TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

TUT Centre for Digital Forensics and Cyber Security

ITC70LT

Onur Aydin Korkmaz, Student No: a122792

COMPREHENSIVE ANALYSIS OF CYBER ATTACKS AND MALWARE USING LOW- AND HIGH-INTERACTION HONEYPOT

MADAL- JA KÕRGINTERAKTIIVSETE MEEPURKIDE KASUTAMINE KÜBERRÜNNAKUTE JA PAHAVARA KOMPLEKSANALÜÜSIKS

Master Thesis

Truls T. Ringkjob

Master's Degree in Cyber Security

Lecturer IT college, Visiting Lecturer TTÜ

Tallinn 2016

Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Onur Aydin Korkmaz

25.05.2016

Abstract

Today's cyber threats are sophisticated, well-funded, organized and cybercrime operations can cause major loss to organizations, national infrastructure. Malware is the major actor as threat to the security of networks and cause crucial security risks to the systems. There are solid tools to protect systems that mitigate the risks but mostly about against known or identified vulnerabilities. Honeypot is a decoy tool; an approach to observe attackers and uncover some of the threats.

In this thesis, the role and concept of honeypot with its advantage and disadvantage in security was studied. A low-interaction honeypot was used to collect malware and data. Then real services were installed, local and virtual network systems were created, virtual hosts were implemented on two more IP addresses that made honeypot high-interaction. A practical analysis of attacks and exploited vulnerabilities was made to understand attack patterns, attacker's behaviors and methods. The collected data was compared between low interaction and high interaction honeypot. The analysis offers a model to set up a system to collect efficiently malware samples. This system runs a low-interaction honeypot that listens as many IP in the network and minimum number of secured real services. The real services helped to receive more connections but not for collecting more malware samples. In order to lure attackers, running real systems with honeypot was risky and unfortunately the system was compromised and host based forensics was done.

This thesis is written in English and is 62 pages long, including 7 chapters, 10 figures and 11 tables.

Annotatsioon

Tänasel päeval on küber ohud väga aktuaalne teema. Küberrünnakud võivad tekitada firmadele väga suurt kahju - nii rahaliselt kui ka maine poolest. Pahavara on üks suuremaid ohu faktoreid võrgu turvalisuses ja see tekitab suuri turvalisuse ohte süsteemi. Süsteemi kaitsemiseks on olemas tarkvara, mis aitab ohte vähendada. Meepott on tarkvara, mis aitab anda ülevaate ründajatest ning leida üles ohud.

Lõputöös kasutati madal- ja kõrginteraktiivset meepotti. Analüüsiti kogutud andmeid ja pakutakse välja mudel, et efektiiselt koguda pahavara näiteid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 62 leheküljl, 7 peatükki, 10 joonist, 11 tabelit.

Table of abbreviations and terms

IDS	Intrusion detection system
IPS	Intrusion prevention system
LAMP	Linux, Apache, MySQL, PHP
DMZ	Demilitarized zone
VM	Virtual machine
SMB	Server Message Block
DDoS	Distributed Denial of Service
SQL	Structured Query Language
SSH	Secure Shell
HVM	Hardware virtual machine
PV	Para virtualization
SA	System administrator
TLS	Transport Layer Security
MSRPC	Microsoft Remote Procedure Call
XMPP	Extensible Messaging and Presence Protocol
LTS	Long term supported
LAN	Local Area Network
ISP	Internet Service Provider
MD5	Message-digest Algorithm
NLP	Natural Language Processing

T-SNE t-Distributed Stochastic Neighbor Embedding

LSI Latent Semantics Indexing

ERM Entity relationship model

DCE/RPC Distributed Computing Environment / Remote Procedure Calls

SRVSVC the Server Service Remote Protocol

DSSETUP Active Directory Services Setup

PNP Plug and Play

DCOM Distributed Component Object Model

RPCSS Remote Procedure Call System Service

ICMP Internet Control Message Protocol

SMTP Simple Mail Transfer Protocol

Acknowledgments

I would like to thank my supervisor Truls T. Ringkjob to have it made possible for me to work on my master thesis and his understanding and availability as well as his thoughtful guidance, supplying necessary equipment.

I also thank to all my teachers of my master programme who efficiently prepared me for this master's thesis.

Table of Contents

1. Introduction 10
   1.1. Background and Motivation 10
   1.2. Scope of study 11
   1.3. Research goals 12
   1.4. Contribution of thesis 12
2. Honeypots 14
   2.1. History of honeypot 15
   2.2. Types of honeypots 15
   2.3. Honeypot deployment on a network 16
   2.4. Value of honeypot in security 17
   2.5. Drawbacks and shortcoming of honeypot 21
3. Low-interaction honeypots and design 23
   3.1. Low-Interaction honeypots 23
   3.2. Dionaea 24
      3.2.1. Protocols 24
      3.2.2. Modules 25
      3.2.3. Configuration file 26
      3.2.4. Utils 26
      3.2.5. Privileges 26
   3.3. Design 26
4. Low to High Interaction 30
      4.1.1. Server installation 30

4.1.2. Tapping 30
4.1.3. Honeyd 31
4.2. Security 33
4.2.1. Detection of VMs 33
4.2.2. Detection of honeypots 33
4.3. Avoiding Dionaea service identification 34
4.4. Iptables 35
5. Analysis of data and attackers' behavior 38
5.1. Value of data 38
5.2. Machine1 39
5.3. Machine2 44
5.4. Machine3 45
5.5. Collected malwares 48
5.5.1. The purpose of malwares that targeted the system 49
6. Forensics of honeypot 52
6.1. System Description 52
6.2. Login and System Logs 53
6.3. Network logs 54
7. Summarizing the research 55
7.1. Conclusion 55
References 57
Appendix 61
1. List of SQL queries 61

List of Figures

Figure 1. Honeypot deployment in network 17

Figure 2. Honeypot taxonomy based on honeypots' data 19

Figure 3. Taxonomy of information protection mechanism 19

Figure 4. TCP/UDP port activity 25

Figure 5. Honeypot placement in network 27

Figure 6. Machine3 Network Schema 31

Figure 7. Virtual network schema 32

Figure 8. Entity relationship model for decision tree 39

Figure 9. Machine1 connections graphic 41

Figure 10. Machine3 connections graphic 46

List of Tables

Table 1. Defensible actions Matrix 20

Table 2. Machine1 statistical results 40

Table 3. Machine1 SQL login info 41

Table 4. Machine1 List of exploited vulnerabilities 42

Table 5. Machine2 statistical results 44

Table 6. Machine2 malware/connections table 44

Table 7. Machine2 SQL login info 44

Table 8. Machine2 List of exploited vulnerabilities 45

Table 9. Machine3 statistical data 46

Table 10. Machine3 SQL login info 47

Table 11. Machine3 Exploited Vulnerabilities 47

1. Introduction

Since the first PC virus appeared, hacking is industrialized, distinct threat cycles appeared and it is challenging to detect, understand and stop. In order to protect a system, there are security measures to be taken but can stop some part of them. The system misconfiguration, poor patch management practices implemented by system and network administrators, code, design, not updated software, unsecure network and especially the user can cause vulnerability. The human is the weakest part of chain in security that the most common cyber threats targeting users are caused by malware. In 2015, there were 230,000 new types of malware each day and 21 million new threats in second quarter according to PandaLabs report [1]. The cyber criminals use existing part of code in order to implement new types of malware that allows them quickly develop signature of new dangerous software and create zero-day exploits. The attackers use a set of malware repeatedly in different attacks and past methods to try to exploit any vulnerability. The attackers prey on human weakness in order to lure insiders to inadvertently provide them access. Honeypot has similar aim; it lures attackers but interact with them collects as much as information about attacker and attacks [2] while keeping them away from critical systems. Honeypot try to trap attacker with emulated services that looks like real systems or with real services that is useful tool together with other security implementations. Honeypot detects malwares and known/unknown attacks (zero-day attacks), observes network to prevent new infections in early stage and allows analyzing attacks, attacker's behavior, tactics and methods. What makes honeypot interesting to study is to have these advantages over other security tools, it focuses on malware and unknown vulnerabilities in system caused by human and it collects all the information in database to analyze. It can make us to hold upper hand to fool attacker and strength the security with the gathered information. Honeypot may introduce risk to system; it is possible attacker can detect honeypot and honeypot can be compromised as anything coded by humans can be compromised.

1.1. Background and Motivation

The topic of the thesis was proposed to me by my supervisor Truls Ringkjob. It is an exciting topic related to security and made me look into working on collected large data,

finding a new malware, observing behavior of attackers and attack patterns. It was first time I studied on honeypots and improving security of a server and running a server.

Having a secured system, waiting to be attacked and hoping nothing will happen is a mostly common attitude. The system will be compromised when we are just about careless about security. Anti-virus software, firewalls and IDSs are important tools for security but greatly limited due to vast number of false positive (misclassified) and false negative (non-detected) and can create false sense of security [3]. Honeypot lures the attackers, logs activities in. This advantage of honeypot motivated me to set up a honeypot to analyze the collected data, the attacker's behavior and methods, collected malware and their purposes so that I could understand and observe what's happening in the system and learn how intruders attempt to gain access to a system.

I attended 2014 Honeynet Project Workshop in Warsaw, Poland from 12-14 May 2014, [4] and met specialists. It was all over great experience.

1.2. Scope of study

The study focuses on attacker, collecting malware, security models, and previous works on honeypots and connecting events from attacks to understand behavior and methods of attacker. It was researched what is the effect of low- and high-interaction honeypots in security among other tools with its advantages and disadvantages, models and risks of using honeypot. To collect malware and information about network activity, low-interaction honeypot is used. Real services were installed and virtual hosts were implemented; a local and a virtual network system are created that seems there are many computers in network. These real services made honeypot high interaction. It was necessary to filter network traffic; basic firewall rules are written and network commands are implemented. The differences between low interaction and high interaction of honeypots about their collected data and effect about collecting number of malware are studied. The useful information can be extracted from each one's collected data to improve security of the system. Also 2 more IPs used by tapping to increase collected data. Scope of study includes forensics of the compromised server since attacker took control of the honeypot that is one of biggest drawback of the honeypot and undesired event.

1.3. Research goals

The aim of this research is to develop a system that gathers malwares through a honeypot and if the honeypot catches a new malware or file, analyze it and understand what its purpose. Real systems and additional honeypot (Honeyd) were implemented to lure attacker (the attacker might think to intrude a LAN or more than one server) and feint attacker about operating system, services, network and security of the system.

This system gives useful information about level of attacker, whether attacks were organized or type of attacks happened, offered URLs to understand what attackers are looking for most or if the system is secured enough or what should be done additional for security.

The research goals;

- Comparison of low and high interaction honeypot for collecting malware; when few real services ran among honeypot.
- Efficiency of honeypot to improve security of a system.
- Making connections from collected data and analyzing attacker's behavior, methods, attack patterns and purpose of malwares
- How attacker compromised the system and what attacker did on it

1.4. Contribution of thesis

The most valuable contribution is both running low and high interaction honeypot in order to learn more about attacker and how honeypot can be useful or harmful for security.

- The role of honeypot as a decoy system in security; the previous works were researched about honeypot how to implement honeypot in system according the goal which to achieve. It has advantage over other security tools which should be used among them since it is not a good security tool alone. Honeypot is a deception tool in security models; it can be used as early warning system since any interaction to honeypot is suspicious and it makes system more difficult to understand for attacker with selected services to run. All these benefits bring risks to system or systems in network.

- Detailed comparison of high and low interaction honeypots through analyzing data; although high interaction honeypot receives more interests, the attacks shifts to different types, there are more aggressive attacks and honeypot collects less malware samples. Only using more IP addresses in network helped to collect more malware samples.
- The collected data by honeypots gave useful information about attack patterns, types and attacker's behavior (by analyzing and using models for malware, offers, links, connections etc.) This data can be useful to implement new IDS rules, to know vulnerabilities which attackers are interested in system, to learn whether attacks are organized and which attack models are used etc. with advanced SQL queries
- Why and how the server was compromised while running honeypot. Since it is host based forensics, there were not much information about whole compromising process.

2. Honeypots

It is a decoy system or fake server that intended to make it appears to attacker as if the system running known vulnerabilities. It is in an isolated environment and open to attacks so any interaction to it is malicious probably done by intruder, attacker. It helps to detect and identify the target and methods of exploiting by collected information [5]. It buys time to system administrator to secure system, also helpful to network administrator to make observation, analyze attacks from captured data and malwares, discover security holes and gives information about network threats.

It is easy to deploy but there are many different configurations/designs, depends on what is desired. Honeypot does not have a single task or a role. Honeypots can achieve tarpitting, detection, countering spam, information gathering, etc. It can be designed for different goals for example diverting attacks and keep attackers stay on the system long enough to collect information about attacker's activity [6]. Honeypot becomes perfect tool together conjunction with firewall and IDSs [7]; honeypot can detect attack that bypassed a firewall and IDS. With a honeypot any process can be detected, like encrypted data that IDSs cannot do that [8]. If attacker does not aware of honeypot, virtualized systems and networks; honeypot collect attacker's data and give time to security implementers and possibility to patch vulnerabilities. It is possible attacker can detect honeypot and honeypot can be compromised as anything coded by humans can be compromised.

