博士論文

A Study on Detecting Cyber Attack Resources by Coordinated Passive and Active Monitoring

受動的観測と能動的観測の融合によるサイバー攻撃リソースの検出に関する研究

横浜国立大学法人　横浜国立大学大学院

環境情報学府

Yin Minn Pa Pa

March, 2016

A Study on Detecting Cyber Attack Resources by Coordinated Passive and Active Monitoring

受動的観測と能動的観測の融合によるサイバー攻撃リソースの検出に関する研究

by

Yin Minn Pa Pa

March, 2016

A doctoral dissertation submitted to
the Graduate School of Environment and Information Sciences,
Yokohama National University
for the degree of Doctor of Engineering
under the academic supervision of
Professor Tsutomu MATSUMOTO

Acknowledgements

First, I would like to express my deepest appreciation to my principal supervisor, Professor Tsutomu Matsumoto, for his tremendous supports, comments and encouragements during the course of my study. His way of learning and working enlightened me how to approach and handle problems not only for academic purpose but also for life. I am heartily thankful to Associate Professor Katsunari Yoshioka, whose encouragements, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. I am deeply impressed with his quality of cool head and warm heart in manipulating his students. Without his help, I would not have been able to complete my dissertation. I owe my deepest gratitude to Associate Professor Junji Shikata whose advices significantly contributed to improve my dissertation. I also would like to thank Professors Tomoharu Nagao and Tatsunori Mori for their supports as my dissertation committee.

I am also greatly indebted to the staffs of Matsumoto Lab and Yoshioka Lab for their valuable helps and Japanese language supports. I really appreciate to all lab members of Matsumoto Lab and Yoshioka Lab, especially Mrs. Ying Tie, Mr. Hiroshi Mori, Mr. Daisuke Makita, Mr. Rui Tanabe and Mr. Shogo Suzuki for their friendships, helps and debates in research. The days I spent together with them are one of the most delightful days of my life in Japan.

I would like to thank my parent for their greatest supports and sacrifices. Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the dissertation.

Abstract

Detecting cyber attack resources is a critical step towards mitigating today's cyber crimes. That is why defenders focus on detection of attack resources such as botnet, malicious domains, malicious network, etc., utilizing different types of monitoring approaches. Namely, darknet monitoring, honeypot, DNS traffic monitoring, etc., can be considered as passive monitoring because it waits and watches attacks passively. On the other hand, active monitoring such as port scans, banner grabbing, OS fingerprinting, Web crawling and DNS crawling, etc., looks for attack resources actively through different types of scans. Previous researches focus on either of passive or active monitoring approach. According to developments in trend of attacks and defenses, focusing only on one monitoring approach is not enough to understand deeper insights of attack for detection of genuine attack resources. For example, almost all incoming packet to darknet (passive monitoring) can be traditionally considered as malicious but it is not true for now as some packets can also be defenders' scans because of the easiness and popularity of active monitoring among defenders. Thus, in this study, we first propose a framework for coordination of passive and active monitoring for the detection of cyber attack resources in Chapter 3.

Based on introduced framework, this dissertation proposes two novel methods on detection of cyber attack resources. Namely, the first method shows how to detect malicious domains and authoritative name servers by coordinated passive DNS traffic (passive monitoring) with DNS crawling (active monitoring). The second method proposes how to detect IoT botnet abused for different types of today's cyber attacks by coordinated honeypot (passive monitoring) with active probing (active monitoring).

The study initially analyzes ISP's Domain Name System (DNS) traffic, which is data set of passive monitoring approach. From this analysis, we could grasp features such as fraction of blacklisted domains, Server Fail response history, TTL of DNS server's domain, and domain flux size to detect malicious name servers. Chapter 5 discusses technique for detection of malicious authoritative name servers using these four features. With these features, we evaluate 74,830 authoritative DNS servers of domains observed at a cache DNS server. As a result, we determine 31, 15, and 85 servers as malicious, respectively using fraction of blacklisted domains, TTL of DNS server's domain, and domain flux. We confirm that 21% of the detected servers are true positive according to several published security reports exhibiting the possibility of these features as metric to find malicious DNS servers. From this preliminary study, we find out that domain flux size feature is quite strong for detection of malicious authoritative name servers. Thus, more specific and carefully categorized features of domain flux size feature are studied and propose a comprehensive detection method explained in Chapter 6.

In Chapter 6, we present a novel method for detecting malicious "domains" (noted as d) and malicious "authoritative name servers" (noted as ns-d) based on their distinct mappings to "IP addresses" (noted as IP). Namely, we present three features to detect them; 1) Single ns-d is mapped to many IP, 2) Single IP is mapped to many ns-d, and 3) Single IP is mapped to both ns-d and d. We evaluate proposed method in terms of accuracy and coverage in detection of malicious d and ns-d. The evaluation shows that our detection method can achieve significantly low false positive rate in detecting both malicious d and ns-d without relying on any previous knowledge, such as blacklists or whitelists.

In Chapter 7, we detect IoT botnet and reveal current IoT threats proposing IoTPOT, which is a honeypot system in which both active and passive monitoring approaches are coordinated. IoTPOT emulates IoT devices and persuade attackers

to intrude it and infect malware (computer virus). While honeypot portion of IoTPOT captures malware as passive monitoring system, the scanner portion of IoTPOT performs active probe of infected IoT devices in order to detect attacker's IoT botnet. With this approach, during 81 days of operation, we observed 481,521 download attempts of malware binaries from 79,935 visiting IP. By analyzing the observation results of honeypot and captured malware samples, we show that there are currently at least 6 distinct DDoS malware families targeting Telnet-enabled IoT devices and one of the families has quickly evolved to target more devices with as many as 9 different CPU architectures. We also reveal that IoT devices are abused for more than 11 different types of today's cyber attacks. Finally, we point out that attacker's current IoT botnet is composed of over 200,000 IP addresses of more than 60 different types of IoT devices. We also shared our malware samples and traffic with more than 11 international organizations for the improvement of IoT related researches.

Table of Contents

List of Figures x

List of Tables xii

Chapter 1 1

Introduction 1

1.1. Motivations and Contributions 1

1.2. Organization 4

Chapter 2 5

Background 5

2.1. Attacker's Tactics on DNS protocol 5

2.1.1 Fluxing in DNS 5

2.1.2 Malicious Authoritative DNS Servers 6

2.2. Telnet Protocol Based Compromises 7

Chapter 3 10

Methodology 10

3.1. Passive Monitoring Techniques 10

3.2. Active Monitoring Techniques 10

3.3. The Need for Coordination of Passive and Active Monitoring 11

3.4. Coordination of Passive and Active Monitoring 12

Chapter 4 16

Related Works 16

4.1. Related Studies for detection of malicious authoritative name servers .. 16

4.2. Related Studies for detection of IoT botnet 17

4.3. Related Studies for coordination of passive and active monitoring 19

Chapter 5 ..... **20**

Finding Malicious Authoritative DNS Servers by DNS traffic **20**

Introduction **20**

5.1. Features for detecting Malicious Authoritative DNS servers **20**

5.1.1. Feature 1: Fraction of blacklisted domains 20

5.1.2. Feature 2: Server Fail Response History 21

5.1.3. Feature 3: TTL of DNS Server's Domain Name 21

5.1.4. Feature 4: Domain Flux 22

5.2. Experiment **22**

5.2.1. Feature 1 (Fraction of blacklisted domains) 23

5.2.2. Feature 2 (Server Fail History) 24

5.2.3. Feature 3 (TTL of DNS server's domain name) 24

5.2.4. Feature 4 (Domain Flux) 24

5.3. Results and Discussions **25**

5.4. Conclusion **27**

Chapter 6 ..... **28**

Detecting Malicious Domains and Authoritative Name Servers Based on Their Distinct Mappings to IP Addresses **28**

