

2.1.4 Honeypot vs. IDS

Honeypot [20]

- Detects compromises by virtue of system activities
- Can identify new or unknown attacks
- No false-positives
- Detects incoming and outgoing activity
- Can be compromised

IDS [20]

- Compares intrusions with known signature.
- Only identifies documented attacks
- Can generate false-positives from valid interactions
- Depends upon signatures to find out about intruder activity

If analyzed carefully, it is possible to conclude that combined, this systems can generate a outstanding duo improving the respective negative points of each other and creating a better component overall, able no detect knowed vulnerabilities (IDS) while comparing with new data or variations of an attack and take aside false-positives (Honeypot).

2.2 General Attack Structure

For a successful attack to take place there's about 7 steps an attacker follows

- **Reconnaissance** is the first procedure to a successful attack, which the attacker will gather as much information about a target systems as possible, all the following steps will the depend of the data gathered and analyzed here.
- **Scanning** After the victim is analyzed and identified in the previous point the following path would be is usually accomplished by scanning an organization's network with tools easily found on the internet to find entry points.
- **Access and escalation** In most cases for having the ability of moving freely in the system and having access to the core of the machine, privileged access is needed. Rainbow tables and similar tools help intruders steal credentials, escalate privileges to admin, and then get into any system on the network that's accessible via the administrator account. Once the attackers gain elevated privileges, the network is effectively taken over and "owned" by the intruders.
- **Sustainment** In this phase, the hackers may secretly install malicious programs like root kits that allow them to return as frequently as they want. And with the elevated privileges that were acquired earlier, dependence on a single access point is no longer necessary. The attackers can come and go as they please.
- **Assault** This is when the hackers might alter the functionality of the victim's hardware, or disable the hardware entirely.
- **Obfuscation** The purpose of trail obfuscation is to divert the forensic examination process covering a variety of techniques and tools.

	LIH	MIH	HIH
Reconnaissance	✓	✓	✓
Scanning	✓	✓	✓
Access and escalation		*1	✓
Sustainment			✓
Assault			✓
Obfuscation			✓