There are new security models and malware detection systems are implemented against attackers. Honeypots do not replace other traditional security systems, they are an additional tool. In a security model, multi-level security controls are needed. Denying unauthorized access and isolate system (IDS and firewall) is not enough, attacker can succeed in penetrating. Degradation and obfuscation mechanism (encryption, stenography, anti-fingerprinting) can slow attacker. As a deception mechanism honeypot can be used with these security controls to collect attacker's behavior and put attacker in a risk. Honeypots have drawbacks; it is important to make design and operate honeypot in order to maximize their effectiveness.

2.1. History of honeypot

It is not clear how the term honeypot was founded; the concept of honeypot was first described by Clifford Stoll in 1989 after a hacker had unauthorized access into computer in Lawrence Berkeley National Laboratory. In his book Cuckoo's Egg, he wrote "Detecting the hacker was easy; I'd just camp out in my office alongside two terminals. One terminal for working, another to watch the system…" [9]. The hacker was captured by a physical honeypot. In 1991 Bill Cheswick published his experiences in "An evening with Berferd" [10] that he developed fake services, password files and wrote a script to pull fake service activity from the logs. He did a honeypot almost looked like today's honeypots. Both of them dealt with advanced attackers. Honeypot systems were researched and deployed within military, government and commercial organizations before 90s but very little of it was public knowledge [11]. In 1997 Dr. Fred Cohen developed Deception Tool Kit [12] that was designed and coded executables to respond hacker probes as they are vulnerable systems. It was first honeypot solutions available to the security community. In 1999, a non profit research group of security professionals under Honeypot Project that Lance Spitzner founded started to research black hat community and shared what they learned. In 2001 one community released all research methods in series of paper known as "Know your enemy" [13] that helped develop awareness, credibility and value of honeypots. Following next year honeypots started to capture new, unknown attacks. On the other side black hat community released an article named "Local Honeypot Identification" in Phrack magazine during 2003 to detect honeypots or defeat honeypots [14]. In fact both side faced similar challenges; both want to monitor activities conducted on the systems they control.

2.2. Types of honeypots

Classification according to the purpose;

- Research honeypots: the primary usage is to learn attacker's motive, methods, tools and collect malware. Any attempt to create new security measures through research allow to face new threats [15]. It is primarily used by companies or corporations.

- Production honeypots; the primary usage is to increase security [16]. It is typically deployed with a certain goal or intent in mind. They are easy to use but capture only limited information [17].

Classification according to level of interaction;

- Low-interaction honeypots; it allows limited interaction for an attacker or malware that reduce risk to a minimum. It can be compared to passive IDS. All services offered are emulated.
- High-interaction honeypots; it provides complex interactions with attackers by incorporating actual operating system and services. It captures large amount of information about attacker. Since it uses actual operating system, when it is compromised it can be used as base to launch attacks in whole network [18].
- Hybrid honeypots; Low-interaction honeypots are not powerful and high-interaction honeypots are too expensive, it offers benefit of both by using their own specialty [16].

Classification according to type of exploitable resources

- Server-side honeypots; the honeypot acts as a server. There are many specialized types of honeypots to detect attacks on web applications, SSH, industrial control systems, VoIP, bluetooth, USB etc.
- Client-side; Honeypot acts as a client application [19].

2.3. Honeypot deployment on a network

Honeypot can be deployed in a variety of locations on a network; it depends on goal. Conceptually honeypots can be placed at three main locations; external, in the DMZ and internal.

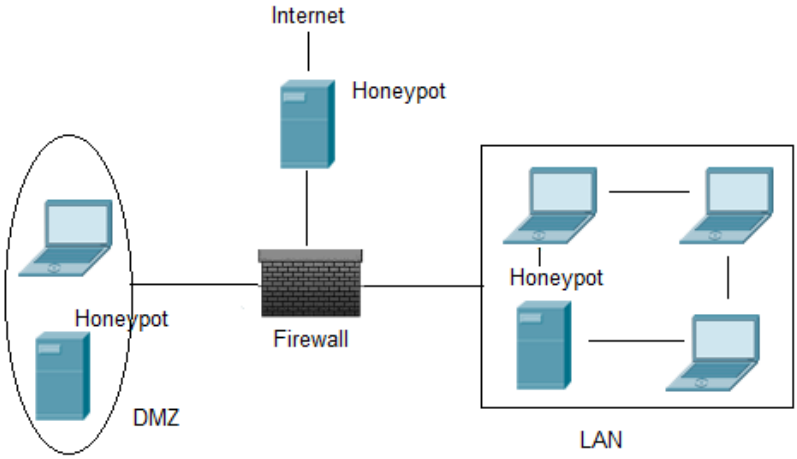


Figure 1 – Honeypot deployment in network

Honeypot deployed externally detects all activities, zero-day vulnerabilities and collect large amount of malware. The risk for internal network doesn't increase. It reduces amount of alerts produced by external firewall. By placing honeypot external doesn't let locate or trap internal attackers easy [20] because firewall limit out-bound traffic. DMZs are not fully accessible as only needed services are allowed to pass the firewall. Honeypot deployed in the DMZ is appropriate for detection intruders. It can collect information, malware and external attacks to the services allowed by the DMZ's firewall. It is most complex placement model. Opening all corresponding ports on the firewall is time consuming and risky [21]. A router can be placed between the DMZ's firewall to add a layer for data control. It is not the best early-warning indicator for internal network but can detect unauthorized actions from internal network [22]. Honeypot deployed inside of network provides semantic-rich view on various aspects of system dynamics and it creates early-warning system but it has administrative overload [23]. A production honeypot can be placed to internal network.

2.4. Value of honeypot in security

Honeypots can be implemented in different ways depends on the aim to achieve in security. The honeypot goal is to capture as much as information about activity of attacker. Honeypots serve protocols, modules, virtual operating systems; can be anything vulnerable to lure attacker.

Any interaction to honeypot is suspicious. It keeps valuable information about malicious activity; it collects smaller, higher-value and datasets since it only logs illegitimate

activity. It is an extra layer of protection when placed internal. It shields servers from direct attacks, if decoy compromised, the real server is safe. It detects, deflects and prevents security breaches. When deployed external, it overcomes the resources problem; it captures activities directed to; when it is deployed same network, system is not overwhelmed by the traffic and any old computer can monitor millions of IP address. Honeypots work in any IP environment, are adaptable in variety of environment; can be anything to be simulated.

It can detect and respond to unknown or zero-day attacks and it works in encrypted environments. It gives possibilities to discover new forms of attack. Comparing to other security tools, honeypots do not require known attack signatures, unlike IDS which have better coverage of attacks, honeypot gets less false positives; no big volumes (less but more valuable data is collected), it distinguishes benign and hostile traffic. [24]. This makes honeypot an early warning system; honeypot can lure attacker and detects attacks that IDS/IPS systems missed and IDS/IPS systems can redirect attacker to honeypot to keep productions resources safe. It will alert hostile activity in network. This way it buys time to SA to secure system. Honeypot shares some values with other tools; *"Sink Holes are the network equivalent of a honey pot."* [25]. Unlikely firewall, it allows connection but not out.

Honeypot best practices are; distraction (it keeps attackers from real systems), deception (it has false/emulated services, data, files etc.), psychological operations (disinformation -providing attacker false information) and perception (presenting false capabilities that gives different perception of activity in organization), intelligence gathering (hacker tools, method, attack signature). [26]

Honeypot taxonomy

All the pieces of data honeypot collects can be gathered as a classified scheme in a relationship through cyber security incidents. Based on Sandia's taxonomy [27], it is modified related to honeypot based on data, the honeypot taxonomy in figure 2;

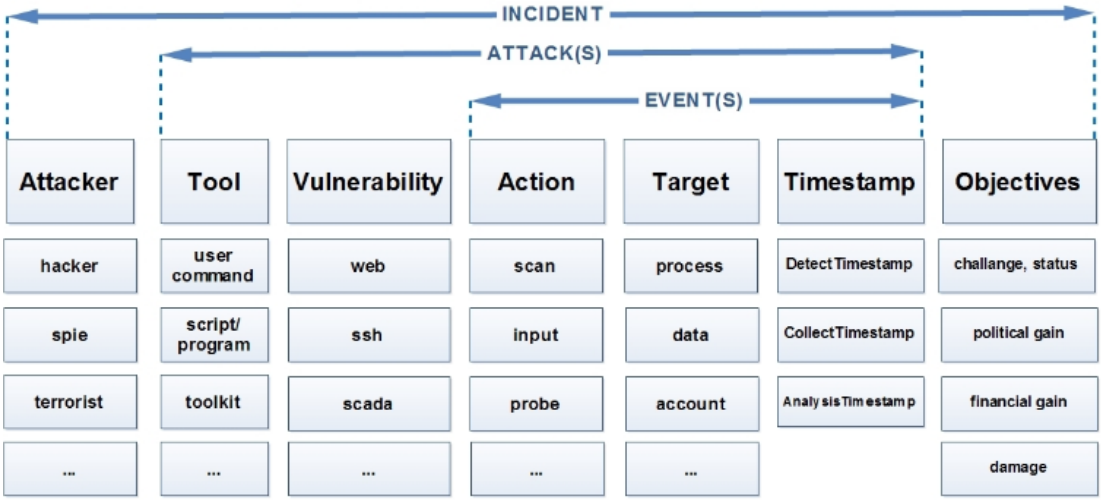


Figure 2 - Honeypot taxonomy based on honeypots' data [28]

Honeypot is the Negative Information and Deception Mechanism in taxonomy of information protection mechanisms (figure 3), especially for detecting abnormal user who tries to access the information where IDS can't detect [29].

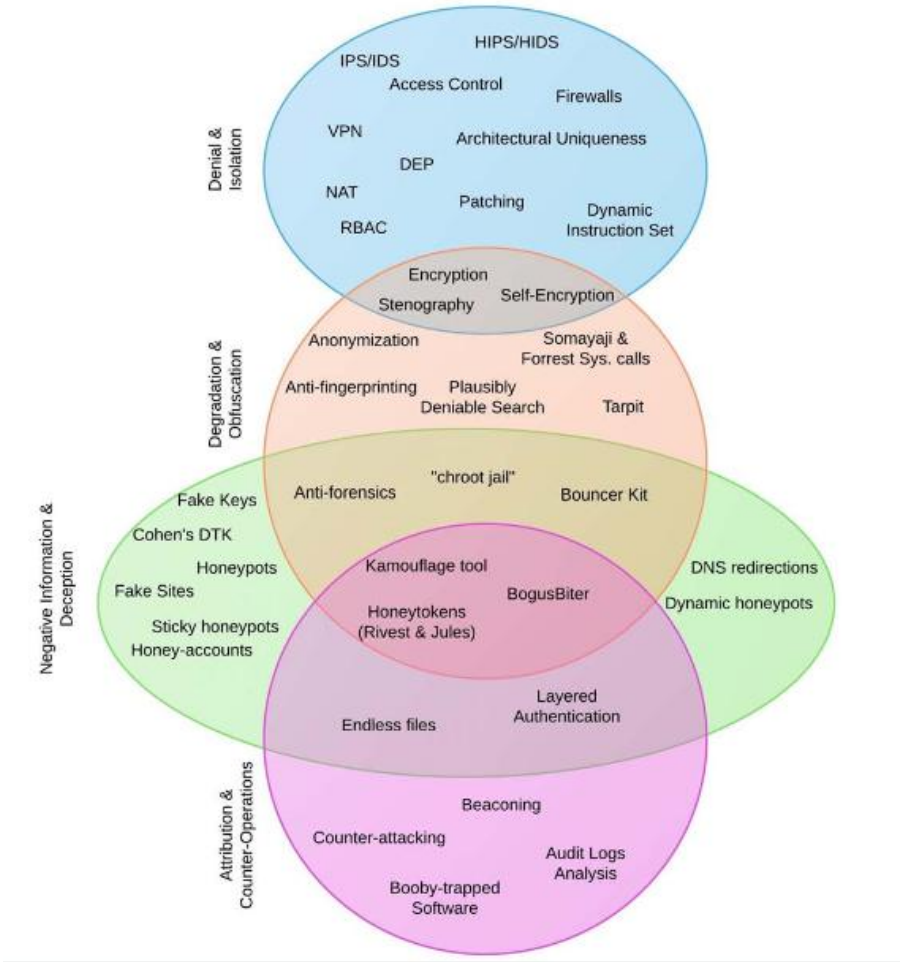


Figure 3 - Taxonomy of information protection mechanisms [29]

Honeypot as defensive deception

Auguste Kerckhoffs in the 19th century stated that a cryptosystem should be secure even if everything about the system, except the key, is public knowledge. It simply emphasizes that defences should not rely on only one dimension of security [30]. As long as deception stays undetected, deception makes attack complexity even greater and turn complexity advantage for defender. Developing good deception techniques benefit from good data on attack patterns. In defensible or action matrix of cyber kill chain, honeypot is used for this purpose in a phase. Kill Chain is an intelligence-driven and intrusion-centric security model introduced by Lockhead Martin Corporation. It is a phase-based model to describe the stages of an attack but also helps inform ways to prevent such attacks. It aims to stop the attack at earlier phase so that the attacker has less information, less likely someone use that information to complete attack later because attacker has to complete all phases successfully. This model gives information about attacker, how system is safe and idea to enhance for secure system.

Whenever a specific attacker has progressed beyond a certain point in the cyber kill chain, administrator can easily disconnect and wrap up a honeypot, terminate compromised image and it can be used for forensics or analysis or redirect malicious outbound communication to internal honeypots to identify compromised hosts.