Introduction **28**

6.1. Features **28**

6.1.1. Mappings of d, ns-d and Respective IP 28

6.1.2. Feature One: single ns-d is mapped to many IP 29

6.1.3. Feature 30

Two: single IP is mapped to many ns-d 30

6.1.4. Feature Three: single IP is mapped to both ns-d and d 30

6.2. Approach **31**

6.2.1. The Proposed Method 31

6.2.2. Step One: Monitoring on Mappings....32
6.2.3. Step Two: Analysis on Mappings....34
6.2.4. Step Three: Expanding Malicious List....35
6.3. Evaluation....35
6.3.1. Experiment and Results....35
6.4. Evaluation Methods and Results....41
6.4.1. Evaluation of d....42
6.4.2. Evaluation of ns-d....44
6.4.3. Evaluation on IP....47
6.5. Discussion on Monitoring Period....48
6.6. Conclusion....49
Chapter 7....51
IoTPOT: A Novel Honeypot for Revealing Current IoT Threats....51
Introduction....51
7.1. Telnet Protocol....51
7.2. IoTPOT Design....52
7.3. IoTPOT Implementation....53
7.4. Observation Results....55
7.4.1. Stage 1: Intrusion....57
7.4.2. Stage 2: Infection....57
7.4.3. Stage 3: Monetization....61
7.5. IoT Sandbox (IoTBOX)....61
7.5.1. IoTBOX Design....62
7.5.2. Analysis Results by IoTBOX....64
7.5.3. Analysis on Attacks....65
7.5.4. Overview of Attacking Botnet....68
7.6. Conclusion....69

Chapter 8 ..... **71**

Conclusion and Future Works **71**

8.1. Conclusion **71**

8.2. Future Works **72**

Bibliography **73**

List of Papers **77**

List of Figures

Figure 1 - Example of TLD zone with malicious domain 6
Figure 2 - Packsts and hosts on 23/TCP per day per darknet IP 8
Figure 3 - Framework for coordination of passive and active monitoring 12
Figure 4 - First Detection Method 13
Figure 5 - Second Detection Method 14
Figure 6 - The flow of experiment 22
Figure 7 - Mapping of d, ns-d and IP 28
Figure 8 - Feature one 28
Figure 9 - Feature two 29
Figure 10 - Feature three 30
Figure 11 - Overview of proposed method 30
Figure 12 - Analysis procedure in each step of the proposed method 31
Figure 13 - Finding the mappings 31
Figure 14 - Typical structure of all three features 33
Figure 15 - FPR and FNR of different threshold values 36
Figure 16 - Example of mapping that meet feature one 38
Figure 17 - Two examples of mappings with a similar structure that meet feature two 38
Figure 18 - Example of mapping that meets feature three 39
Figure 19 - Number of d, ns-d and respective IP obtained by step two 39
Figure 20 - Number of d, ns-d and respective IP obtained by step 3 40
Figure 21 - FPR and FNR of d 41
Figure 22 - Example of some evaluation results on d 42
Figure 23 - FPR and FNR of ns-d 45
Figure 24 - Some Malicious IP 46

Figure 25 - FPR by each monitoring period (one month, two months and three months, etc...) 47
Figure 26 - FNR by each monitoring period (one month, two months and three months, etc..) 48
Figure 27 - Number of detected malicious domains in each monitoring period 48
Figure 28 - Telnet protocol 51
Figure 29 - Overview of IoTPOT 54
Figure 30 - Overview of IoTBOX 63
Figure 31 - Overview of attack by IoTBOX 65
Figure 32 - Overview of observed attacks by IoTPOT and IoTBOX 66
Figure 33 - Botnet architecture 69
Figure 34 - Coordinated attack of ZORRO family observed by IoTPOT 69

List of Tables

Table 1 - Scanning hosts and device models 9
Table 2 - DNS Servers with high % of black domains 25
Table 3 – DNS servers involving with flux-flux 26
Table 4 - Different Combinations of Features 34
Table 5 - Numbers of d, ns-d and respective IP 36
Table 6 - Benign and malicious instances from output of step two 41
Table 7 - Keywords and type of malicious activities 44
Table 8 - Major log in patterns observed by IoTPOT 58
Table 9 - Patterns of command sequence observed by IoTPOT 59
Table 10 - Clustering results of collected samples by characteristic strings in the binaries 60

Chapter 1

Introduction

1.1. Motivations and Contributions

Detecting cyber attack resources is a critical step towards mitigating today's cyber crimes. That is why defenders focus on detection of attack resources such as botnet, malicious domains, malicious network, etc., utilizing different types of monitoring approaches. Namely, darknet monitoring, honeypot, domain name system (DNS) traffic monitoring, etc., can be considered as passive monitoring because it waits and watches attacks passively. On the other hand, active monitoring such as port scans, banner grabbing, OS fingerprinting, Web crawling and DNS crawling, etc., looks for attack resources actively through different types of scans. According to developments in trend of attacks and defenses, focusing only on one monitoring approach is not enough to understand deeper insights of attack for detection of genuine attack resources. For example, almost all incoming packet to darknet (passive monitoring) can be traditionally considered as malicious but it is not true for now as some packets can also be defenders' scans because of the easiness and popularity of active monitoring among defenders. Thus, recently, defenders focus on detection of attack resources by both passive and active monitoring. However, how to coordination passive and active monitoring efficiently for the detection of cyber attack resources is not proposed yet. Thus, in this study, we first propose a framework for coordination of passive and active monitoring for the detection of cyber attack resources.

Based on introduced framework, this dissertation proposes two novel methods on detection of cyber attack resources. The first method shows how to detect malicious domains and authoritative name servers by coordinated passive DNS traffic (passive monitoring) with DNS crawling (active monitoring). The

second method proposes how to detect IoT botnet abused for different types of today's cyber attacks by coordinated honeypot (passive monitoring) with active probing (active monitoring).