Table 2.2: Level of attack recognition by the honeypot

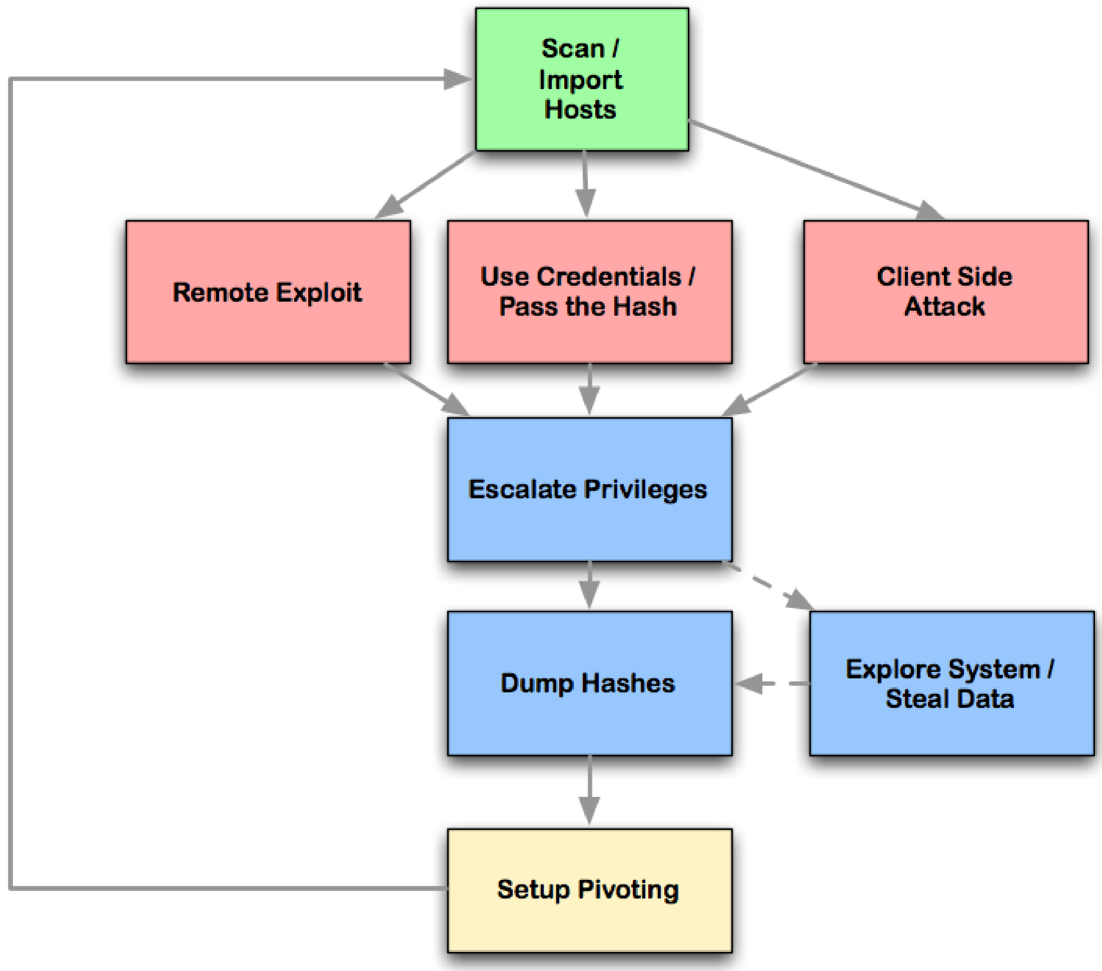


Figure 2.3: Hacking Process

2.3 Indicators of Compromise

Indicators of Compromise are forensic artifacts that are used as signs that a system has been compromised by and attack or that it has been infected with a particular malicious software [3].

When a malware attack takes place, traces of its activity can be left in system and log files. These IoCs present the activity on your network that you may not otherwise be able to see in real-time and that could suggest potentially malicious activity is taking place. If a security breach is identified, the IoC or "forensic data" is collected from these files[15]

It mostly give valuable information about an attack that has happened already but can also be used to prepare for the future and prevent against similar attacks. Antimalware software and similar security technologies use known indicators of compromise, such as a virus signature, to proactively guard against evasive threats.

According to Hutchins[7], there are 3 types of indicators:

- Atomic - Atomic indicators are those which cannot be broken down into smaller parts and retain their meaning in the context of an intrusion. Typical examples here are IP addresses, email addresses, and vulnerability identifiers.
- Computed - Computed indicators are those which are derived from data involved in an incident. Common computed indicators include hash values and regular expressions.
- Behavioral - Behavioral indicators are collections of computed and atomic indicators, often subject to qualification by quantity and possibly combinatorial logic. An example would be a statement such as ”the intruder would initially used a backdoor which generated network traffic matching regular expression at the rate of some frequency to some IP address, and then replace it with one matching the MD5 hash value once access was established.

2.4 Identifying malware

There has been a recent increase in the availability of intelligence (IOC's) related to malware, for example new IP addresses, hostnames, MD5 hashes, mutex values, and other attacker artifacts. There is many host-based and network-based standard methods to utilize these artifacts, such as intrusion detection/prevention systems, firewalls, anti-virus, and file whitelisting. While these solutions provide various benefits, they can fall short of confirming an infection. Malware can be identified by seeking out common TTP's (tools, techniques, and procedures) used during development and infection. These relationships help analyst gain a better understanding of the adversary and develop attacker profiles. From the profiles we can draw inferences to better adapt and respond to attacks. [15]

Is possible to identify malware and create attack patterns because humans aren't perfect. "It works because humans are creatures of habit and tend to use what they know. They use the same tools every day and don't rewrite their malware every morning. They put their digital footprints out on the Internet long ago – and it's usually just a few clicks away from discovery. There is a reflection of the threat actor behind every intrusion. To discount this is to discount forensic science." (Hoglund, 2011)

Identification of singular characteristics of malware helps to automate future identification tasks. Often malware from the same family share these characteristics, which helps analysts, identify related files. These characteristics can be used by analysts to organize malware within repositories, root out additional infections present on the network, or identify new variants of malware. [15]

Malware detection/identification can be split in two types:

- **Anomaly based detection** observe the traffic statistics and host behavior to detect previously unknown breaches. To detect malicious traffic it require to understand normal traffic behavior, which in turn require efficient training which in real time scenario is much difficult to achieve, as the behavior of legitimate activities are largely unpredictable.[10]
- **Signature based detection** does not take interest in propagation or transmission scheme; neither they look into host behavior. They look for specific byte sequence in each packet. If any match found with signature stored in database it will be identified as malicious.[10]

Anomaly based detection	Signature based detection
• Generates high false alarm	• uses the string algorithm, which can only generate static approach of detection
• Malicious activity that falls within normal usage patterns is not detected	• Do not work well against the multitude of attack patterns created by a human or a worm with self-modifying behavioral characteristics.
• Difficultly of defining rules	

Fortunately, is possible to combine signature based detection with anomaly based detection, and get the advantage of both schemes in automated signature generation, overcoming some of the negative aspects of each to improve a reliable detection.