Phase	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Web analytics	Firewall ACL				
Weaponization	NIDS	NIPS				
Delivery	Vigilant user	Proxy filter	In-line AV	Queuing		
Exploitation	HIDS	Patch	DEP			
Installation	HIDS	“chroot” jail	AV			
C2	NIDS	Firewall ACL	NIPS	Tarpit	DNS redirect	
Actions on Objectives	Audit log			Quality of Service	Honeypot	

Table 1 - Defensible actions Matrix

Equally as important as thorough analysis of successful compromises is synthesis of unsuccessful intrusions. If defenders implement countermeasures faster than their known adversaries evolve, they maintain a tactical advantage.

2.5. Drawbacks and shortcoming of honeypot

The honeypots are worthless if no one attacks them and they introduce risks. It can lure attackers to network and cause them to launch further attacks [31]. Depends on goal, if it is placed internal or with preliminary firewall, it is required to adjust firewall. It is one of the biggest drawbacks that made honeypot never reached its expected success, it brings administrative overhead by adding complexity to network that requires close and strict management, resources [32] and analyst overload from collected data. Cloud-based security solutions solve zero-day attacks problem with low cost in this way. Honeypot alone is not proper security tool; there is a need multiple detection methodologies, not just a honeypot for security [33].

Honeypot has legal ethical concerns; entrapment, privacy, liability. Honeypot lure attacker to provoke crimes. It may not be entrapment if is someone is induced to do something they would not normally do and bad guy has already decided to commit unauthorized activity. Honeypot captures data about an attacker; transactional and content information. Low interaction honeypots collects transactional information like IP address, time and date of communication. There are many different legal statues which privacy statues will apply is unsolved. It is not required to get consent of attacker with a banner. Honeypot can be a liability if it is compromised to take control of other systems, resources. Honeypots are no different than firewalls, IDS with vulnerability [34].

Detection Risk of honeypot

Honeypot is a deception tool and it needs to be covert. Honeypot and stenography share same characteristics; once the existence of the honeypot/communication channel is discovered by attacker. If attacker watches out carefully for signs of deception, he will find it soon or later. [35]. Once attacker understands server is running honeypot but not real system, an attacker can avoid or bypass the honeypot network or introduce misleading data into a honeypot.

There are several ways to detect a honeypot. Especially low interaction ones have some unique characteristics, which can be finger printed, such as hardcoded strings, specific service banners or incorrect protocol implementation [36]. In arms race, honeypot attackers and developers had back and forth. Detection of honeypot is a big risk, most of honeypots are open source and user should follow potential honeypot problems and wait developers to fix the issue.

Shortcomings of choice of this technology

Honeypot has no production/actual value; it captures attacks targeting it only. A low interaction honeypot captures automated basic attacks. It captures less information about attacker than high interaction honeypots. Gathering malware depends on capability of low interaction honeypot. Due to the types of vulnerabilities and protocols emulated, collection of malware types is limited**.**

There is roughness of simulation; a more accurate simulation would be more expensive. Simulated operational services are limited and can fail to interact with unknown attack before the vulnerability itself is triggered [37].

3. Low-interaction honeypots and design

3.1. Low-Interaction honeypots

Low interaction honeypots started as relatively simple network emulation tools and evolved in many ways over in 16 years. Over the years there have been many regular cycles in the arms race. Researchers/developers tried to improve better collection of data to know attacker's tools, behavior and faced efforts to fingerprint, detect honeypots [38]. Early honeypots often simply captured data off the wire via a pocket logger that attackers find detecting such monitoring difficult but they can be defeated easily. Use of encrypted protocols by attackers made developers to use keyloggers or patched command shells but detect or disable them were easy for skilled attackers. Later on, the attackers developed rootkit tool but developers modified a rootkit to develop a nonkernel program named Sebek. Sebek was a significant advance over earlier honeypots but it had shortcomings; capturing data was partly successful. Sebek2 overcame many of shortcomings and could hide its network traffic at the kernel level [39]. In 2006 Nepenthes is released. Low interaction honeypots extended to effectively develop a method to collect malware. Nepenthes came over two problems; first one by emulating more complex protocols and secondly scalability that allowed to deploy many honeypots in parallel. The actual work is carried out by several modules [40]. Limitation of nepenthes was collecting only autonomously spreading malware and possibility of detection [16]. Dionaea is created by Markus Kotter for Summer of Code 2009 as a successor of Nepenthes. Dionaea had improvements over Nepenthes in malware acquisition, attack accuracy and logging. Nepenthes used pattern matching on attack communications payloads to detect attacks that had drawback when attacks were made to complex protocols like SMB. Dionaea solved this problem by emulating SMB and MSRPC protocols. Dionaea can detect exploitation of unknown vulnerabilities with Libemu x86 emulator by automatically detecting shellcode within a payload [41].

Among the server side honeypots which specifically designed to collect malwares are Honeybow, Sebek, Amun, Dionaea, Nepenthes and Google Hack Honeypot [19][43]. Honeybow is outdated and not supported, Amun project is not active, Google Hack Honeypot is a simple web honeypot and it has no active user community/public mailing list. Nepenthes is rendered obsolete by Dionaea [19]. In May 2014, during Honeynet

Project Workshop in Warsaw, Poland, David Watson (Chief Research Officer for the Honeynet Project), Angelo Dell'Aera (Chief Executive Officer of the Honeynet Project), Emiliano Martinez (software engineer at VirusTotal) highly recommended Dionaea. Dionaea was chosen for the purpose of collecting malware and data.

3.2. Dionaea

Dionaea is written in C with an interface based on Python that makes it easy to add new modules without recompiling the base. It supports IPv6, IPv4 and Transport Layer Security (TLS) and has real time notification using XMPP. A SQLite 3 database logs the information on each attack and can also produce graphical statistics. It has one of long term supported (LTS) honeypot.

Dionaea honeypot can

- Identify OS with fingerprints that come from IP stack with p0f

- Emulate services

- Has real time notifications

- Collect data in database that easily read or make graph of the data

- Can reply attacks for testing or troubleshooting purposes, by modifying bistreams it is possible to create a metasploit module.

3.2.1. Protocols

Dionaea emulates SMB, HTTP, HTTPs, FTP, TFTP, MSSQL, MySQL, SIP, EPMAP protocols. TFTP functions as a TFTP server. HTTP, HTTPs act as a web server. SMB and EPMAP protocols are essential to collecting malware with dionaea.

SMB protocol

SMB is the protocols for Windows PCs to share files, printers, directories. It is the most attacked port for windows machines because of its long standing history of bugs. Conficker worm [42], one of fastest and largest worm, spread through this protocol. Dionaea implements a python-based version of the Windows Server Message Block

(SMB) protocol as main protocol that allowing it to be properly establish sessions before being exploited by attacking machines. Other low-interaction honeypots cannot simulate this vulnerability that makes Dionaea take advantage over other honeypots [43]. Dionaea tries to provide very detailed emulations of vulnerable services but attacker can perform only certain operations; dissemination of malware, shellcode etc. Only if certain actions occurred after acceptance of connection, then these ones considered as an attack. Some attacks may not create valid SMB session depends on session created by windows api or metasploit even vulnerability is emulated properly. If valid SMB session is created and attacker could send shellcode, libemu may fail to handle or detect properly. Another issue about SMB protocol is about ISPs and source of attacks; ISPs scan their customer machines for vulnerability (default being to scan the machine if specifically not requested to) then they can block whichever port. [44]. In figure, there are decreasing of sources targets 445 ports over the years;

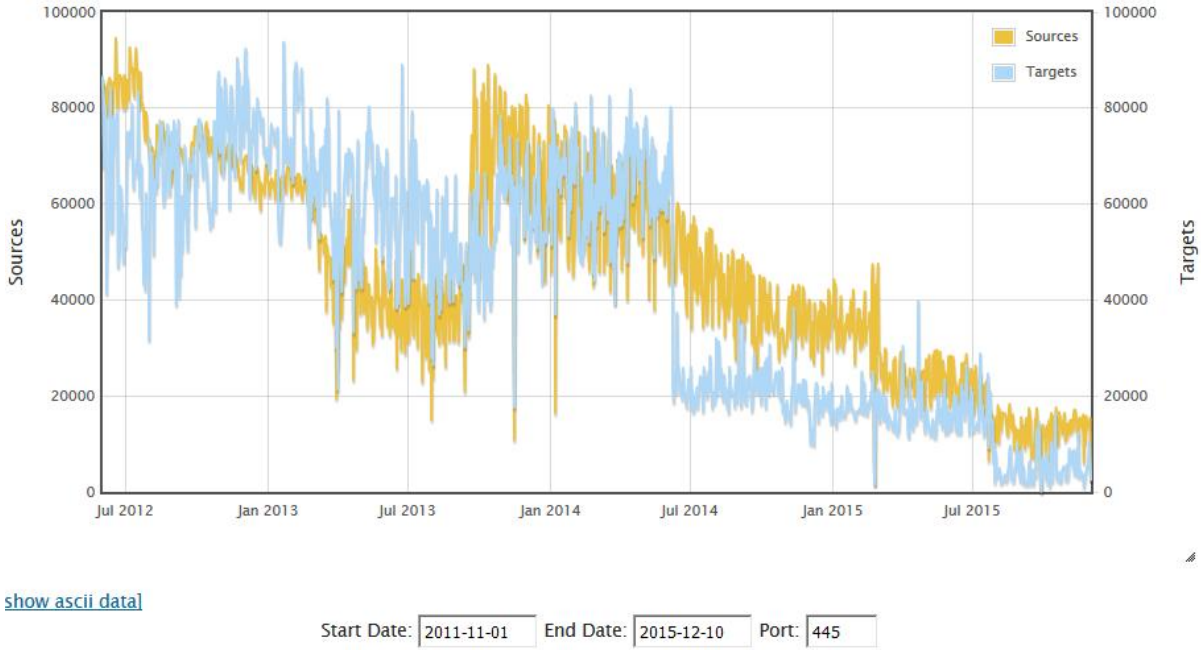


Figure 4 – TCP/UDP port activity [45].

3.2.2. Modules

- Pcap; it detects rejected connections just to record this information.
- Curl; it is used to transfer files from and to server.
- Emu; it is used to detect, profile and if required to execute shellcode.

- Python; it allows controlling some scripts that dionaea uses such as logxmpp, readlogsqltree.py, gnuplotsql, logsql.
- P0f; it gives information about the attackers operating system.

3.2.3. Configuration file

The configuration file is changed according to desired conditions; collect malware and hardening security. Logging part is changed to observe attacker behavior after making changes in network. P0f module is opened to collect more information about connections. HTTP, HTTPs, FTP, TFTP, mirror services are removed to emulate.

Dionaea has 3 modes to listen; a specific IP, more than one address and IPv6. The log levels are all, debug, info, message, warning, critical, error. Dionaea logs are quite verbose, logrotate can be activated.

3.2.4. Utils

Dionaea utilities are written in Python that is useful for quick information and visual view of the attacks. Dionaea utilities are in dionaea/modules/python/util/

- Readlogsqltree.py gives the report about connections according to time requested
- Gnuplotsql.py is a python script that runs queries on sqlite database and creates graphs and index of the data.

3.2.5. Privileges

Dionaea can drop privileges, to run certain action that needs privileges; it creates child process at start up to run these processes. This just make more difficult to get root access to the system.

3.3. Design

Among honeypots Dionaea is the most preferable honeypot for the purpose of collected malware. It has good quality of collecting data, reliability and many support tools [46]. It was placed in DMZ in IT Collage at Solaris server and it has a public IP. Xen Server

was preferred as server. Dionaea is suitable for unix systems and Ubuntu server was chosen. At first Zyntal was used as router but for the first running it was removed.

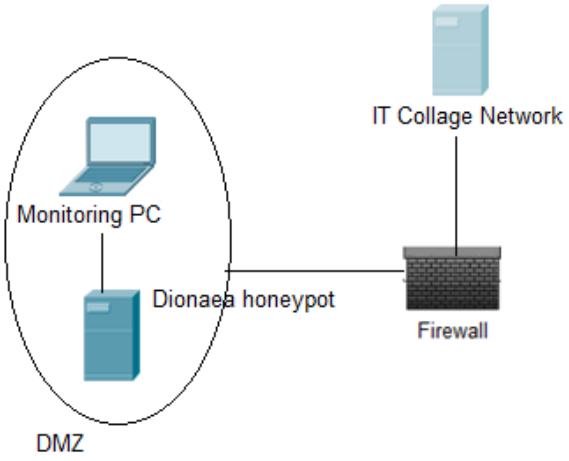


Figure 5 – Honeypot placement in network

Xen Server

Xen Server is completely open source and free, offers more agility and flexibility than other virtualization software. Most commands are used related to network and virtual machines are;

```
xe vif-unplug, vif-destroy, vif-create, template-list,
template-param-set, vm-install, vm-disk-list
```

Other install media template is not supported by Xen Server so it will be created as HVM guests; it is not able to use the native high-performance drivers (PV drivers) included in modern kernels. It is necessary to copy Ubuntu 10.04 template for Ubuntu 12.04.

```
xe vm-install template=Ubuntu\ Lucid\ Lynx\ 10.04\ \(64-bit\)
sr-name-label=Local\ storage  new-name-label="Ubuntu Precise
(64-bit)"
```

Ubuntu Server

The recommended OS for installing dionaea is Ubuntu or Debian Linux. Ubuntu Server 12.04 is used because it has kernel version 2.7 that allows more to do with virtual networking. Version 10 was used but it doesn't support some apps in research then later

Dionaea installed to 12.04 version. It is confirmed that ISPs doesn't block network traffic and ports are open.