The first method focuses on detection of malicious domains and authoritative name servers. As DNS is a very efficient, robust and low-cost communication channel, domains are widely abused for malicious online activities, such as connecting a large number of compromised hosts and attacker's command and control (C&C) servers, phishing, etc. Attackers manage these malicious domains at authoritative name server, for example, changing corresponding IP address of malicious domain over time to hide IP addresses of C&C servers. There can be different cases in which attackers obtain control of authoritative name server. For example, the authoritative name server that attackers are abusing can be a server setup by DNS hosting service or attackers themselves. However, how attackers are abusing authoritative name servers to manage their malicious domains is not well studied. If we know this, there is a possibility to detect not only malicious domains but also malicious authoritative name servers. To detect such resources, the study initially analyzes ISP's Domain Name System (DNS) traffic, which is data set of passive monitoring approach. From this analysis, we could grasp features such as fraction of blacklisted domains, Server Fail response history, TTL of DNS server's domain, and domain flux size to detect malicious name servers. Chapter 5 discusses the novel technique for detection of malicious authoritative name servers using these four features. From this preliminary study, we find out that domain flux size feature is quite strong for detection of malicious authoritative name servers. Thus, more specific and carefully categorized features of domain flux size feature are studied and we present a novel method for detecting malicious "domains" (noted as d) and malicious "authoritative name servers" (noted as ns-d) based on their distinct mappings to "IP addresses" (noted as IP) in

Chapter 6. Namely, we present three distinct features to detect them; 1) Single ns-d is mapped to many IP, 2) Single IP is mapped to many ns-d, and 3) Single IP is mapped to both ns-d and d. We evaluate the proposed method in terms of accuracy and coverage in detection of malicious d and ns-d. The evaluation shows that our detection method can achieve significantly low false positive rate in detecting both malicious d and ns-d without relying on any previous knowledge, such as blacklists or whitelists.

Second method focuses on detection of IoT botnet as we are entering a new era of Internet of Things (IoT). In the past, Internet is nothing but an **inter**national **net**work of computers. But, nowadays, Internet has been changed into **inter**national **net**work of almost everything. Our smart phone, gaming console, camera, watch, glasses, TV, refrigerator, air-con, and even washing machine are connected to Internet. In addition, our critical infrastructures such as dam, transportation systems, financial services systems, health care facilities and industries are connected to Internet. These Internet connected things (IoT devices) change the way we live and work to a smarter and more efficient directions. On the other hand, IoT devices are attractive playgrounds for attackers, as opposed to personal computers. Most IoT devices are 24/7 online, have no antivirus installed and have weak login passwords. Seeing these trends, we believe that we are also in era of danger by exploits on these IoT devices.

In order to know how much we are in danger of such exploits and how to solve the problems, we analyze the increasing threats against IoT devices. Our preliminary research reveals that attacks to IoT devices have rocketed since 2014. To know more on currently very active attacks, we propose IoTPOT, in which both active and passive monitoring approaches are coordinated. While honeypot portion of IoTPOT captures malware as passive monitoring system, the scanner portion of IoTPOT performs active probe of infected IoT devices visiting to honeypot in

order to detect attacker's IoT botnet. With this approach, during 81 days of operation, we observed 481,521 download attempts of malware binaries from 79,935 visiting IP. We also confirm that none of these binaries could have been captured by existing honeypots that handle Telnet protocol such as honeyd and telnet password honeypot because they are not able to handle different incoming commands sent by the attackers. Active probing of IoTPOT reveals that attacker's current IoT botnet is composed of more than 60 different types of IoT devices including more than 200,000 IoT devices.

1.2. Organization

The rest of this dissertation is organized as follows. Chapter 2 presents the background. Chapter 3 introduced a framework for detection of cyber attack resources. Chapter 4 describes related works.

Chapter 5 discusses the novel technique for detection of malicious authoritative name servers by passive DNS analysis. The work on Chapter 5 is presented in Information and Communication System Security (ICSS-2013, paper number T-1 in "Technical Reports" of "List of Papers" section).

Chapter 6 proposes a novel method for detecting malicious "domains" and malicious "authoritative name servers" by DNS crawling using features understood by passive DNS traffic analysis. This work explained in Chapter 6 is published in Journal of Information Processing (JIP, Japan, Vol 23, No.5, pages 623-632, paper number J-1 in "Reviewed Papers in Journals" of "List of Papers" section).

Chapter 7 presents a novel IoT honeypot for detecting IoT botnet and understanding insights of it. The work is presented in 9th USENIX Workshop on Offensive Technologies (WOOT's-2015, paper number I-2 in "Reviewed papers in International Conference Proceedings" of "List of Papers" section).

Chapter 2

Background

2.1. Attacker's Tactics on DNS protocol

2.1.1 Fluxing in DNS

Attackers such as bot headers need technologies to resist blacklisting of their domains and IP addresses to keep the channel between their bot agents and C&C infrastructure. For that, fluxing is one of the most suitable technologies. There are two types of fluxing: IP flux and domain flux.

IP flux refers to the constant change of IP addresses related to a particular fully qualified domain name (FQDN). As the changes of IP addresses happen in a short time, IP flux is commonly referred to as "fast-flux". There are two types of fast-flux: single-flux and double-flux [1]. Single flux is an IP flux in which the associating IP address for a particular FQDN changes rapidly. The native DNS's round robin and TTL configuration of A record are abused to realize the single flux. In double flux, not only the IP address of FQDN (A RR) but also IP address of domain DNS server (NS RR) changes rapidly.

Domain flux is the inverse of IP flux. The domain flux can be referred to the constant change of FQDN related to a particular IP address. Native wildcard feature of DNS is abused for realizing domain flux. The list of FQDN may be hard-coded in the bot agents, obtained from remote hosts, or internally generated by Domain Generation Algorithm (DGA) in the bot agents. DGA creates a dynamic list of multiple FQDN. Since the domain names are dynamically generated in volume and typically have a life of only a single day, the rapid turnover makes it very difficult to investigate or block every possible domain name [2].

2.1.2 Malicious Authoritative DNS Servers

In this study, we consider an authoritative DNS server that is heavily involved in the malicious online activities as a malicious authoritative DNS server. There can be at least four types of malicious authoritative DNS servers:

- The DNS servers setup by the attackers
- The compromised DNS servers with which an attacker has full control
- The DNS servers on server hosting services (e.g. bullet proof hosting services)
- The dynamic DNS services abused by attackers

For fast flux domains, attackers need to have full control in changing RR of an authoritative DNS server so that he or she can abuse on round robin feature of DNS. For this, they need to register NS record for their SLD domain in TLD zone through registrar. An example zone file of a TLD DNS server with malicious domains is shown in Figure 1.

```
malicious.tld.      360  IN  NS  ns1.malicious.tld.
malicious.tld.      360  IN  NS  ns2.malicious.tld.
ns1.malicious.tld.  180  IN  A   1.2.3.4.
ns2.malicious.tld.  180  IN  A   5.6.7.8.
```

Figure 1 - Example of TLD zone with malicious domain

After the registration, the attacker has control on “malicious.tld” zone that is stored in his authoritative DNS servers, namely, 1.2.3.4 and 5.6.7.8. In the single flux, these two NS records will be static. In the double flux, the attackers change these two A records in time by adding a proxy layer to prevent their own DNS server [3] from being spotted.

Another existing technique is the domain flux with DGA generated domains. The attackers implement an algorithm to internally generate domain names of C&C servers for their bot agents to contact. Because the input of the algorithm often includes time information, the output domains can vary over time.

For this scenario, the attacker registers a portion of DGA generated domains beforehand. The registration of such DGA domains can be realized with all types of DNS servers described above.

In case of W32.Morto worm [4], it has added another C&C communication vector by supplying remote commands through DNS records. The record type that W32.Morto uses for its communication protocol is the TXT record [4]. In this case, the authoritative DNS server replying TXT records may be attackers own DNS server or compromised one.