2.5 Creating signatures from malware activity

A signature is used to detect one or multiple attacks, is the pattern inside the data packet that may be present in different parts of it depending upon the nature of the attack. For example, you can find signatures in the IP header, transport layer header (TCP or UDP header) and/or application layer header or payload. Tools like IDS depends upon signatures to find out about intruder activity.[9]

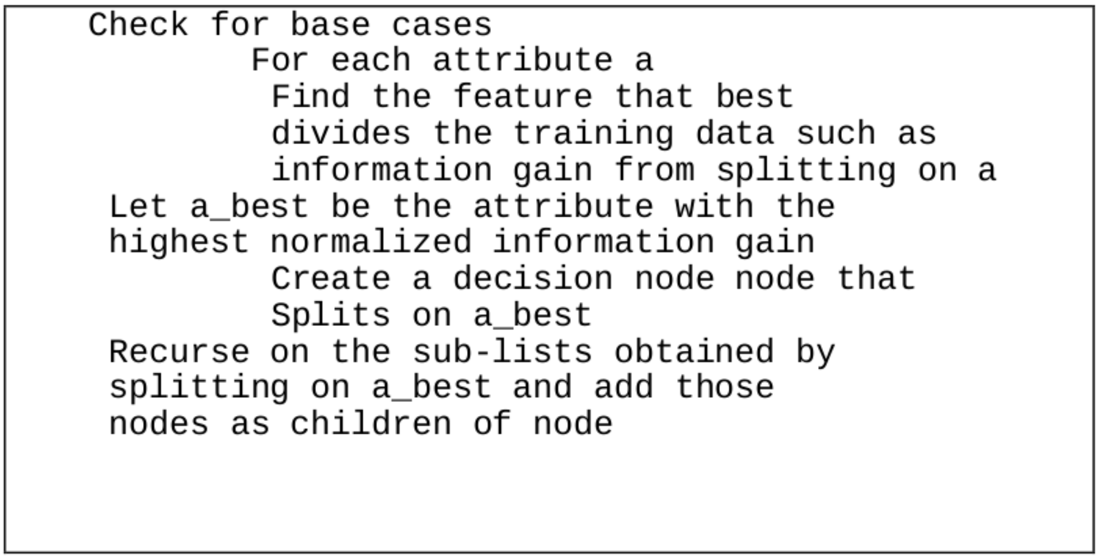
The goal is to describe the characteristic elements of attacks. A signature can be a portion of code, a pattern of behavior, a sequence of system calls, etc. A good signature must be narrow enough to capture precisely the characteristic aspects of exploit it attempts to address; at the same time, it should be flexible enough to capture variations of the attack. [9]

Content Based Signature Generation is process of extracting the attack signatures based on selection of the most frequently occurring byte sequences across the flows in the suspicious flow pool. [9]

Using the honeypot technology it's possible to isolate suspicious traffic from normal traffic to collect the information and forward it to generate automated signature of the attacking tools.[9]

2.6 Automated Signature generation

To automate the signature's creation for each hacking tool, a classification model will be applied to the isolated traffic gathered by the honeypot, aiming to extract singular identifying characteristics.



```
    Check for base cases
            For each attribute a
              Find the feature that best
              divides the training data such as
              information gain from splitting on a
      Let a_best be the attribute with the
      highest normalized information gain
              Create a decision node node that
              Splits on a_best
      Recurse on the sub-lists obtained by
      splitting on a_best and add those
      nodes as children of node
```

Figure 2.4: General Classification Model Algorithm[9]

2.6.1 C4.5 Algorithm

Because there is many hacking tools, it is difficult to know all the ways an attack can occur. C4.5 algorithm is going to be used because is well-known into classification problem. "The most well-know algorithm for building decision trees is the C4.5" [6]. C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. This is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurses on the smaller subsists [6].

2.6.2 IOC Clustering

Clustering is an unsupervised method which takes a different approach by grouping objects into meaningful subclasses so that members from the same cluster are quite similar and different to the members of different cluster. If we use clustering for intrusion detection then we have in anomaly detection model which is developed based on normal data and deviations are searched over this model. It is difficult to say that there are no attacks during the time traffic collected from the network.[13]

3 Architecture

Honeypots are planned and conveyed to gather intrusion datasets that help in measuring and understanding network dangers being a great tool for malware research and IDS's are great are detecting already found malware signatures. This chapter exlains the system used in this case and how the malware data is going to be extracted.

3.1 T-Pot Honeypot System

T-Pot is a honeypot system that is based on docker, which includes dockerized versions of the following honey pots :- conpot, cowire, dionaea, elasticpot, emobility, glastopf and honeytrap. T -Pot uses suricata a Network Security Monitoring engine and the ELK stack to beautifully visualize all the events captured by T-Pot. TPot is based on Ubuntu Server 14.04.4 LTS. The honeypot daemons as well as other support components being used have been paravirtualized using docker. This allowed the run of multiple honeypot daemons on the same network interface without problems and it make the entire system very low maintenance. The encapsulation of honeypot daemons in docker provides a good isolation of the runtime environments and easy update mechanisms. This honeypot system comprises of different detection engine and types of honeypot such as low, high interaction honeypot system. In this paper T-Pot will be used to collect intrusion packets in the network. T-Pot as the honeypot type for this project is chosen from other types of honeypot because its availability, easy installation.

3.2 Data Analysis

Malware data was collected from honeypot implemented in an isolated network. T-Pot system provides logfile monitor system which re-checks the logfile to filter out normal traffic that was recorded as an attack.