Dionaea configuration file

Only SMB, EPMAP, MSSQL, MySQL protocols are chosen for services to work in Dionaea. FTP does not currently have exploit detection so it is disabled for security reasons.

More than one address chosen; addrs = { eth0 = ["0.0.0.0"] }

Running dionaea

Before running it, the file permissions were changed to nobody and nogroup;

sudo chown nobody:nogroup /opt/dionaea/var/dionaea –R

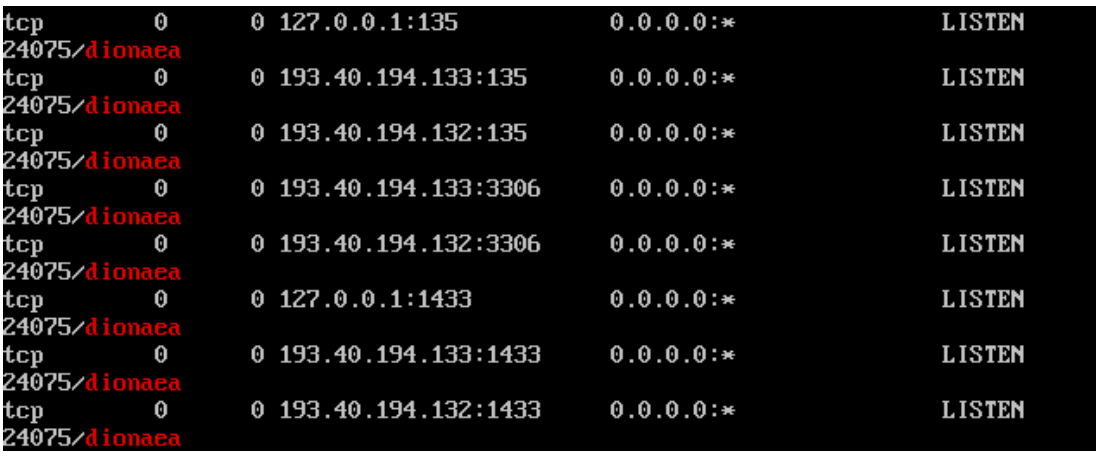
dionaea -l all,-debug -L '*' – running command

It is possible to decide how detailed logging can be. Dionaea can be run as daemon;

sudo /opt/dionaea/bin/dionaea -u nobody -g nogroup -w /opt/dionaea -p /opt/dionaea/var/run/dionaea.pid –D

To check if dionaea is running; ps -ef | grep dionaea

To check which ports Dionaea is listening; netstat -tnlp | grep dionaea



Nmap scan give information which Dionaea services are running. Any unskilled attacker can understand easily that server is running honeypot, emulated services;

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	Dionaea honeypot ftpd
135/tcp	open	msrpc	?
443/tcp	open	ssl/https	?
445/tcp	open	microsoft-ds	Dionaea honeypot smbd
1433/tcp	open	ms-sql-s	Dionaea honeypot MS-SQL server
3306/tcp	open	mysql	MySQL 5.0.54
5060/tcp	open	sip	(SIP end point; Status: 200 OK)
5061/tcp	open	ssl/sip	(SIP end point; Status: 200 OK)
5060/udp	open	sip	(SIP end point; Status: 200 OK)

For dionaea logs, the logrotate is activated. ./readlogsqltree.py -t $(date '+%s')-1*3600 /opt/dionaea/var/dionaea/logsql.sqlite – it lists the connections for last one hour.

4. Low to High Interaction

Honeypot attractiveness can be increased by advertising honeypot involve assigning domain names, running various servers or providing eye-catching content [46]. The real systems will be running that brings more risk but more information about attack and attacker [47]. The compare of data from low and high interaction is important for the research.

4.1.1. Server installation

These servers are installed for research purposes;

- DNS server
- LAMP (Linux, Apache, MySQL, PHP) server

With DNS server attacker may think that there is a LAN in network to intrude. LAMP server is installed to make attacker to think there is a webserver.

Basic network layout designed with DNS server;

```
server.hom.     IN      NS      homepc.server.hom.
server.hom.     IN      MX      10      homepc.server.hom.

$ORIGIN   server.hom.
localhost       IN      A       127.0.0.1

homepc          IN      A       193.40.194.134
print-srv       IN      A       192.168.0.9
router          IN      A       192.168.0.1
server          IN      A       192.168.0.5
ubuntu          IN      A       192.168.0.2
xbox            IN      A       192.168.0.3
```

4.1.2. Tapping

Dionaea can listen to list of IPs in network and bind a service to each IP. Tap interface creates purely virtual interface. Tap works with Ethernet frames. Many tap interfaces can be created as how many free IP there is. Tap interfaces are connected with bridge so no physical interface is used. Multimac [48] package was used to create taps. Different IP and MAC addresses are assigned to each taps.

./multimac 1 (it will create tap0 and tap1)

```
# sudo brctl addbr br0
# sudo brctl addif br0 eth0
# sudo brctl addif br0 tap0 tap1
# sudo ifconfig eth0 down
# sudo ifconfig eth0 0.0.0.0 up
# sudo ifconfig tap0 0.0.0.0 up tap1 0.0.0.0 up
# sudo ifconfig br0 193.40.194.133 up
# sudo ifconfig tap1 hw ether e4:3c:e9:15:42:e6
# sudo ip addr add 193.40.194.132 dev tap1
# sudo ifconfig tap1 193.40.194.132 up
# sudo route add default gw 193.40.194.220
# sudo ifconfig tap0 promisc sudo ifconfig br0 promisc
# sudo nmap –iflist
```

```
************************INTERFACES************************
DEV  (SHORT) IP/MASK           TYPE     UP MAC
lo   (lo)    127.0.0.1/8       loopback up
tap0 (tap0)  193.40.194.132/32 ethernet up E4:3C:E9:15:42:E6
br0  (br0)   193.40.194.133/24 ethernet up 9A:E5:E1:E8:BD:FF

**************************ROUTES**************************
DST/MASK       DEV GATEWAY
193.40.194.0/0 br0
0.0.0.0/0      br0 193.40.194.220
```

Zenmap network schema;

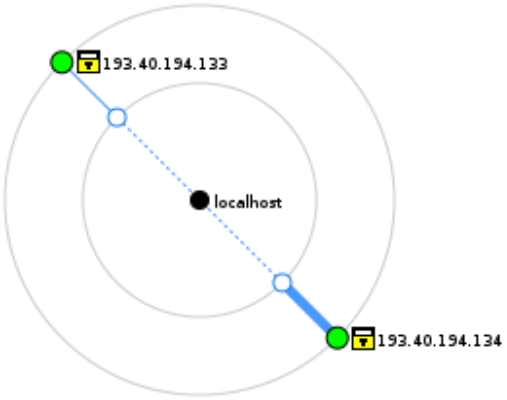


Figure 6: Machine3 Network Schema

4.1.3. Honeyd

Honeyd is a low interaction honeypot client that creates virtual hosts in a network. It can be configured to act like a real operating system or router, switch; the hosts can be configured to run arbitrary services, and their personality can be adapted so that they

appear to be running certain operating systems. There are approximately 1000 personalities of OS's to be chosen. Any services like FTP, HTTP, and SMB can be configured for those operative systems to activate. Honeyd can be used to defeat fingerprinting; option –p for nmap, option –x to react ICMP fingerprinting tools.

Operating system, router and service specification made into /etc/honeypot/honeydstats.conf file;

```
create windows
set windows default tcp action reset
set windows default udp action reset
set windows default icmp action block
set windows personality "Microsoft Windows 2000 SP2"
add windows tcp port 80 "sh /usr/share/honeyd/scripts/win32/win2k/iis.sh $ipsrc$
add windows udp port 137 open
add windows tcp port 137 open
bind 193.40.194.133 windows
```

Running honeyd with options;

```
honeyd -d -i eth0 -f honeyd.conf -p nmap.prints -x xprobe2.conf -a nmap.assoc -0 pf.os
-l /var/log/honeyd/honeyd-packet.log -s /var/log/honeyd/honeyd-service.log
193.40.194.134
```

Running Honeyd with these options will give IP personalities desired for the machines in network. After creating taps and bridge with multimac, assigning windows OS with honeyd, whole virtual network will look like this;

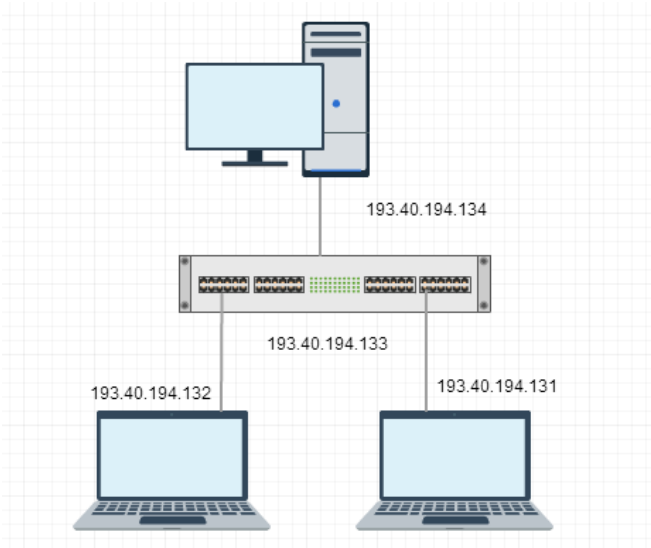


Figure 7: Virtual network schema

4.2. Security

One of the main reason attackers' concern to know if they are dealing with honeypot, they don't want their methods known, especially if they are going to perform zero-day attack. Attackers and honeypot developers came up with different methods to protect themselves.

4.2.1. Detection of VMs

Virtual machines (VM) became useful tool to set them up as honeypots. VM helps to analyze samples with the benefit of snapshot, multiple operating systems, easy to monitoring and isolation [49]. There are different ways to detect VMs. It is possible to detect from its hardware. There specific pieces of hardware that are not configurable but it is possible to patch VM binary to different are values. Another way is the fingerprint of VM; MAC address of the network interface (with arp –a command to get cached MAC address) or from interface (with ipconfig or ifconfig /all command). It can be possible to detect VM from I/O backdoor [50]. Malware also can detect if it is VM or not On average, one in five malware samples will detect virtual machines and abort execution. A malware can figure out if it is in a virtual machine with blue pill [51]. Blue pill tool runs single machine language instruction; takes location of the Interrupt Descriptor Table Register and stores it in table.

4.2.2. Detection of honeypots

Latency is biggest issue about detection of honeypots. Honeypots are logging as the attacker using the machine and execution of commands processed by honeypot can take longer than it has to be. Reply of services like dropped packages or discrepancies in network behaviour can raise suspicion. The common exploits sent to server named in banner can give uncommon response and give conclusion of honeypot to attacker. Many or uncommon open ports from a scan can give strong hint that machine is running honeypot [52].

Detection of Honeyd

Honeyd can respond to malformed network packet as it has received a valid packet that is not expected to get this response. Honeyd handling to fingerprinting visa TCP/IP

stack interactions can raise suspicions; it is needed to validate six major NMAP scan types for that [53]. Another way to understand is checking semantic errors in configuration [54].

4.3. Avoiding Dionaea service identification

After installation Dionaea, Nmap scan reveals Dionaea services in HTTP, SMB, SQL and HTTPS; cat /usr/share/nmap/nmap-service-probes | grep Dionaea

- match ftp m|^220 Welcome to the ftp service\r\n| p/Dionaea honeypot ftpd/
- match http m|^HTTP/1\.0 200 ... </html>\n$| p/Dionaea honeypot httpd/
- match microsoft-ds ... p/Dionaea honeypot smbd/
- match ms-sql-s .... p/Dionaea honeypot MS-SQL server

There is need to change 4 files before compiling Dionaea;

ftp.py → the line self.reply(WELCOME_MSG, "Welcome to the ftp service"), welcome message is changed to “Home server”

smbfields.py → the line OemDomainName: WORKGROUP changed to Webserver and the line Servername is changed to Homeserver.

mssql.py → the line r.VersionToken.TokenType = 0x00 changed to 0xAA

connection.c → in the line X509_NAME_add_entry_by_txt Nepenthes Development Team is changed to Home Web Server.

Nmap scan after changes, Dionaea name is removed;

```
Nmap scan report for 193.40.194.134
Host is up (0.0052s latency).
Not shown: 999 open|filtered ports, 993 filtered ports
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh?
53/tcp   open  domain?
80/tcp   open  http          Apache/2.2.22 (Ubuntu)
135/tcp  open  msrpc?
445/tcp  open  microsoft-ds?
1433/tcp open  ms-sql-s?
3306/tcp open  mysql?
53/udp   open  domain?
```

4.4. Iptables

The network rules had to be flexible not to block everything but also allow attacker to try to exploit. For example if there are more than 5 connections in 30 seconds, this IP is going to be blocked for some minutes.