All these attacks take advantage of the existing DNS infrastructure. We point out that in order to efficiently realize such attacks as their needs the attackers should have authoritative DNS servers in their control and finding such malicious servers is the objective of this study.

2.2. Telnet Protocol Based Compromises

Until now, there are only anecdotal reports on Telnet-based compromises. Thus, we investigate how the situation of Telnet-based compromises has changed. To this end, we analyze a darknet of NICTER [5] Japan's darknet monitoring system that monitors over 209,000 IP addresses presently. Figure 2 shows the traffic on 23/TCP since 2005, both in terms of packets and source IP addresses per day (averaged over all IP addresses in the darknet). The data shows a recent increase of scans for Telnet. According to the previous study [6], the large peak in the end of 2012 is caused by the activities of Carna botnet, created by anonymous hacker for Internet Census by compromising a large number of IoT devices such as routers [7] Since 2014, even after the deactivation of Carna botnet, both the number of packets on 23/TCP and their senders have rapidly increased and dominated the darknet – observing more than 209,497 average scanning sources

per day, which is 52.5% of all sources, in the darknet in the first week of March 2015.

We used p0f for passive OS fingerprinting [8] and determined that among the scanning 29,844 hosts (sampled from 148 darknet IP, 2015/03/05 to 2015/03/10), 91% of them runs Linux. We also connected back to these hosts on 23/TCP and 80/TCP, collected Telnet banners and web contents if any, and manually categorized them by device types. For example, if there is a telling keyword such as "DVR" in HTTP title, we categorize this device as Digital Video Recorder (DVR). If not, we search on Internet using HTTP title as key word and carefully categorize devices by reading available manuals. We also group device models of a particular device type by different HTTP titles. For example, HTTP titles such as "NetDVrV1" and "NetDvrV3" will be counted as two device models of DVR device type. With this way, we found more than 34 different types of IoT devices including 19 different models of DVR, 16 models of IP Camera, 45 models of wireless routers. Moreover, devices such as metrological satellite, heat pumps, parking management system, fire alarm system, solid-state recorders and TV have scanned our darknet on 23/TCP. Table 1 shows top ten attacking hosts and device models of inferred device types. Summarizing, these results show that various IoT devices are already involved in the ongoing attacks.

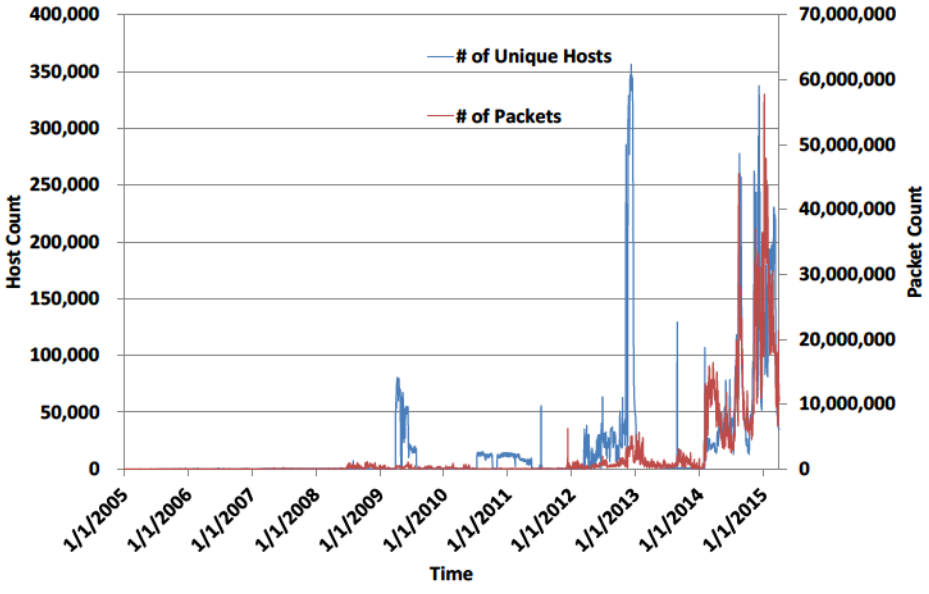


Figure 2 - Packets and hosts on 23/TCP per day per darknet IP

Table 1 - Scanning hosts and device models

Device Type	Host Count	Device Model Count
DVR	1,509	19
IP Camera	523	16
Wireless Router	118	45
Customer Premises Equipment	65	1
Industrial Video Server	22	1
TV Receiver	19	2
Heat Pump	10	1
EMU System	9	1
Digital Video Scalar	5	2
Router	4	3

Chapter 3

Methodology

Attackers manage cyber attack resources and make money by attacking victims of various types such as government, business, industry, etc. In such situation, it is very important to mitigate these cyber attack resources. The first step towards mitigation is detection of cyber attack resources. This chapter presents methodology for detection of cyber attack resources by coordination of active and passive monitoring techniques.

3.1. Passive Monitoring Techniques

Passive monitoring techniques wait and watch attacks passively. It can be mainly categorized into two types, dedicated and operational monitoring. Those, dedicated monitoring includes many different types of honeypot systems attracting the attackers in term of service, system and data [9][10][11] [12][13][14][15][16][17][18]. For example, honeypot systems luring services of different protocols such as web, ssh and DNS exist. Moreover, honeypots mimicking industrial control systems, window systems and those with attractive data for attackers such as e-mails accounts, user accounts and FTP accounts exist. Operational monitoring includes darknet monitoring, which is unused IP monitoring, and other network traffic monitoring such as DNS and HTTP traffics.

3.2. Active Monitoring Techniques

Active monitoring techniques include different types of network scans. Depending on the purpose of scans, it can be generally categorized into three; scan for fingerprinting, vulnerability detection and malicious detection. Scans for fingerprinting purpose includes banner grabbing, OS fingerprinting, port scanning, network tracing, location checking, whois checking and reverse DNS techniques [19][20][21][22]. In case of scans for vulnerability detection, web crawling for

detection of vulnerability in web applications such as cross-site scripting, SQL injection, Wrodpress and Joomla running on web servers [23], DNS crawling for detection of misconfiguration of zone transfer [24] [25] and Heartbleed scanner [26], etc., exist. For malicious detection, DNS crawling for profiling of DNS resource records such as domain and IP, scans using first payload of malware to find C&C servers and web client honeypot [27][28] to detect malicious URL exists.

As the main purpose of this study is to detect malicious cyber attack resources, we do not consider active monitoring for the purpose of vulnerability detection in further discussions.

3.3. The Need for Coordination of Passive and Active Monitoring

Passive monitoring such as darknet and passive DNS traffic are highly resourceful. In the past, we could simply assume that most of the incoming traffic on darknet were relating to malicious activities. Recently, due to the increase in network scans by defenders for security measurement, it is difficult to assume that these traffics are malicious. In the same way, in case of passive DNS traffic, due to the advancement in techniques of handling domain name, for example, domain to IP mapping in content delivery networks, it is not easy to say exactly which domain is malicious just by analyzing it. Thus, in order to improve current situations, rather than focusing on one particular monitoring technique, it is necessary to learn valuable knowledge such as behaviors of maliciousness, trend or tactics of attackers from one monitoring technique and coordinate with another appropriate monitoring technique for the improvement in detection of cyber attack resources.