Base on the intrusion information gathered from T-Pot honeypot, the training dataset is analyzed , focusing on the behavior of intrusion data and network activities

3.3 Attack Analysis

Malware analysis is a procedure to perform analysis of malware and how to study the behavior of malware. Many works have been done in malware analysis as many uses static methods, which is a method of malware analysis done without running the malware. This paper work will use dynamic analysis as the method adopted to analyze the malware data. Dynamic analysis is a method of malware analysis which the malware is executed in a secure system, to enable the study of the malware life cycle and extract its behavior /features .

4 Analysis of Results

In this chapter, is presented a attack analysis of Armitage scan where is described what data is possible to gather in each component in the honeypot and the conclusions that are possible to be draw from it.

4.1 IDS vs. Honeypot

Now it's possible to confirm previous statements with the analyzed data. The attacks detected by the IDS are far less the ones detected by the honeypots, because not all the signatures are documented and the fact that normal traffic also triggers a notification in the IDS can make it's data not completely reliable.

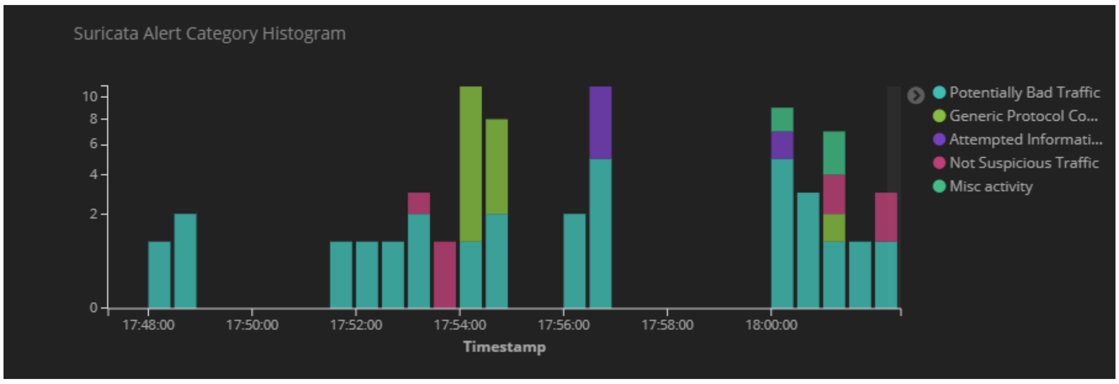


Figure 4.1: IDS Detection Graph

On other hand, we can see that because T-pot has a group of virtualized honeypots can overcame the handicap of less protocols emulated than a normal system because the different interaction levels honeypots can complete and improve each other.

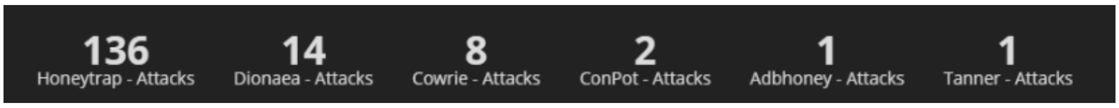


Figure 4.2: Honeypot Detection

Fortunately, is possible to join both and get the reliability of the honeypot's data while also completing the Indicators created with the IDS already knowed signatures.

4.2 Armitage

Armitage is a scriptable red team collaboration tool for Metasploit that visualizes targets, recommends exploits, and exposes the advanced post-exploitation features in the framework. Armitage organizes Metasploit's capabilities around the hacking process. There are features for discovery, access, post-exploitation, and maneuver.

Metasploit is a console driven application. Anything you do in Armitage is translated into a command Metasploit understands, therefore we can control more the parameters to try to identify the attack on the victim side.

In this study case, only the Scan function from Armitage will be taken into account. Which consists of a advanced NMAP scan followed by services version detection script for the ports found by nmap.

- **Honeypot Data**

  Dionaea Log Segment

```
{
    "dst_port":21,
    "src_ip":"10.8.25.108",
    "src_port":44533,
    "timestamp":"2019-01-20T18:00:14.530322",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"ftpd",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}
```

With Dionaea records we can gather that there's someone trying to compromise our honeypot and the fact that all of the services were interacted with and ,withing milliseconds, it's possible to affirm that the attacker it's performing a port scan and we are in the Reconnaissance/Scanning part of an attack.

Honeytrap Log Segment

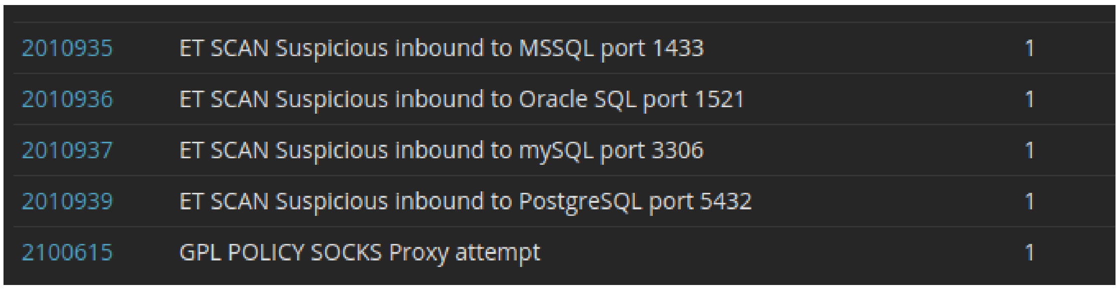
```
{
    tcp 10.8.25.108:36997 -> 10.8.25.202:10203
    d41d8cd98f00b204e9800998e
    cf8427ecf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d3
    ce9ce 47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af9
    27da3e (0 bytes)
}
```