In Dionaea logs it is observed that there can be undesirable automated attacks from connections. The rules below prevent these attacks;

Clearing all rules in firewall

```
iptables -F
iptables -X
```

Creating three new chains to filter the attacks

```
iptables -N ATTKED
iptables -N ATTK_CHECK
iptables -N SYN_FLOOD
```

Dropping not syn incoming packets;

```
iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

Drop fragmented packets

```
iptables -A INPUT -f -j DROP
```

Drop Xmas packets

```
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
```

Drop null packets against Nmap null scan

```
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
```

Forwarding any incoming tcp packets in the SYN_FLOOD chain

```
iptables -A INPUT -p tcp --syn -j SYN_FLOOD
```

Using hashlimit module to create database of each ip in order to drop any packet that exceed 100 packet per second and keep it in database for 3600 seconds

```
iptables -A SYN_FLOOD -p tcp --syn -m hashlimit --hashlimit
100/sec --hashlimit-burst 3 --hashlimit-htable-expire 3600 --
hashlimit-mode srcip --hashlimit-name testlimit -j ACCEPT
```

Any other packets that are not matched as syn flood will be forwarded in ATTK_CHECK chain

```
iptables -A SYN_FLOOD -j ATTK_CHECK
```

Accept legitimate traffic

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

Keep all IPs matched below for 1800 seconds, if there is no incoming packet from same IP, it removes the IP from database BANNED

```
iptables -A INPUT -p tcp -m tcp --dport 135 -m recent --update
--seconds 1800 --name BANNED --rsource -j DROP

iptables -A INPUT -p tcp -m tcp --dport 445 -m recent --update
--seconds 1800 --name BANNED --rsource -j DROP
```

All new packet with destination port 135 and 445 are forwarded to ATTK_CHECK chain.

```
iptables -A INPUT -p tcp -m tcp --dport 135 -m state --state
NEW -j ATTK_CHECK

iptables -A INPUT -p tcp -m tcp --dport 445 -m state --state
NEW -j ATTK_CHECK
```

Set up log options for the chain ATTKED drop any packet in that chain putting the source ip in BANNED chain

```
iptables -A ATTKED -m limit --limit 5/min -j LOG --log-prefix
"IPTABLES (Rule ATTKED): " --log-level 7

iptables -A ATTKED -m recent --set --name BANNED --rsource -j
DROP
```

Defining new chain for incoming packets that are not matched as attack

```
iptables -A ATTK_CHECK -m recent --set --name ATTK
```

Put IP to ATTK chain if it hits 20 times in 180 seconds

```
iptables -A ATTK_CHECK -m recent --update --seconds 180 --
hitcount 20 --name ATTK --rsource -j ATTKED
```

Put IP to ATTK chain if it hits 6 times in 60 seconds

```
iptables -A ATTK_CHECK -m recent --update --seconds 60 --
hitcount 6 --name ATTK --rsource -j ATTKED
```

Permit rest of the traffic

```
iptables -A ATTK_CHECK -j ACCEPT
```

save ip tables; sudo /sbin/iptables-save

5. Analysis of data and attackers' behavior

5.1. Value of data

Dionaea use incident handler that is logsql python script. It writes interesting incidents to a sqlite database, one of the benefits of this logging is the ability to cluster incidents based on the initial attack when retrieving the data from the database.

The data collected by honeypot gives idea about;

- Attacks are organized or not; attacks were carries out by same person or some group
- Level of attacker; it is automated or professional or amateur hobbyist
- Attacker's behaviour; what attacker were looking for or attacker's intention
- Identify anomalous malicious activity that targeted vulnerability; new IDS/firewall rules can be generated

It is expected that the attacker use a set of malware repeatedly in different attacks; in database files can appear in different names and variants and attacker change IP addresses for different attacks. From datasets which include files, IP addresses left by attacker, it can be easily figured out level of attacker whether same folder is used again even by different source IPs. For different files like binary files, using file similarities with block hashing algorithm, context triggered piecewise hashing (fuzzy hashing), bloom filter hashing to find out whether folders arranged or malicious binary improved. The result will expose level of attack and attacker and same malwares that has totally different file names and MD5 hashes [55].

There can be similarities in attacks in database. One of the models to analyze attack similarities to use document-term matrix which each attack is a document, malware used in an attack is a word. If malware1 was not used in attack1 but malware1 is most similar to malware2 and malware2 was used in attack1 then the similarity is 90%. Use of malware hashes in Natural Language Processing (NLP) technique is for malicious language processing in static analysis and in T-SNE (t-Distributed Stochastic Neighbor Embedding) technique to illustrate segmentation of attacks and in Latent Semantics Indexing (LSI) to compute the similarity level for the attack pairs.

Dionaea database has a total of 26 tables. Connection table is the main table and other tables are dependent. The entity relationship model for dionaea database can be seen in Figure 8.

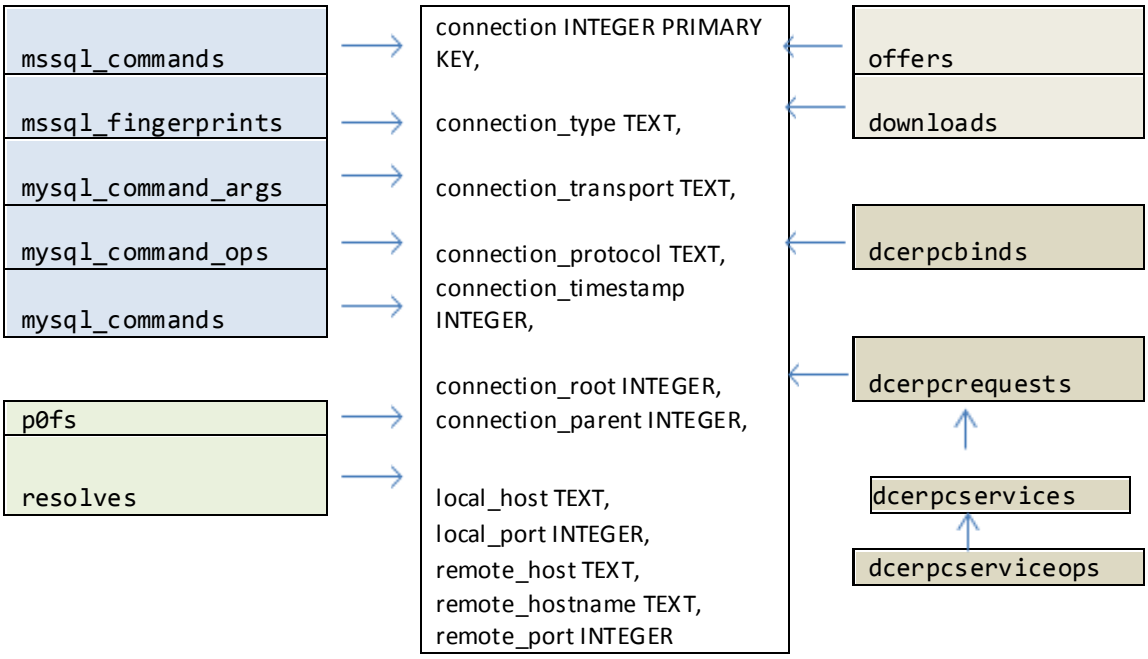


Figure 8; Entity relationship model for data analysis

The related tables are grouped in ERM model in order to have pre-classified datasets. P0fs and resolves tables have the information for honeypot firewall logs and required rules to secure high interaction honeypot against organized and Ddos attacks etc. MySQL/mysql stores commands and arguments to compromise the service, user names and passwords in brute force attacks. Downloads and offers has data of MD5 hashes, offered files and URLs; these data is useful to analyze attack and file similarities. Dcerpc tables have information about exploited and targeted vulnerabilities/services that is very valuable information about attacker's behavior. For this data, decision tree technique can be used to distinguish malicious traffic from legitimate one in order to create IDS rules [56].

5.2. Machine1

First machine run only Dionaea for 57 days and 16 days as high-interaction with DNS server and LAMP server. During only Dionaea was running it could collect 17 malware and during high-interaction no malware was collected.

Quantitative analysis

Quantitative analysis gives general idea. The data in The Table and Graph is from the main table (Connections) where all other tables are dependent to. Connections table has column of connectionID, connection type, connections protocol, time, local port, remote host. Since attacker change IP for different attacks, attacks become anonymous; only peak of connections, correlation of connections through protocols and other parameters included can give useful information about whether attacks were organized or conducted by same group/person, whether there are persistent IPs, type of attacks and effects of systems/servers running etc.

After first time Dionaea started to run, it had 5222 accepted connections out of 58352 [Q1] that 27.82 percentages of connections are to SMB protocol where Dionaea collects malware (Table 2). Dionaea collected 17 malware during this period. There are distinct 55 offers and 34 download of md5 hashes, 18 of them are unique, offered by 11 different URLs.

Comparison of low-interaction and high interaction through connections can be viewed in Table 2 and Figure 9;

Machine 1	Low-interaction [Q4]		High-interaction[Q4]		
	Honeypot Runtime	RF (%)	DNS Server and LAMP	RF (%)	Total
Days	57		16		
# connections accepted [Q2]	3584	100	1638	100	5222
Connections to SMBD [Q3]	997	27.82	58	3.54	1055
Number of malware	17		0		17
SipSession [Q3]	33	0.92	23	1.40	56
epmapper	68	1.90	33	2.01	101
emulation	7	0.20	0		7
ftpd	45	1.26	22	1.34	67
httpd	431	12.03	887	54.15	1518
mssqld	1439	40.15	205	12.52	1644
mysqld	542	15.12	207	12.64	749
remoteshell	22	0.61	3	0.18	25

Table 2 – Machine1 statistical results

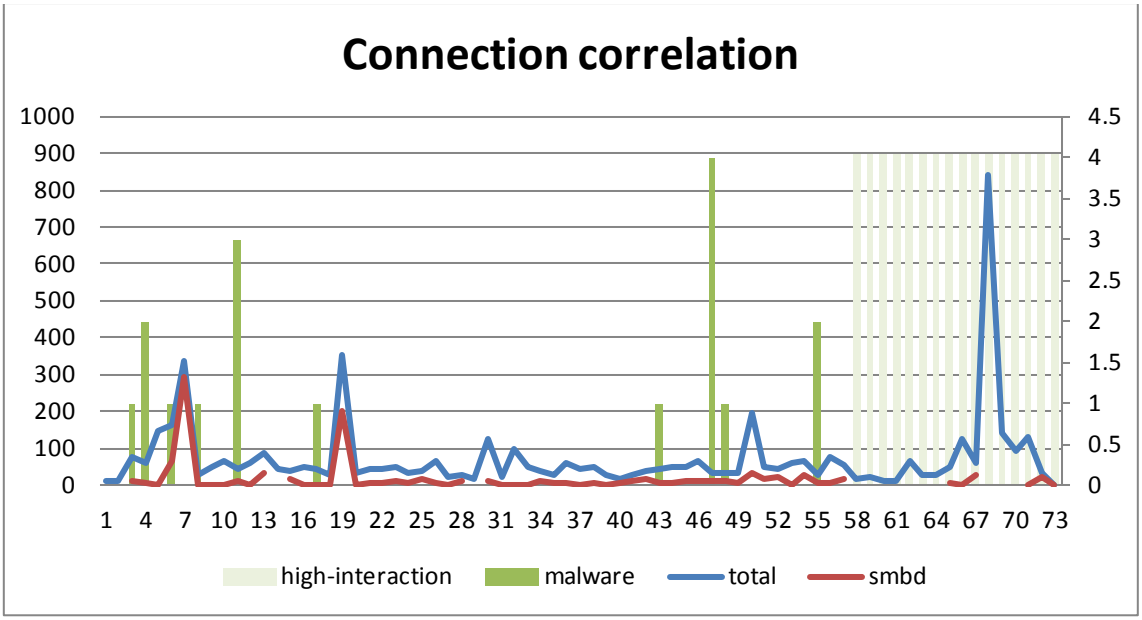


Figure 9 – Machine1 connections graphic

There are 2199 attempt for logins to mysql and mssql from unique 223 source IP, 3551 mssql command, 695 mysql command argument [Q5] can be seen in Table 3;

Number	User	Protocol
1473	sa	mssql
487	root	mysql
106	admin	mysql
106	mysql	mysql
19		mssql
2	IASISUGEL	mysql
2	ddfdffdfdfdfd	mysql
1	cron	mysql
1	database	mysql
1	lemon	mysql
1	opensips	mysql

Table 3 – Machine1 SQL login info

Exploited vulnerabilities are 5 different Microsoft Security Bulletins (From Table 4; MS08-67 – 84 times, MS04-12 – 40 times, MS04-11 – 10 times, MS05-39 – 10 times and MS03-26 – 1 time) and Dionaea couldn't relate 391 unique services targeted to known security bulletin. This may happen if they are new vulnerabilities for Microsoft security bulletin.

Vulnerability [Q6]	Low-interaction [Q7]	High-interaction [Q7]
MS08-67	68	16
MS04-12	36	4
MS04-11	10	0
MS05-39	10	0
MS03-26	1	0

Table 4 – Machine1 List of exploited vulnerabilities

Data analysis

After real services which DNS and LAMP server installed, even though there are more records per day, there is significant decrease for SMB protocol to be targeted. Httpd got majority of connections during this period. This shows that server became suspicious with unrelated open ports and there were more interest to real systems, eventually Dionaea couldn't collect any malware.

There are 3 interesting things in Figure 9. First one, there is a positive correlation between total connections (blue) and connections to SMB protocol (red) but not for collection of malware (green) at low interaction period. Second one, there is no positive correlation between total and connections to SMB protocol during high interaction period. Third one, there are 3 peak points related to connections in the graph;

Peak 1; It has high number of connections in 3 following days; Day 1 has 148 (6 httpd, 30 mssqld, 109 mysqld, 3 smbd), Day 2 has 164 (69 httpd, 24 mssqld, 4 mysqld, 65 smbd), Day 3 has 339 (13 httpd, 30 mssqld, 2 mysqld, 293 smbd) connections. Most of these connections are from India. It is suspicious to be an organized attack.

Peak 2; It has 351 connections (123 http, 24 mssqld, 202 smbd). Two IPs (from China and Virginia, USA) has 227 of these connections to SMB protocol which no malware was collected.

Peak 3; there are 819 connections out of 823 which targeted to httpd from one IP from Chicago, USA. There is not much information, it maybe application layer D-dos attack (HTTP flood) taking half an hour long. It is one-off attack from this IP during all period.

In only suspicious time interval, different IPs has many connections right after HTTP flood in one hour time intervals. Amateur hobbyists or professionals may conduct it [Q11];

Connections | remote host | local port | 1 hour time interval