On the other hand, although data sets of active monitoring, such as Shodan [29], Scanio [30], Censys [31] are quite resourceful, such dataset alone cannot lead

to the concrete understanding of malicious activities and detection of cyber attack resources. For example, active monitoring data set of full IPV-4 FTP banners grabs [30] is quite resourceful. However, it is still difficult to understand what types of devices are highly targeted for what malicious activity with that data alone. Thus, to grasp the concrete story of today's cyber attack, coordination of active and monitoring in accordance with detected malicious activity by each monitoring is required.

In addition, coordination between passive and active monitoring in appropriate timings helps in effective detection of cyber attack resources. Due to the dynamic behavior of IPV-4 addresses, the IP address can be changed in a short time. Thus, after malicious activity of a particular IP is confirmed by passive monitoring, active monitoring on that IP should be started as soon as possible. In other words, for effective detection of cyber attack resources, passive and active monitoring should be a series of continuous actions in coordination.

One more important point we should care in today's security research is that the active monitoring should not be a noise on Internet. For example, rather than scanning the entire Internet, the scan should be on purpose for tracing a particular malicious activity. To fulfill such situation, coordination of passive and active monitoring plays in an important role. Namely, we can detect malicious activity by passive monitoring initially and then perform active monitoring based on knowledge gained by passive monitoring.

3.4. Coordination of Passive and Active Monitoring

Figure 3 shows framework for detection of cyber attack resources by coordinated passive and active monitoring. Firstly, new threat in Internet can be detected by operational passive monitoring techniques (P), such as darknet or passive DNS traffic. From this, we can extract out suspicious behaviors together with the suspicious attack resources such as IP addresses.

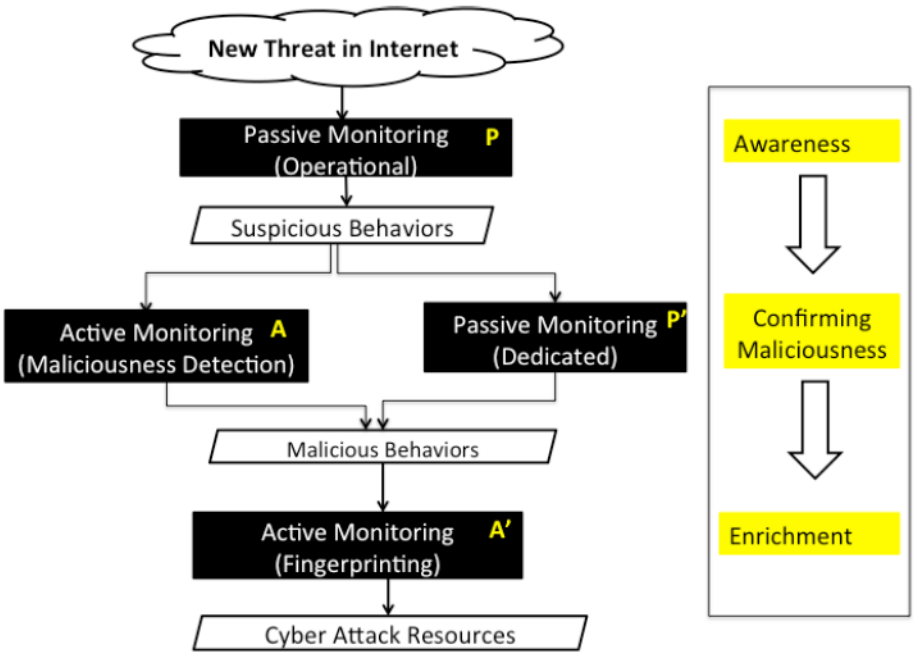


Figure 3 - Framework for coordination of passive and active monitoring

This first stage is the awareness stage for the detection of new cyber threat. In the second stage, we confirm maliciousness with active monitoring for malicious detection (A), or dedicated passive monitoring (P') or both in combined. From this, malicious behaviors can be confirmed for the detection of malicious attack resources. This stage is confirming maliciousness. In the third stage, we enrich information of already confirmed malicious cyber attack resources. For example, we can search what type of device from which country or which ISP by active monitoring for fingerprinting purpose (A'), such as banner grabbing, OS fingerprinting or location checking. We call this final stage as enrichment. In stage two, as we already confirmed maliciousness of attack resources, we can enrich information of these malicious cyber attack resources by monitoring techniques (A'), in stage three. With this way, we can reduce unnecessary traffic load of blinded network scans on Internet. Using this framework, in this study, we propose two novel methods for the detection of cyber attack resources. Using introduced framework in Figure 3, two novel detection methods are proposed in this study at Chapter 6 and 7.

In method one, we try to detect malicious authoritative name servers and malicious domains by coordination of P, A and A' as show in blue line in Figure 4. Firstly, we analyze passive DNS traffic, which is the operational passive monitoring data set and try to extract out suspicious behaviors from it. For example, we look at how domain names are being resolved to how many number of IP addresses and respective authoritative name servers of a domain, etc. As a result, we could extract out suspicious behaviors and resources such as domain and IP relating to these behaviors. Then, in the second stage, based on extracted behaviors, we keep on doing the DNS profiling and confirm maliciousness. For example, for a particular domain, we keep on watching how corresponding IP addresses are changing and confirm maliciousness. As a result, we got malicious behaviors and theirs corresponding resources such as domain and IP addresses. Finally, at the information enriching stage, we make location checking of the malicious resources. With this way, by coordination of PAA' effectively, we try to detect malicious authoritative name servers and domains. The detail explanation of method one is in chapter 6.