Having the possibility to cross the logs from all the honeypots it's easy to realize that most of the packets have no data inside of them witch further proves that we are dealing in fact with port scanning because these protocols generally transmit data between them, therefore it must be a program just testing what ports and services are running and doesn't need to actually send more than a empty packet

- **IDS Data**

Suricata Log Segment

```
{
    "timestamp":"2019-01-20T17:56:37.606152+0000",
    "flow_id":561277975216072,
    "in_iface":"enp0s3",
    "event_type":"alert",
    "src_ip":"10.8.25.108",
    "src_port":54401,
        "dest_ip":"10.8.25.202",
        "dest_port":161,
        "proto":"TCP",
        "alert":{
            "action":"allowed",
            "gid":1,
            "signature_id":2101418,
            "rev":13,
            "signature":"GPL SNMP request tcp",
            "category":"Attempted Information Leak",
            "severity":2
        },
        "payload_printable":"",
        "stream":0
    }
```



2010935	ET SCAN Suspicious inbound to MSSQL port 1433	1
2010936	ET SCAN Suspicious inbound to Oracle SQL port 1521	1
2010937	ET SCAN Suspicious inbound to mySQL port 3306	1
2010939	ET SCAN Suspicious inbound to PostgreSQL port 5432	1
2100615	GPL POLICY SOCKS Proxy attempt	1

Figure 4.3: Suricata Detection

On the IDS side of things multiples flags are raised and it's possible to complement and justify that indeed there is a scan happening and multiple other suspicious interaction happen after it like HTTP interactions with big HTML files and many other interactions with all the services emulated and on ports where it's not usual to have those.

```
    "app_proto":"http",
    "fileinfo":{
        "filename":"\/",
        "magic":"HTML document, UTF-8 Unicode text, with very long lines",
        "gaps":false,
        "state":"CLOSED",
        "md5":"c9c65774fd2401c660a14616a73a1890",
        "stored":false ,
        "size":13383,
        "tx_id":0
    }
```

IDS Detected threats

- ET SCAN NMAP -sS window 1024
- GPL POLICY SOCKS Proxy attempt
- ET SCAN Suspicious inbound to MSSQL port 1433
- ET SCAN Suspicious inbound to mySQL port 3306
- ET POLICY External MYSQL Server Connection
- GPL POLICY VNC server response
- ET SCAN Suspicious inbound to PostgreSQL port 5432
- ET SCAN Potential VNC Scan 5900-5920
- ET VOIP Q.931 Call Setup - Inbound
- ET SCAN Suspicious inbound to mySQL port 3306
- ET POLICY HTTP Request on Unusual Port Possibly Hostile
- ET POLICY HTTP POST on unusual Port Possibly Hostile
- ET INFO GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1
- ET VOIP Q.931 Call Setup - Inbound