```
756|66.225.221.124|80|2015-04-09 22:59:59
63|66.225.221.124|80|2015-04-09 23:00:31
2|85.114.141.217|80|2015-04-10 02:01:38
4|104.233.142.206|1433|2015-04-10 05:18:30
2|190.176.130.169|80|2015-04-10 05:28:27
3|107.160.20.152|1433|2015-04-10 05:33:22
5|117.21.176.109|1433|2015-04-10 06:27:39
5|219.235.1.82|1433|2015-04-10 07:51:46
```

The services called related to targeted vulnerabilities (at picture below) are SRVSVC (attempted to run NetPathCanonical to convert a path into a canonical name), ISystemActivator (attempted to run RemoteCreateInstance to execute the buffer overflow), DSSETUP (attempted to call DsRolerUpgradeDownlevelServer to query the configuration of an Active Directory domain member system), PNP (attempted to run PNP_QueryResConfList to exploit a buffer overflow vulnerability in the Windows Plug and Play) and DCOM (attempted to run RemoteActivation to exploit a stack buffer overflow in the RPCSS service) [Q6].

```
84|MS08-67|4b324fc8-1670-01d3-1278-5a47bf6ee188|SRVSVC|NetPathCanonicalize|31
40|MS04-12|000001a0-0000-0000-c000-000000000046|ISystemActivator|RemoteCreateInstance|4
10|MS04-11|3919286a-b10c-11d0-9ba8-00c04fd92ef5|DSSETUP|DsRolerUpgradeDownlevelServer|9
10|MS05-39|8d9f4e40-a03d-11ce-8f69-08003e30051b|PNP|PNP_QueryResConfList|54
1|MS03-26|4d9f4ab8-7d1c-11cf-861e-0020af6e7c57|DCOM|RemoteActivation|0
```

The picture below shows how many times attacker had tries through which service for each malware [Q7]

```
16|166.111.55.67|smbd|SVCCTL|1fab3fab216d9d5266249cb6ea2f918f
10|186.74.238.213|smbd|SVCCTL|7867de13bf22a7f3e3559044053e33e7
7|186.74.238.213|smbd|samr|7867de13bf22a7f3e3559044053e33e7
6|192.210.53.54|smbd|SVCCTL|bf79e90feed96f50c0ba5d7f212757e9
1|200.32.90.66|smbd|SRVSVC|0f302c856d688340076859a02510507a
1|202.39.38.50|smbd|SRVSVC|ec047cf0a85b1787c23a92fd85b9c586
2|213.79.117.28|smbd|PNP|9b5367e777ef931117f113f65cb94502
1|222.90.209.178|smbd|SRVSVC|e8b9b59e93c6c3486191bb26d3089ea7
2|31.168.67.9|smbd|SRVSVC|8d2932dffe1b62d81df3dcd47f7799bc
2|31.168.67.9|smbd|SRVSVC|c99e235fd91a121bbf193c471229d07b
1|66.165.195.28|smbd|SRVSVC|952098cf3c65cfcb52282d8959ddffd3
1|80.58.137.86|smbd|SRVSVC|4ed217391b897fc2d46ec9ce8af282cf
1|82.76.29.197|smbd|SRVSVC|c99e235fd91a121bbf193c471229d07b
1|82.76.29.197|smbd|SRVSVC|d78e79d86b15ed5732c5ddd002f5d38d
1|84.111.156.198|smbd|SRVSVC|9a3f0010617a6fff3f360657bfd6a0ca
6|84.111.156.198|smbd|SRVSVC|d41d8cd98f00b204e9800998ecf8427e
1|84.111.156.198|smbd|SRVSVC|d78e79d86b15ed5732c5ddd002f5d38d
1|94.142.35.218|smbd|SRVSVC|9a3f0010617a6fff3f360657bfd6a0ca
1|94.142.35.218|smbd|SRVSVC|d78e79d86b15ed5732c5ddd002f5d38d
18|94.78.73.119|smbd|SVCCTL|4d4c2729b8aa56e70eaf9ef84e9d5d3d
1|95.42.72.99|smbd|SRVSVC|3875b6257d4d21d51ec13247ee4c1cdb
```

5.3. Machine2

The machine ran as high interaction honeypot including Dionaea, LAMP and DNS server for 6 days.

Machine 2	High Interaction Honeypot	RF (%)
Days	6	
# connections accepted	381	100
Connections to SMBD	222	58.27
Number of malware	1	
epmapper	1	0.26
emulation	1	0.26
ftpd	3	0.79
httpd	28	7.35
mssqld	114	29.92
mysqld	12	3.15

Table 5 – Machine2 statistical results

SMB had 58.27 % of connections that is higher ration than previous machine (Table 5). During 6 days it could collect only 1 malware on the day which had the highest number of connection (Table 6);

	smb	connections	Malware
D1		22	
D2		24	
D3	16	47	
D4	196	228	1
D5	6	27	
D6	4	33	

Table 6 – Machine2 malware/connections table

The login usernames for the attempt to access for mssql and mysql (Table 7);

Number	User	Protocol
109	sa	mssql
6	root	mysql

Table 7 – Machine2 SQL login info

The type and number of exploited vulnerabilities (Table 8)

Vulnerability	Number
MS08-67	5
MS04-12	4
MS04-11	4
MS05-39	4

Table 8 – Machine2 List of exploited vulnerabilities

Only interesting connection in the data is that 210.71.170.3 made 175 connections to smb protocol that took 2 hours 18 minutes.

P0f has detected 250 operating systems; 20 of them empty and other 230 of them has these operating systems and details [Q8];

```
176|Windows|XP SP1+, 2000 SP3
34|Windows|2000 SP4, XP SP1+
20||
11|Windows|XP/2000 (RFC1323+, w+, tstamp-)
4|Windows|2003 (2)
3|Linux|2.6, seldom 2.4 (older, 4)
2|Windows|XP/2000 while downloading (leak!)
```

The last one XP/2000 while downloading (leak!) is suspicious. In the connections, this IP belongs to 210.71.170.3 and it probed 178 times to port 445 in one attack. This is a botnet attack. It tried to exploit 4 different Microsoft security bulletins [Q10];

```
2|210.71.170.3|2015-04-17|MS04-12|445
2|210.71.170.3|2015-04-17|MS04-11|445
1|210.71.170.3|2015-04-17|MS08-67|445
2|210.71.170.3|2015-04-17|MS05-39|445
```

5.4. Machine3

The machine3 run 12 days in total and collected 6 malware. First it started to run listening 3 IPs in network. 8 days later, DNS and LAMP server installed. 2 days later Honeyd installed to give personalities to other 2 IPs.

Machine 3	Honeypot (3 IPs) Runtime			High Interaction			Honeyd			Total
	IP1	IP2	IP3	IP1	IP2	IP3	IP1	IP2	IP3	
Days	8			2			2			12
# connections accepted	181	13	144	47	9	18	33	32	209	688
Connections to SMBD	97	4	65	12		8	9	7	181	383
Number of malware	3	1	2							6
epmapper	4		4	7	2	2	7			26
emulation	3	1	2							6
mssqld	66	7	63	16	6	6	33	24	24	226
mysqld	9	1	8	12	1	2	17	5	6	50
remoteshell	2		2							4

Table 9 – Machine3 statistical data

In first day, Dionaea collected 4 malware from unique source IPs. Number of total connection began to decrease and also collection of malware. After LAMP and DNS server installed, total number of connection started to increase and after honeyd installation, there was significant increase in total number of connection and connection to SMB protocol.

Compare to machine1, when only Dionaea was running standalone, machine3 had better performance about number of malware collected. The reason Dionaea was listening to different IPs and it had no fingerprints against scans.

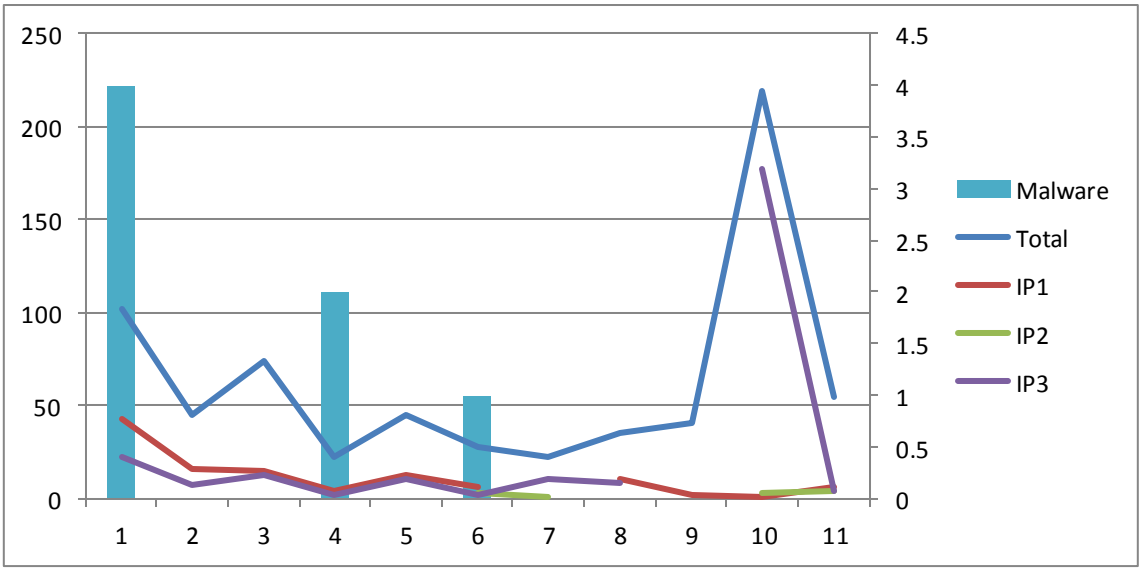


Figure 10 – Machine3 connections graphic

The number of records of names used to login mssql and mysql (Table 10);

Number	User	Protocol
201	Sa	mssql
22	Root	mysql
3	administrator	mysql

Table 10 – Machine3 SQL login info

The exploited vulnerabilities with numbers (Table 11)

Vulnerability	Low-interaction	High-interaction	Honeyd	Total
MS08-67	25	2	2	29
MS04-12	17	6	2	25
MS04-11	12	4	2	18
MS05-39	14	4	2	20

Table 11 – Machine3 Exploited Vulnerabilities

The services used to exploit vulnerabilities;

25|MS04-12|000001a0-0000-0000-c000-000000000046|ISystemActivator|RemoteCreateInstance|4

29|MS08-67|4b324fc8-1670-01d3-1278-5a47bf6ee188|SRVSVC|NetPathCanonicalize|31

18|MS04-11|3919286a-b10c-11d0-9ba8-00c04fd92ef5|DSSETUP|DsRolerUpgradeDownlevelServer|9

20|MS05-39|8d9f4e40-a03d-11ce-8f69-08003e30051b|PNP|PNP_QueryResConfList|54

P0f has detected 38964 operating systems; 28867 of them empty and other 10097 of them has these operating systems and details [Q8];

2|Linux|2.4 (Google crawlbot) - (spammer from misconfigured Linux mail server)
2|Novell|NetWare 5.0
8|ExtremeWare|4.x
22|Linux|2.6? (barebone, rare!)
537|SunOS|4.1.x
5260|Linux 4268|Windows
28867||

Link type of these connections [Q9];

6654|ethernet/modem
1982|pppoe (DSL)
509|sometimes DSL
264|
208|GPRS, T1, FreeS/WAN
195|(Google/AOL)
189|IPv6/IPIP
32|sometimes modem
26|ISDN ppp
18|IPIP tunnel
12|PIX, SMC, sometimes wireless
4|vLAN
4|vtun

5.5. Collected malwares

Dionaea collected many malwares, unfortunately all of them are known (Clamav detected). There could be unknown malware. Malware analysis can be done by dynamic analysis (analyzing stack in memory and dumb of live memory), static analysis (file printing, strings, disassembly etc.)

Hash value and IP addresses about received malware by machine3 with ClamAV;

1|dca8713db4f5b7b84a66b51d925e7f9c|88.160.196.174 - Worm.Allaple-2 FOUND
1|d09151e0d22e416302602c3a9939ec83|84.237.234.179 - Worm.Allaple-2 FOUND
1|c2d8dd5025217b458d0daec026ab8670|193.248.19.76 - Worm.Allaple-316 FOUND
1|ab4f1dbe0ce781ae69dc4cfc8857a483|200.35.53.121 - Trojan.Spy-78857 FOUND
1|3875b6257d4d21d51ec13247ee4c1cdb|110.5.16.69 - Worm.Allaple-306 FOUND
1|3875b6257d4d21d51ec13247ee4c1cdb|113.37.124.51 Worm.Allaple-306 FOUND
1|2f672e2bb137b65bf163bcec2057863a|64.166.97.82 - Worm.Allaple-2 FOUND

file *

2f672e2bb137b65bf163bcec2057863a: PE32 executable (GUI) Intel 80386, for MS Windows
3875b6257d4d21d51ec13247ee4c1cdb: PE32 executable (GUI) Intel 80386, for MS Windows
ab4f1dbe0ce781ae69dc4cfc8857a483: PE32 executable (GUI) Intel 80386, for MS Windows
c2d8dd5025217b458d0daec026ab8670: PE32 executable (GUI) Intel 80386, for MS Windows
d09151e0d22e416302602c3a9939ec83: PE32 executable (GUI) Intel 80386, for MS Windows
dca8713db4f5b7b84a66b51d925e7f9c: PE32 executable (GUI) Intel 80386, for MS Windows

From machine2;

4715dd7a260ec8821a7b621948610795: Worm.Allaple-315 FOUND

From machine1;