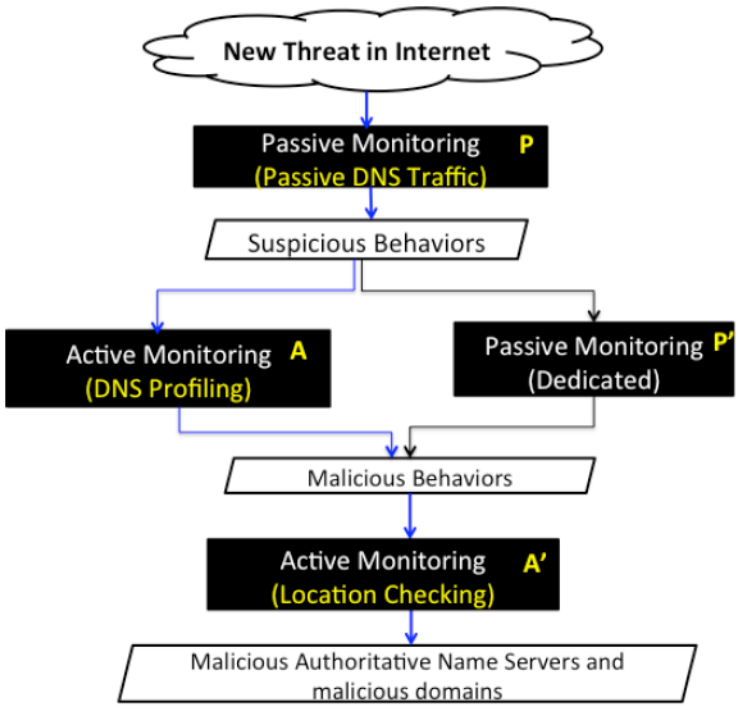


Figure 4 - First detection method

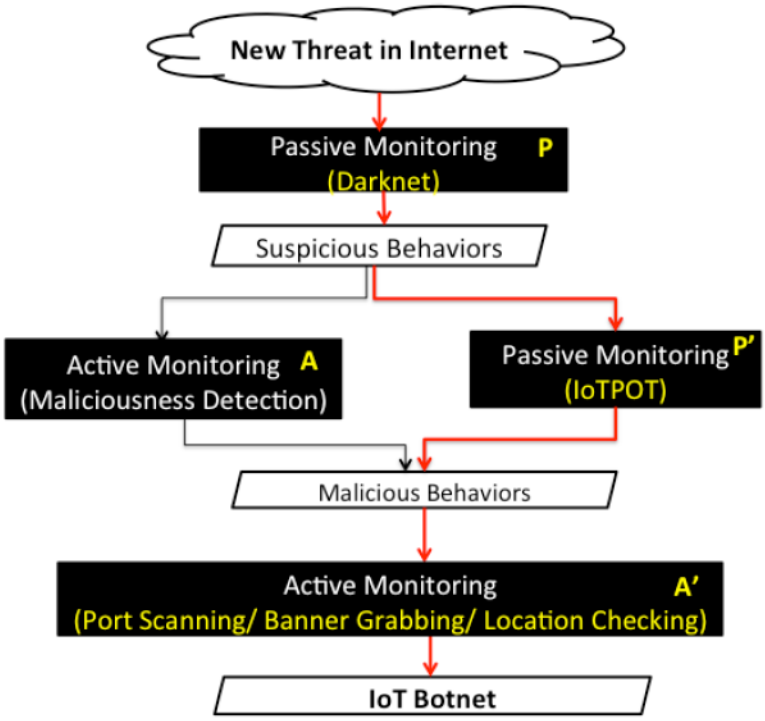


Figure 5 - Second detection method

In method two, we try to detect IoT botnet by coordination of P P' and A' as show in red line in Figure 5. Firstly, we keep on watching darknet traffic which is the operational passive monitoring data set. As results, we could extract out suspicious behaviors and we become to know that telnet scans are dramatically increasing comparing with other protocols. Together with this behavior, we know resources of Telnet scans. Continuously, in the second stage, we try to set up a honeypot, IoTPOT, in order to confirm the maliciousness. As results, we confirm malicious behaviors and find resources such as IP addresses relating these behaviors. Finally, in order to enrich the information of these malicious attack resources, we make port scanning, banner grabbing and location checking. With this way, by coordination of PP' and A' effectively, we try to detect IoT Botnet. The detail explanation of method two is given in chapter 7.

Chapter 4
Related Works

4.1. Related Studies for detection of malicious authoritative name servers

There are previous research efforts in finding malicious domains using passive DNS data, zone files or DNS whois database. In contrast with previous studies, we are not just focusing on finding malicious domains. We take a further step into understanding of how attackers are abusing authoritative name servers to manage their malicious domains. Based on this understanding, we try to detect not only malicious domains but also malicious authoritative name servers. We call domains and authoritative name servers that are relating to malicious online activities as malicious domains and malicious authoritative name servers, respectively. There may be variety of cases how authoritative name servers are prepared by attackers, such as setting up a dedicated server as malicious authoritative name server or abusing a legitimate server for malicious purposes, however, we do not differentiate them and consider both cases malicious in this study.

Antonakakis et al. developed a reputation based classification system called Notos [32] in which domains were reputed based on network based, zone based and evidence based. Bilge et al. designed EXPOSURE [33] in which behaviors of domains were analyzed focusing mainly on time series of domains being queried together with other features such as DNS answers based, TTL value based and domain name pattern based features. In both studies, only the mapping between d and respective IP was considered and ns-d was not considered.

Hao et al. [34] studied behavior of spam domains combining with active DNS behavior and registration information. Although they found that IP spaces

used by spam domains were small, how d, ns-d and IP were related was not studied.

Hu et al. [35] studied active detection of fast-flux domain in which IP usage of fast flux domains were analyzed. They found IP overlap between fast flux domain and their authoritative name server. This finding is similar to feature three of our method although we are not focusing on detection of fast flux domains only. In comparing with previous studies, contribution of the proposed method is two-fold: (1) it can detect unknown malicious domains, name servers' domains, and their corresponding IP addresses that are not in existing blacklists, (2) it uses data that is publicly accessible and easy to obtain by a single DNS resolver while the existing methods rely on additional data that is available for certain entities such as a long period of historical data of domains and IPs, DNS traffic captured at large networks such as ISP, and DNS responses obtained by a large number of resolvers in different locations (continents).

4.2. Related Studies for detection of IoT botnet

We implemented the first honeypot tailored for IoT devices, IoTPOT, and to the best of our knowledge, there is still no honeypot like IoTPOT that mimics IoT devices of many different CPU architectures while listening on 23/TCP with the ability to learn unknown command interactions. Although Honeyd [36] listens on 23/TCP, it is a low-interaction honeypot and cannot handle not only Telnet options but also command interactions interactively, as explained in Sect. 3.4.2. Although there is another honeypot known as Telnet password honeypot [37], its main focus is collecting Telnet password and command interactions are not supported. Other popular low interaction honeypots such as Dionaea [38] and Nepenthes [39] do not support Telnet. Kishimoto et al. [40] proposes a novel honeypot that dynamically assigns IPv6 address to appropriate high interaction

honeypots by checking the destination IP address of incoming NS message which includes vendor information. SGNET [41] is a honeypot system that have distributed low-interaction sensors to handle known attacks and centralized backend high-interaction honeypots to handle unknown attacks redirected from the distributed sensors. The conceptual mechanism of IoTPOT is similar to SGNET and the IPv6 honeypot mentioned above. As in SGNET, *Frontend Responder* of IoTPOT responds known attacks and unknown attacks are redirected to IoTBOX. As in the IPv6 honeypot, it tries to deal with different hosts and devices. The main difference between IoTPOT and these existing honeypots is that IoTPOT implements functionality to perform automated active scanning to the attacking IP addresses to learn their interactions, namely banner profiles. With this functionality, we can obtain and enrich profiles for presumably vulnerable and infected devices, which is essential for monitoring diverse IoT threats. In other words, IoTPOT learns the banners from vulnerable devices to pretend to be themselves. Moreover, as an initial goal, we highly focused on Telnet attacks which are emerging threats according to the recent observations of darknet as explained in Sect. 2, emulate the Telnet services of large variety of IoT devices to attract attacks, and succeeded to observe the ongoing attacks to the depth of capturing the malware binaries, which are hardly included in a large malware database like Virus Total. In order to analyze the captured malware binaries, we also implemented IoTBOX, the first sandbox that handle to run malware of 8 different CPU architectures. Out of more than 15 surveyed sandbox systems in [32], none support different CPU architecture such as MIPS, ARM.