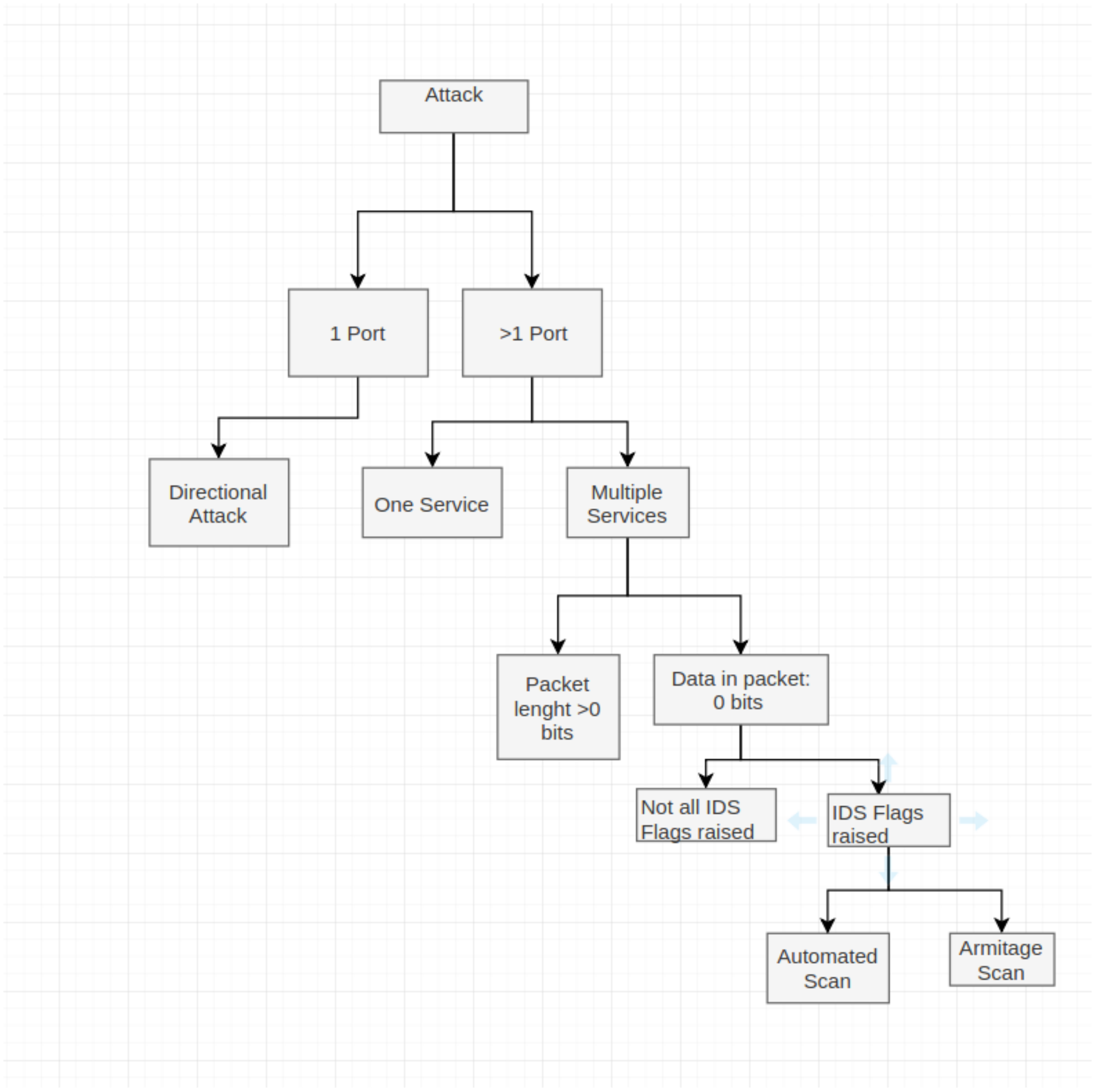


Figure 4.4: Proposed C4.5 aplication

4.3 C4.5 Aplication

From the data collected it's possible to detect that the following characteristics (Figure 4.4) can be the ones used to data split a classification tree to apply a automated process in the IOC creation.

This is a proposed tree, in future works can still be adapted and more hybrid so it can detect little variations so it handle some flags that can change the behaviour of the attack.

4.4 IOC Rule

With the previous indicators in mind it’s possible to conclude that is possible to cluster all of them in a indicator root node for Armitage and share it for the community with the purpose of adding more details to already discovered indicators.