```
ec047cf0a85b1787c23a92fd85b9c586: Worm.Allaple-306 FOUND
bf79e90feed96f50c0ba5d7f212757e9: Trojan.Spy-78857 FOUND
7867de13bf22a7f3e3559044053e33e7: Trojan.Agent-173287 FOUND
3875b6257d4d21d51ec13247ee4c1cdb: Worm.Allaple-306 FOUND
4ed217391b897fc2d46ec9ce8af282cf: Worm.Allaple-2 FOUND
0f302c856d688340076859a02510507a: Worm.Allaple-2 FOUND
e8b9b59e93c6c3486191bb26d3089ea7: Worm.Allaple-2 FOUND
8b48f59fb263b1b3ed5f9f2a8cd8fd26: OK
1fab3fab216d9d5266249cb6ea2f918f: Trojan.Spy-78857 FOUND
8d2932dffe1b62d81df3dcd47f7799bc: OK
9b5367e777ef931117f113f65cb94502: Worm.Allaple-80 FOUND
c99e235fd91a121bbf193c471229d07b: INF.Autorun-43 FOUND
9a3f0010617a6fff3f360657bfd6a0ca: INF.Autorun-43 FOUND
4a6e5980ad7d1a4bbe71ec46fa96755e: Trojan.Dropper-27284 FOUND
d41d8cd98f00b204e9800998ecf8427e: Empty file
d78e79d86b15ed5732c5ddd002f5d38d: OK
4d4c2729b8aa56e70eaf9ef84e9d5d3d: Trojan.Spy-78857 FOUND
952098cf3c65cfcb52282d8959ddffd3: Worm.Allaple-223 FOUND
```

```
0f302c856d688340076859a02510507a: PE32 executable (GUI) Intel 80386, for MS Windows
1fab3fab216d9d5266249cb6ea2f918f: data
3875b6257d4d21d51ec13247ee4c1cdb: PE32 executable (GUI) Intel 80386, for MS Windows
4a6e5980ad7d1a4bbe71ec46fa96755e: PE32 executable (GUI) Intel 80386, for MS Windows
4d4c2729b8aa56e70eaf9ef84e9d5d3d: PE32 executable (GUI) Intel 80386, for MS Windows
4ed217391b897fc2d46ec9ce8af282cf: PE32 executable (GUI) Intel 80386, for MS Windows
7867de13bf22a7f3e3559044053e33e7: MS-DOS executable, MZ for MS-DOS
8b48f59fb263b1b3ed5f9f2a8cd8fd26: PE32 executable (console) Intel 80386, for MS Windows
8d2932dffe1b62d81df3dcd47f7799bc: data
952098cf3c65cfcb52282d8959ddffd3: PE32 executable (GUI) Intel 80386, for MS Windows
9a3f0010617a6fff3f360657bfd6a0ca: ASCII text, with CRLF line terminators
9b5367e777ef931117f113f65cb94502: PE32 executable (GUI) Intel 80386, for MS Windows
bf79e90feed96f50c0ba5d7f212757e9: PE32 executable (GUI) Intel 80386, for MS Windows
c99e235fd91a121bbf193c471229d07b: ASCII text, with CRLF line terminators
d41d8cd98f00b204e9800998ecf8427e: empty
d78e79d86b15ed5732c5ddd002f5d38d: data
e8b9b59e93c6c3486191bb26d3089ea7: PE32 executable (GUI) Intel 80386, for MS Windows
ec047cf0a85b1787c23a92fd85b9c586: PE32 executable (GUI) Intel 80386, for MS Windows
```

5.5.1. The purpose of malwares that targeted the system

Worm.Allaple and win32 Virut derivatives

Hash; 2f672e2bb137b65bf163bcec2057863a (Worm.Allaple-2),
Hash; ab4f1dbe0ce781ae69dc4cfc8857a483 (W32/VIRUT)
Hash; c2d8dd5025217b458d0daec026ab8670: (Worm.Allaple-316)
Hash; 952098cf3c65cfcb52282d8959ddffd3 (NetWorm.Win32.Allaple.GEN)

Description; These malware are self replicating and spread by network and removable disks or through shared files with another infected computers. It uses polymorphic encryption.

Function; It affects files, registry files and network communication. It runs as service in network and try to access other computers in LAN.

Worm.Allaple.B injects other viruses or malware into the computers by opening backdoors on system and able to send your personal information to hackers by connecting your computer to remote servers. All viruses belonging to the Virut family also contain an IRC-based backdoor that provides unauthorized access to infected computers.

Backdoorbot derivatives

Hash; dca8713db4f5b7b84a66b51d925e7f9c (Backdoorbot)
Hash; 3875b6257d4d21d51ec13247ee4c1cdb (Bakdoor.Robt)

Description; It allows the computer to be remotely controlled by another user. It is used for financial purposes. By attacker it is used for schemes such as pay per install, sending spam emails, and harvesting personal information and identities are all ways to generate revenue.

Function; When executed it copies to Windows system folder as a random file name. It modifies registry run section to load automatically on the next startup. It can be used to perform Dos attack on other computers. Due to the backdoor abilities, the hacker can steal data from the infected systems. It also disables antivirus and security settings in the infected system.

RemAdm-ProcLaunch – Hacktool (hash; 8b48f59fb263b1b3ed5f9f2a8cd8fd26)

This remote access program is a command line tool that can be used to remotely execute processes on target systems. A user, however, must specify several parameters to properly use this tool.

It is essentially harmless but its ability makes it attractive to users with malicious intent.

Worm.gen.[variant]

Hash; 8d2932dffe1b62d81df3dcd47f7799bc (Worm.Generic.432900)
Hash; d78e79d86b15ed5732c5ddd002f5d38d (Worm.Generic.428092)

Generic Worm does not spread automatically using its own means. It needs an attacking user's intervention in order to reach the affected computer. The means of transmission used include, among others, floppy disks, CD-ROMs, email messages with attached

files, Internet downloads, FTP, IRC channels, peer-to-peer (P2P) file sharing networks, etc.

Unlike single-file detections which identify unique files, a Generic Detection looks for broadly applicable code or behaviour characteristics to evaluate a file's potential for causing harm.

Win32/Autorun variants (INF/GEN)

Hash; c99e235fd91a121bbf193c471229d07b (W32/Autorun.worm.aapq)

Description; It is to detect for 'autorun.inf' files that may be used by worms when spreading to local, network, or removable drives.

Function; It creates a file named 'autorun.inf' in the root of the targeted drive. An autorun.inf file is a Windows instruction file associated with the Autorun feature; the file instructs the operating system what actions it may take with certain files (Open Files, Copy Files, etc) when a drive is accessed.

A malicious program may subvert the Autorun feature to automatically execute a malicious file when the host drive is opened or trick the user into executing a malicious file, usually by using social engineering techniques

dumpsys.exe

Hash; 4a6e5980ad7d1a4bbe71ec46fa96755e (Trojan.Downloader)

It starts servers listening on 127.0.0.1:0. It makes SMTP requests and possibly sending spam and it generates some ICMP traffic.

6. Forensics of honeypot

The server was placed in DMZ in IT Collage network. Somebody hacked honeypot and installed gateway tapping software. By using Arp poisoning, it took over the IP of default gateway and could access to traffic. This hack was not perfect and real gateway couldn't get packets anymore that firewall discovered all those attempts.

A virtual machine was used to install low-interaction honeypot. After 4 days later installing real services and using tapping, the server was compromised. Every individual involving running honeypot was ready for this kind of incident. I didn't panic and did a quick assessment and reported problem to system administrator. In recent Computer Security Incident Handling Guide by NIST [57], the incident response life cycle;

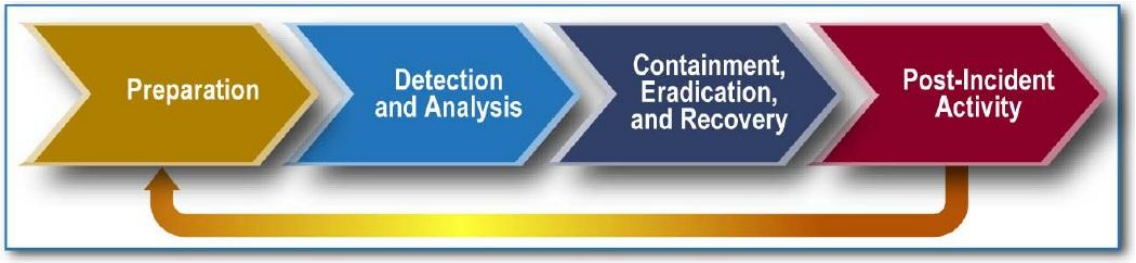


Figure 11 - Incident Response Life Cycle

6.1. System Description

The honeypots were running on virtual machine. After the incident, a snapshot was taken so preservation of evidence was kept. The analysis of state of machine;

1. $/etc/init/rc-sysinit.conf shows the system normally boots into runlevel 2
2. $ sudo who –b shows system rebooted on 2016-05-07 18:14
3. $ last -f /var/log/wtmp wtmp begins Sun May 8 06:25:07 2016
4. Installed and running security tools are apparmor, chkrootkit, ufw, openSSH, denyhosts,
5. The open ports are 3306, 139, 80, 53, 953, 443, 445, 1122
6. The last record Dionaea received was 2015-05-20, 05:30:07 and last sql entry 2015-05-20 02:30:07

The existing users and groups in the machine are;

```
$ getent passwd | awk -F: '{print $1}' | while read name; do groups $name; done
root : root
nobody : nogroup
sshd : nogroup
hopelesslopster : hopelesslopster adm cdrom sudo dip www-data plugdev lpadmin
sambashare
honeyd : honeyd
```

SSH service configuration shows that it doesn't allow root login;

```
$/etc/ssh/sshd_config
PermitRootLogin no
RhostsRSAAuthentication no
HostbasedAuthentication no
#IgnoreUserKnownHosts yes
```

Comparion of passwd and shadow is OK

```
$pwck -r /etc/passwd
$pwck -r /etc/shadow
```

6.2. Login and System Logs

There can be information about a malware incident, attacker IP addresses, compromised user accounts/services and installation of rootkits.

1. $ last -f wtmp; all logins seems legitimate

```
hopeless hvc0          Thu May 21 14:05 - crash (352+04:09)
hopeless hvc0          Thu May 21 14:05 - 14:05  (00:00)
hopeless hvc0          Wed May 20 07:30 - 14:02 (1+06:32)
hopeless hvc0          Wed May 20 07:30 - 07:30  (00:00)
```

2. $ less /var/log/syslog syslog1 restart May 20 07:29:28
3. Chkrootkit found nothing, no infected rootkits
4. There is no result in auth.log for connection
   sudo cat /var/log/auth.log | grep "Accepted password"
   sudo cat /var/log/auth.log | grep "(sshd:session): session opened"
5. ls -lrt | tail -15; nothing suspicious about last modified files
6. lastlog: OK
7. find / -name '.*' | grep '\.[\.]*[^\!-~][^\!-~]*' : OK

8. find / -type f -user root -perm -4000 -exec ls -l {} \; OK
9. grep '[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}' /var/log/auth.log ; OK

6.3. Network logs

There was no capture of traffic during the compromising. Only netstat command gave the result below, it is not sure if it is related to IP that compromised the system because the command was run right after incident handling.

#netstat –anp

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0    768 193.40.194.133:50913    192.58.128.30:53        ESTABLISHED -

Country:United States
Hostname:       j.root-servers.net
```

7. Summarizing the research

The server ran low-interaction honeypot Dionaea almost for 2 months. Later DNS server is installed to show like there is a local network. Honeyd honeypot is used to change fingerprints of operating systems. A virtual network, services and IP are created with taps. For each machine network and firewall rules are written, SQL queries are used for data analysis.

Only safe and efficient system would be running real machines with hardened security. In this research two level of honeypot ran to collect data and a comparison of data made to understand level and types of attacks, whether attacks were organized or not, attacker's behavior from exploited vulnerabilities and what the attackers were looking for. Creating virtual network and IP improved the system. Honeypot could listen to more IPs as they are real machines and services. It is observed that in any change of network or services; there were more connections to honeypot that means it is a very interactive environment with attacker. Hiding operating system or labeling them differently didn't change efficiency since attacker is still looking for vulnerability, chatting with services with packages. It didn't help to fool the attacker.