Main differences of proposed method against existing works are as follow:

- IoTPOT implements functionality to perform automated active scanning to the attacking IP addresses to obtain their banner profiles. With this functionality, we can obtain and enrich profiles for presumably vulnerable

and infected devices, which is essential for monitoring diverse IoT threats. In other words, IoTPOT "learns" the banners from vulnerable devices to pretend to be themselves.

- Although mechanism is similar to existing honeypots, we are the first to focus on Telnet-based honeypot that can handle banner interactions, authentication interactions and command interactions till the depth of attacks where actual malware binaries can be captured for detailed analysis.
- We propose IoTBOX, a multi-architecture malware sandbox that is used as high interaction system as a component of IoTPOT and also independently used as malware sandbox for analyzing captured binaries.
- We succeeded to report for the first time about details of currently menacing IoT threats targeting vulnerable IoT devices over the world while capturing IoT malware that are hardly included in existing malware database of Virus Total. We also reveal their monetization behaviors and architectures as botnet.

4.3. Related Studies for coordination of passive and active monitoring

In order to detect cyber attack research resources, previous cyber security researches heavily stress one or more data sets of either of active or passive monitoring approaches [42] [43]. In this study, we try to coordinate passive and active monitoring approaches for detection of cyber attack resources. Thus, to the best of our knowledge, we think that we are first in introducing the idea on coordination of passive monitoring and active monitoring for the detection of cyber attack resources efficiently.

Chapter 5

Finding Malicious Authoritative DNS Servers by DNS traffic

Introduction

In this chapter, we explain about our initial studies on ISP DNS traffic in order to understand the behaviors of malicious authoritative name servers. By this study, we could grasp four features of malicious authoritative name servers and propose a method to detect malicious authoritative name server based on these features. From this preliminary study by passive DNS traffic, we find out that out of all features, domain flux size feature is quite strong for detection of malicious authoritative name servers. Thus, more specific and carefully categorized features of domain flux size feature are studied and propose a comprehensive detection method explained in Chapter 6.

5.1. Features for detecting Malicious Authoritative DNS servers

5.1.1. Feature 1: Fraction of blacklisted domains

In the first feature, the fraction of blacklisted domains for which the evaluated DNS server is authoritative is calculated for its evaluation. The matching can be done with existing blacklists such as EXPOSURE [33], Zeus Tracker [44] , and Malware domain list [45] and Spybot domains of our dynamic malware analysis. However, the coverage of these blacklists is limited and we can miss some malicious DNS servers. In our experiment described in the next chapter, we extend the blacklists by considering all domains sharing the same IP address with a blacklisted domain as black. The simplest way to apply this feature for detecting

malicious DNS servers is adopting a threshold. Namely, we can determine that a DNS server is malicious if the fraction of blacklisted domains that the server is authoritative for exceeds the threshold. In the experiment, we set the threshold to 0.9

5.1.2. Feature 2: Server Fail Response History

The DNS servers of the popular and benign domains are normally very stable. In fact, Server Fail response error is rarely found in our study of authoritative DNS servers hosting popular top 1000 domains of Alexa list. In contrast, in fast flux network, normal malware infected PC can be used as proxy to redirect to actual DNS servers. In such case, the quality of service of DNS server cannot be as high as real DNS servers because the PC may be shut down by its user and server fail errors can be occurred. That is why we focus on the history of DNS Server Fail error response for evaluating DNS servers. At this moment, we have not determined how exactly we are going to use this feature for detecting malicious DNS servers.

5.1.3. Feature 3: TTL of DNS Server's Domain Name

Time to Live (TTL) value of the DNS server's domain is also an important factor of differentiating malicious DNS servers. When a cache (recursive) DNS server queries the authoritative DNS server for a resource record, it will cache that record for the time in seconds specified by the TTL. The A records of malicious DNS server involving in fast flux service network change rapidly. That is why, the TTL for each A resource record is set to very low value such as a few seconds Again the simplest way to apply this feature for detection of malicious DNS servers is to adopt a threshold. Namely, if a DNS server has a domain name whose

TTL value is smaller than the threshold value, we determine that the server is malicious.

5.1.4. Feature 4: Domain Flux

In this feature, we check the existence of domain flux in each of DNS server. For finding domain flux, we count the number of domains sharing the same IP address. If the number exceeds a threshold, then we consider there is a domain flux. In the experiment of the next chapter, we set the threshold as 100.

5.2. Experiment

The experiment for the evaluation of the four features is done using operational traffic of a cache DNS server. The process of the experiment is shown in Figure 6.

In the first step, we extract domains from the DNS reply packets of the analyzed traffic of the cache DNS server. The data used for the evaluation of the proposed method is 65-minute-long DNS traffic captured between a cache DNS server and its clients of approximately 1 to 2 million. There are 3 to 4 million domains resolved in the traffic.

In the second step, we filter out certain domains by three filtering rules. Firstly, domains relating to security software and domains used for DNS blacklist check and the reverse lookup domains are filtered out. Secondly, the domains matching with top 1,000,000 popular domains of Alexa domain list are filtered out. Thirdly, the domains that do not have proper domain format as described in RFC 1035 [46] are dropped.

In the third step, the authoritative DNS servers of each domain are looked for. The resolver program built on Perl Net::DNS::Resolver module is used for this

step. In this step, for each of investigated domains, NS, A, SOA RRs are queried programmatically to receive a list of authoritative DNS servers.

In the fourth step, the analysis on the outputs of the third step is conducted. The database of DNS servers and their domains are reconstructed based on the outputs of the third step. In the fifth step, the four features described in the previous chapter are evaluated.