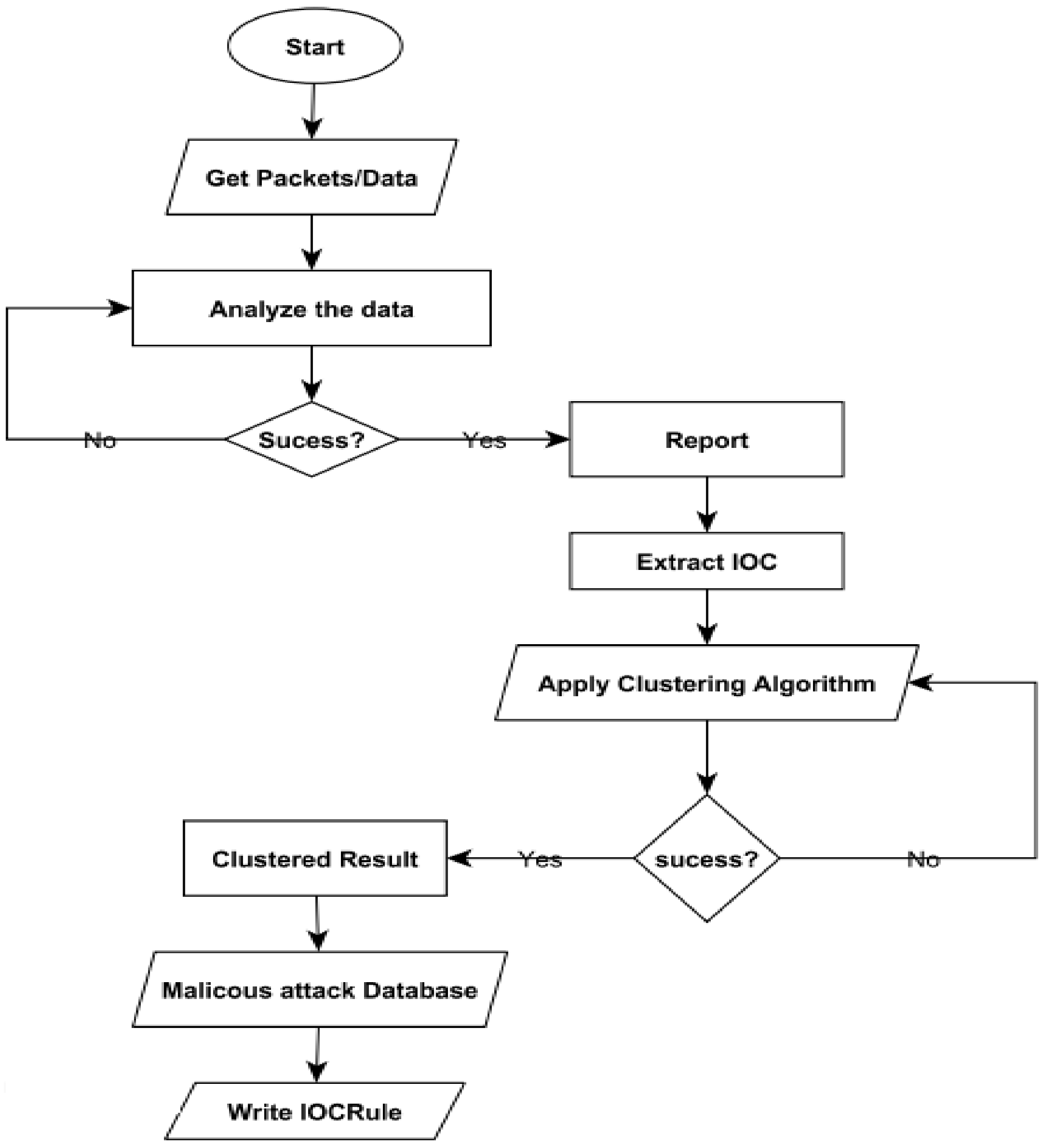


Figure 4.5: IOC Creation Algorithm [14]

5 Conclusion

In this paper, were examined how honeypots can be great tools for malware analysis. Where was shown how we can successfully gather valuable information about a incoming attack.

The system implemented could successfully gather the activity of a attacking tool in the network and all the interactions. And the method proposed can help the construction of more detailed Indicators of Compromise.

If applied to more hacking tools with a more extensive research can be possible to gather even more information and map the usage of a lot more applications and with that information, security defenders can infer more about the attacker enabling a wider peak into to the attacker world.

The creation of strong attacker profiles will allow to the defender to have a better protection because is possible to know who are they dealing with and make more assumptions on the reason the machine is being attacked.

Bibliography

[1] *Threat landscape report q2 2017.* Fortinet, 2017

[2] Ashoor, Asmaa S.: *Importance of Intrusion Detection System (IDS).* International Journal of Scientific and Engineering Research, 2011. – URL https://www.ijser.org/researchpaper/Importance_of_Intrusion_Detection_System.pdf

[3] Catakoglu, Onur ; Balduzzi, Marco ; Balzarotti, Davide: *Automatic Extraction of Indicators of Compromise for Web Applications.* WWW '16 Proceedings of the 25th International Conference on World Wide Web, 2016. – URL https://dl.acm.org/citation.cfm?id=2883056

[4] Cohen, Fred: *The Deception ToolKit.* The Risks Digest, 1998. – URL http://catless.ncl.ac.uk/Risks/19.62.html#subj11

[5] Fan, Wenjun ; Du, Zhihui ; Fernández, David ; Villagrá, Víctor A.: *Enabling an Anatomic View to Investigate Honeypot System: A survey.* IEEE, 2017. – URL https://ieeexplore.ieee.org/document/8098608

[6] Gusfield, D.: *Algorithms on Strings, Trees and Sequences.* Cambridge University Press: Cambridge, 1997

[7] Hutchins, Eric M. ; Cloppert, Michael J. ; Amin, Rohan M.: *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains.* Lockheed Martin Corporation, 2011. – URL https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

[8] Kreibich, Christian ; Crowcroft, Jon: *Honeycomb – Creating Intrusion Detection Signatures Using Honeypots.* 2004

[9] Mohssena, Eisa A. ; Z., M. ; V, Neco: *An automated signature generation method for zero-day polymorphic worms based on c4.5 algorithm.* The Ninth International Conference on Software Engineering Advances, 2014

[10] Paul, Sounak ; Mishra, Bimal K.: *Honeypot-based Signature Generation for Polymorphic Worms.* International Journal of Security and Its Applications, 2014

[11] Peter, Eric: *A Practical Guide to Honeypots.* URL https://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/, 2008

[12] Provos, Niels: *A Virtual Honeypot Framework.* Google, 2004. – URL http://www.citi.umich.edu/techreports/reports/citi-tr-03-1.pdf

[13] Ranjan, Ravi ; Sahoo, G.: *A NEW CLUSTERING APPROACH FOR ANOMALY INTRUSION DETECTION.* International Journal of Data Mining and Knowledge Management Process, 2014