7.1. Conclusion

The main idea of the thesis is to focus on malwares and analyze purpose of these malwares. It would be a great achievement to collect a new malware. The low interaction honeypot Dionaea is an efficient honeypot to collect malware and makes very easy to analyze attacks with well designed database. High-interaction honeypot couldn't help to increase efficiency of collecting malware samples. Although there were more connections to honeypot and SMB protocol where Dionaea collects malware, the high-interaction honeypot collected no malware samples. Adding two more IPs for Dionaea increased collection of malware without real services running. From these conclusions, this design can be proposed to collect malware samples efficiently; a low-interaction honeypot running on as many IPs on network with few real services. The service identification of honeypot should be prevented and low-interaction honeypot should run few services including SMB protocol. The running real or honeypot services should be logical; it shouldn't give suspicion to attacker.

There are severe concerns to run a system like this and security should be taken seriously. Honeypot is a potentially dangerous tool. Both honeypot and virtual machines have vulnerability. It is recommended to place honeypot in DMZ. The person who implements this kind of system should excel his skills before doing it. Especially for running high-interaction honeypot requires advanced knowledge of firewall and network rules because honeypot is a helpful tool, not a standalone security tool; honeypot should run among other security tools.

Besides knowledge of security, it is needed to have Python knowledge to change modules and knowledge about network rules to respond attacks efficient in Dionaea and SQL experience to analyze collected big sized data.

The proposed system can be enhanced depends on resources like hardware, network. In this design, low- and high-interaction honeypots were installed on a single server but many virtual machines were used and operating system, tools were open source. It had limitations due to hardware and knowledge.

Honeypots are very useful for security but it has important drawback like requiring continues rule implementation for high interaction. Low interaction honeypot has limited risk and excellent tool for beginner to learn about network traffic, attack types and patterns, attacker's behavior, tools and methods

Reasons for failure;

- Server is in DMZ; it requires advanced skills for firewall rules when real systems are installed.
- The idea of using more IPs applied much more later
- It requires good knowledge about Linux kernel support and network rules to keep system secure while taking more risk

References

1. Pandalabs report Q2 2015, April - June 2015 – Panda Security. (http://www.pandasecurity.com/mediacenter/src/uploads/2014/07/Pandalabs-2015-Q2-EN.pdf) (15.06.2015) (Article from quarterly report)
2. Phil Bandy, Michael Money & Karen Worstell, IDFAQ: What is a honeypot? Why do I need one? (https://www.sans.org/security-resources/idfaq/what-is-a-honeypot-why-do-i-need-one/1/11) (01.02.2001) (Article from a database)
3. Dar Ning Kung, An Evolution in Security: Intrusion Prevention (https://www.giac.org/paper/gsec/3594/evolution-security-intrusion-prevention/104648) (02.10.2003) (Article from Global Information Assurance Certification Paper)
4. The Honeynet Project, The Honeynet Project Workshop 2014 (warsaw2014.honeynet.org) (12-14.05.2014) (Reference for honeynet project)
5. Lance Spitzner. Honeypots: Tracking Hackers. September 13, 2002. ISBN: 0-321-10895-7 (Book)
6. William Stallings. Cryptography and Network Security: Principles and Practices November 16, 2005. ISBN 0-13141098-9 (Book)
7. L. J. Locher. Maximum Windows 2000. January 2001. ISBN-13: 075-2063319659
8. R. C. Joshi, Anjali Sardana, Honeypots: A New Paradigm to Information Security, CRC Press, 2011 (Book)
9. Clifford Stoll, The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage, 1989. (Book)
10. Bill Cheswick, An Evening with Berferd, 1991 (Book)
11. Al-Sakib Khan Pathan, The State of the Art in Intrusion Prevention and Detection, 1st edition, 2014. (Book)
12. The Deception ToolKit (http://all.net/dtk/dtk.html) (website)
13. The Know Your Enemy (http://old.honeynet.org/papers/kye.html) (website)
14. Joseph Corey, Local Honeypot Identification, Phrack Inc. 2003, September (http://repo.hackerzvoice.net/depot_ouah/p62-0x07.txt) (Article)
15. Maria Manuela Cruz-Cunha, Irene Maria Portela, Handbook of Research on Digital Crime, Cyberspace Security, and Information Security. 1st edition, IGI Global, 2014. (Book)
16. Provos, Niels; Holz, Thorsten, Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional. July 2007. (Book)
17. Thomas M. Thomas, Donald Stoddard, Network Security First-Step. 2nd edition, Cisco Press, 2014. (Book)

18. Eric Cole, Ronald L. Krutz, James Conley, Network Security Bible By, 2005. (Book)
19. ENISA. Proactive Detection of Security Incidents, November 11, 2012 (Many authors) (Publication)
20. Reto Baumann, Cristian Plattner, Honeypots, 2002 (White paper)
21. R. C. Joshi, Anjali Sardana, Honeypots: A New Paradigm to Information Security**,** CRC Press, 2011. (Book)
22. Roger A. Grimes, Honeypots for Windows, Apress; 2005 edition (Book)
23. L. J. Locher. Maximum Windows 2000. January 2001. ISBN-13: 075-2063319659
24. Greene, Barry Raveendran & Danny Macpherson, Sinkholes: A Swiss Army Knife ISP Security Tool Version 1.8. Nano, 2003 (Book)
25. Keith D. Willett, Information Assurance Architecture, Auerbach Publications, 2003 (Book)
26. John D. Howard, Thomas A. Longstaff, A Common Language for Computer Security Incidents, Sandia National Laboratories, 1998. (Report)
27. Pavol Sokol, Patrik Pekarčík, Tomáš Bajto, Data Collection and Data Analysis in Honeypots and Honeynets, 2015 (spi.unob.cz/papers/2015/2015-19.pdf) (Article)
28. Mohammed H Almeshekah, Eugene H. Spafford, The Case of Using Negative (Deceiving) Information in Data Protection, CERIAS Tech Report 2015-11. (Article)
29. Mohammed H. Almeshekah, Eugene H. Spafford, Using Deceptive Information in Computer Security Defenses, 2014. (Article)
30. Miles A. McQueen Wayne F. Boyer, Deception Used for Cyber Defense of Control Systems, May 2009 (Book)
31. L. J. Locher. Maximum Windows 2000. January 2001. ISBN-13: 075-2063319659
32. Stilianos Vidalis, Eric Llewellyn, Christopher Tobb, ICIW2007- 2nd International Conference on Information Warfare & Security, 2007. (Article)
33. David Watson, Low Interaction Honeypots Revisited (https://www.honeynet.org/node/1267), 2015 (Article)
34. Lance Spitzner, Honeypots: Are They Illegal? (http://www.symantec.com/connect/articles/honeypots-are-they-illegal), 2010, (Article)
35. Thorsten Holz and Frederic Raynal, Detecting Honeypots and other suspicious environments, IEEE, 2005. (Article from an journal)
36. Slavcho Manolov, Roumen Trifonov, The Honeypots – Effective tools for proactive detection of computer security incidents, Proceedings of the International Conference on Information Technologies, Bulgaria, 2013 (Conference Paper)
37. Specification of Advanced Methods for Incident and Security Threats' Detection and Mitigation in a Multi-Domain Environment, GEANT report, 2011. (many author)

38. David Watson, Low Interaction Honeypots Revisited (https://www.honeynet.org/node/1267), 2015 (Article)
39. Wiliam Mccarty, The honeynet arms race, IEEE, 2003. (Article)
40. The Nepenthes Platform: An Efficient Approach to Collect Malware, Nepenthes Team (many author), RAID'06 Proceedings of the 9th international conference on Recent Advances in Intrusion Detection, 2006. (Conference paper)
41. Internet Infrastructure Review, Internet Initiative Japan May 2011 (report)
42. Kelly Burton, The Conficker Worm, accessed on 06.11.2015. (https://www.sans.org/security-resources/malwarefaq/conficker-worm.php) (article)
43. Michael Ligh , Steven Adair, Blake Hartstein, Malware Analyst's Cookbook, 1st edition, 2010. (Book)
44. Gibson Research Corporation, 2008, accessed on 20.01.2016. (https://www.grc.com/port_445.htm) (article)
45. Internet Storm Center, SANS. (https://isc.sans.edu/port.html?port=445)
46. ENISA. Proactive Detection of Security Incidents, November 11, 2012 (Many authors) (Publication)
47. Mohssen Mohammed, Al-Sakib Khan Pathan, Automatic Defense Against Zero-day Polymorphic Worms in Communication Networks, Auerbach Publications; 1st edition. (Book)
48. Multimac (https://www.primianotucci.com/os/multimac) (package source)
49. Tom Liston, On the Cutting Edge: Thwarting Virtual Machine Detection, SANS Internet Storm Center, 2006. (Article)
50. Thorsten Holz and Frederic Raynal, Detecting Honeypots and other suspicious environments, IEEE, 2005. (Article from a journal)
51. Joanna Rutkowska & Alexander Tereshkin, Bluepilling the Xen Hypervisor. Invisible Things Lab, 2008 (Article)
52. Sylconia (http://books.gigatux.nl/mirror/honeypot/final/ch09lev1sec1.html) (website)
53. Craig Valli, Honeyd – A OS Fingerprinting Artifice, Conference Proceeding, Edith Cowan University, 2011. (Conference paper)
54. Simon Innes, Craig Valli, Honeypots: How do you know when you are inside one, Proceedings of Australian Digital Forensics Conference, 2006, 78-83. (Book)
55. Richard Xie, Hunting for Honeypot Attackers: A Data Scientist's Adventure, 2015. Accessed on 21.04.2016. (https://www.endgame.com/blog/hunting-honeypot-attackers-data-scientist%E2%80%99s-adventure). (Article)
56. Pedro Henrique Matheus, Using Decision Trees to Extract IDS Rules from Honeypot Data, International Journal of Cyber-Security and Digital Forensics. (Article from an e-journal)

57. Computer Security Incident Handling Guide, Recommendations of the National Institute of Standards and Technology, Special Publication 800-61, Revision 2 (Book, many author)

Appendix

1. List of SQL queries

1. select count(*) from connections;
2. select count(*) from connections where connection_type = 'accept';
3. select count(*), connection_protocol from connections where connection_type = 'accept' group by connection_protocol;
4. select count(*), connection_protocol as cp, date (connection_timestamp, 'unixepoch') as date from connections group by date, cp;
5. select count(logins.login_username||logins.login_password) as count, logins.login_username, logins.login_password, connections.connection_protocol, connections.local_port from logins, connections where connections.connection = logins.connection group by (logins.login_username||logins.login_password) order by count desc;
6. SELECT COUNT(*), dcerpcserviceop_vuln, dcerpcrequests.dcerpcrequest_uuid, dcerpcservice_name, dcerpcserviceop_name, dcerpcrequest_opnum FROM dcerpcrequests JOIN dcerpcservices ON(dcerpcrequests.dcerpcrequest_uuid == dcerpcservices.dcerpcservice_uuid) LEFT OUTER JOIN dcerpcserviceops ON(dcerpcserviceops.dcerpcserviceop_opnum = dcerpcrequest_opnum AND dcerpcservices.dcerpcservice = dcerpcserviceops.dcerpcservice) WHERE dcerpcserviceop_vuln is not NULL and dcerpcserviceop_vuln != '' GROUP BY dcerpcrequests.dcerpcrequest_uuid,dcerpcservice_name,dcerpcrequest_opnum ORDER BY COUNT(*) DESC;
7. select count(*),date (connection_timestamp, 'unixepoch') as date , dcerpcservice_uuid as suuid ,dcerpcserviceop_vuln as vuln, local_port from connections as c, dcerpcbinds as db, dcerpcrequests as dreq, dcerpcserviceops as sop, dcerpcservices as serv where c.connection = db.connection and db.connection = dreq.connection and dreq.dcerpcrequest_uuid = serv.dcerpcservice_uuid and serv.dcerpcservice = sop.dcerpcservice and sop.dcerpcserviceop_opnum = dreq.dcerpcrequest_opnum and dcerpcserviceop_vuln != '' group by suuid, date, vuln order by date;

8. select count(*), p0f_genre, p0f_detail as d from p0fs group by p0f_genre, d;
9. select count(*), p0f_link from p0fs where p0f_genre <> '' group by p0f_link order by count(*) desc;
10. select count(*), remote_host, date (connection_timestamp, 'unixepoch') as date , dcerpcservice_uuid as suuid ,dcerpcserviceop_vuln as vuln, local_port from connections as c, dcerpcbinds as db, dcerpcrequests as dreq, dcerpcserviceops as sop, dcerpcservices as serv where c.connection = db.connection and db.connection = dreq.connection and dreq.dcerpcrequest_uuid = serv.dcerpcservice_uuid and serv.dcerpcservice = sop.dcerpcservice and sop.dcerpcserviceop_opnum = dreq.dcerpcrequest_opnum and dcerpcserviceop_vuln != '' and remote_host = "210.71.170.3" group by suuid, date, vuln order by date;
11. select count(*), remote_host, local_port, datetime (connection_timestamp, 'unixepoch') as date from connections where date (connection_timestamp, 'unixepoch') = '2015-04-09' group by strftime('%Y%m%d%H0', connection_timestamp, 'unixepoch') + strftime('%M', connection_timestamp, 'unixepoch')/60 order by date;