[14] Rene, Chiadighikaobi I. ; Abdullah, Johari: *Malicious Code Intrusion Detection using Machine Learning And Indicators of Compromise.* International Journal of Computer Science and Information Security, 2017

[15] Robertson, Chad: *Indicators of compromise in memory forensics.* SANS, 2013. – URL https://www.sans.org/reading-room/whitepapers/forensics/indicators-compromise-memory-forensics-34162

[16] Spitzner, Lance: *Honeypots: Tracking Hackers.* 2002

[17] Spitzner, Lance: *Honeypots: catching the insider threat.* IEEE, 2003. – URL https://ieeexplore.ieee.org/abstract/document/1254322

[18] Spitzner, Lance: *Honeytokens: The Other Honeypot.* Security Focus, 2003. – URL http://www.securityfocus.com/infocus/1713

[19] Wagner, David ; Soto, Paolo: *Mimicry Attacks on Host-Based Intrusion Detection Systems.* URL https://dl.acm.org/citation.cfm?id=586145, 2002

[20] Zhang, Feng ; Zhou, Shijie ; Qin, Zhiguang ; Liu, Jinde: *Honeypot: a Supplemented Active Defense System for Network Security.* IEEE, 2003. – URL https://ieeexplore.ieee.org/document/1236295

A Appendix

A.1 Dionaea Log

```
{
   "dst_port":21,
   "src_ip":"10.8.25.108",
   "src_port":44533,
   "timestamp":"2019-01-20T18:00:14.530322",
   "src_hostname":"",
   "connection":{
      "type":"accept",
      "protocol":"ftpd",
      "transport":"tcp"
   },
   "dst_ip":"10.8.25.202"
}{
   "dst_port":36209,
   "src_ip":"10.8.25.108",
   "src_port":42,
   "timestamp":"2019-01-20T18:00:14.490444",
   "src_hostname":"",
   "connection":{
      "type":"connect",
      "protocol":"mirrorc",
      "transport":"tcp"
   },
   "dst_ip":"10.8.25.202"
}{
   "dst_port":42,
    "src_ip":"10.8.25.108",
    "src_port":44733,
    "timestamp":"2019-01-20T18:00:14.519010",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"mirrord",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":81,
    "src_ip":"10.8.25.108",
    "src_port":33073,
    "timestamp":"2019-01-20T18:00:14.540448",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"httpd",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":135,
    "src_ip":"10.8.25.108",
    "src_port":42433,
    "timestamp":"2019-01-20T18:00:15.274642",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"epmapper",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":445,
    "src_ip":"10.8.25.108",
    "src_port":39123,
    "timestamp":"2019-01-20T18:00:15.634374",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"smbd",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":1723,
    "src_ip":"10.8.25.108",
    "src_port":41767,
    "timestamp":"2019-01-20T18:00:16.074005",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"pptpd",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":1433,
    "src_ip":"10.8.25.108",
    "src_port":38689,
    "timestamp":"2019-01-20T18:00:16.085206",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"mssqld",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":3306,
    "src_ip":"10.8.25.108",
    "src_port":39815,
    "timestamp":"2019-01-20T18:00:17.113112",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"mysqld",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":5060,
    "src_ip":"10.8.25.108",
    "src_port":44377,
    "timestamp":"2019-01-20T18:00:17.663259",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"SipSession",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":21,
    "src_ip":"10.8.25.108",
    "src_port":41533,
    "timestamp":"2019-01-20T18:00:39.033336",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"ftpd",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":445,
    "src_ip":"10.8.25.108",
    "src_port":45883,
    "timestamp":"2019-01-20T18:01:03.208905",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"smbd",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":443,
    "src_ip":"10.8.25.108",
    "src_port":41111,
    "timestamp":"2019-01-20T18:01:07.958363",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"httpd",
        "transport":"tls"
    },
    "dst_ip":"10.8.25.202"
}{
    "dst_port":3306,
    "src_ip":"10.8.25.108",
    "src_port":46785,
    "timestamp":"2019-01-20T18:01:26.345952",
    "src_hostname":"",
    "connection":{
        "type":"accept",
        "protocol":"mysqld",
        "transport":"tcp"
    },
    "dst_ip":"10.8.25.202"
}
```

A.2 Honetrap Log

```
tcp 10.8.25.108:38175 -> 10.8.25.202:1582 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:45439 -> 10.8.25.202:1900 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:40707 -> 10.8.25.202:1720 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:33939 -> 10.8.25.202:1581 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:41091 -> 10.8.25.202:1755 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:41491 -> 10.8.25.202:1521 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:36507 -> 10.8.25.202:1533 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:41511 -> 10.8.25.202:1434 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:44357 -> 10.8.25.202:2222 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
tcp 10.8.25.108:44841 -> 10.8.25.202:2947 d41d8cd98f00b204e9800998ecf8427e
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d
85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e (0 bytes)
```

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: ________________________________

Vorname: ________________________________

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Automated Identification of Attacking Tools in a Honeypot

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

__________	__________	________________
Ort	Datum	Unterschrift im Original