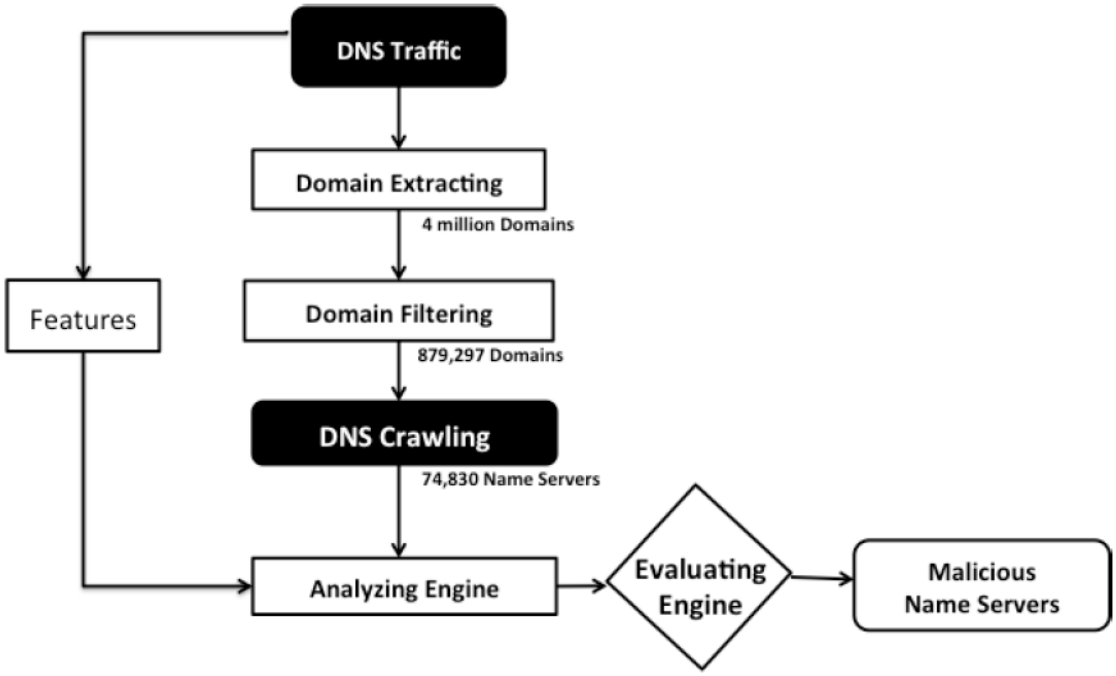


Figure 6 - The flow of experiment

5.2.1. Feature 1 (Fraction of blacklisted domains)

Firstly, the total of 111,883 known black domains are collected from EXPOSURE [33] , Zeus Tracker [44] , Malware domain list [47] and Spybot domains observed by our malware sandbox analysis. Then, we extended the blacklist by considering all domains sharing the same IP address as a blacklisted domain as black.

In order to extend the blacklist, A records associated with the domains of each of the DNS servers are queried by the resolver script based on Net:DNS:Resolver. Then, for each DNS server, domains with the same A records

are clustered. Each of the clustered groups is matched again with known blacklisted domains. If one domain of the cluster matches with a known black domain, the other domains in each cluster are considered as extended black domains.

Finally, the fraction of black domains is calculated for evaluating each DNS server. In the experiment, we determine that an authoritative DNS server with more than 90% of its observed domains blacklisted is a malicious one although the threshold should be discussed further.

5.2.2. Feature 2 (Server Fail History)

We evaluate each DNS server by checking whether any client has received Server Fail response when querying for an authoritative answer to it.

5.2.3. Feature 3 (TTL of DNS server's domain name)

We evaluate each DNS server by the TTL value of its domain name. The domain name of a DNS server can be obtained by using dig command with trace option. The automated trace route queries to A records of the DNS servers' domain names are investigated in this feature.

5.2.4. Feature 4 (Domain Flux)

The experiment is conducted on 74,830 name servers. The domains for which each of the DNS servers is authoritative are first clustered by their corresponding IP addresses. Then, we extract the clusters with a domain flux using a threshold of flux domains of 100.

5.3. Results and Discussions

From the cache DNS server traffic described above, approximately 20 to 30 million DNS response packets are extracted. From these response packets, 4 million domains are extracted. In the second step, after applying three filtering rules to the extracted domains, the remaining domain is 879,297. In the third step, authoritative DNS servers of each of the domains are looked for. As a result of the third step, we found 74,830 authoritative answers for 294,059 domains. Other domains receive errors like NXDomain and ServFail. In the fourth step, the analysis on these DNS servers is conducted. The database for 74,830 DNS servers and their respective domains are constructed in this step.

As the first feature of evaluation engine, DNS servers hosting black domains are investigated. From this analysis, 430 DNS servers, for which at least one of their domain names is blacklisted, are found. Out of 430 DNS servers, 31 DNS servers are found with 90% of their domains blacklisted. The list of these DNS servers and the percentage are shown in the Table 2. In addition, out of the 430 DNS servers, 22 are listed on KnujOn [48] as the top 20 spam domain hosting DNS servers.

As the analysis result of the second features, we confirm that 60% of the 31 DNS servers found in the previous analysis have server fail history of at least one time.

As for the third feature in which TTLs are investigated, 40 DNS servers have very low TTL values ranging from zero to 5 minutes. Out of these 40 servers, 15 DNS servers have very low TTL value of zero to 100 seconds. These DNS servers and their TTL values are shown in Table 3.

We check on web in order to know whether these 15 DNS servers are concerning with malicious online activities or not. In report for spam domains of KnujOn, dns01.gpn.register.com is reported as DNS server serving many

spamming domains. In addition, at malwareurl.com [49] dns01.gpn.register.com to dns05.gpn.register.com are reported as DNS servers hosting 129 malicious domains relating with 8 different types of malware, click fraud and exploits. The analysis result on each of the DNS server's domains name based on the information on web is shown in column 3,4 and 5 of Table 2. Finally, 9 out of 15 DNS servers are confirmed as DNS servers relating with malicious online activities.

As for the fourth feature, by analyzing 74,830 DNS servers, we found 85 servers with at least one flux of more than 100 domains. We found a DNS server with as many as 145 fluxes. Out of the 85 servers, 13 are found on web reports as worst name servers of this year hosting spam domains, illicit Pharmacies domains and malware domains. In addition, 22 name servers out of the 85 are hosting at least one known black domain derived in the experiment for the first feature.

DNS Servers with high % of black domains

Table 2 - DNS Servers with high % of black domains

Name Server's domain	Known Black	Existing Domain	Extended Black	% of black
ns1.pulsarserve.net	1	2	2	100
ns1.salenames.ru	1	14	14	100
ns2.ndoverdrive.com	2	17	17	100
ns2.pulsarserve.net	1	2	2	100
ns2.salenames.ru	1	14	14	100
ns37.coopertino.org	1	2	2	100
ns38.coopertino.org	1	2	2	100
ns5.no.cg.shawcable.net	1	3	3	100
ns6.so.cg.shawcable.net	1	3	3	100
sk.s2.ns1.ns92.kolmic.com	1	466	466	100
sk.s2.ns2.ns92.kolmic.com	1	466	466	100
ns1.namebrightdns.com	2	391	384	98.2097187
ns2.namebrightdns.com	2	391	384	98.2097187
ns1.dsredirection.com	44	1520	1485	97.6973684
ns3.domainingdepot.com	1	43	42	97.6744186
ns4.domainingdepot.com	1	43	42	97.6744186
ns2.dsredirection.com	44	1520	1481	97.4342105
ns.counter.co.kr	1	31	30	96.7741935
ns.induce.com	1	31	30	96.7741935
sell.internettraffic.com	32	1239	1197	96.6101695
buy.internettraffic.com	32	1239	1198	96.6908797
ns1.csof.net	7	23	22	95.6521739
ns2.csof.net	7	23	22	95.6521739
ns1.wordpress.com	49	570	541	94.9122807
ns1.parkingcrew.net	3	208	197	94.7115385
ns2.parkingcrew.net	3	208	197	94.7115385
ns2.bodis.com	10	414	389	93.9613527
ns1.bodis.com	9	413	388	93.9467312
ns1.dnslink.com	2	260	242	93.0769231
ns2.dnslink.com	2	260	242	93.0769231
ns2.wordpress.com	49	570	520	91.2280702

Table 3 – DNS servers involving with flux-flux

No	Domain Name of DNS Server	TTL in Sec	Report on Web	Malicious Domain in Report	Detail
1	dns01.gpn.register.com.	60	Malwareurl.com/KnujOn	129/many	Malware
2	dns02.gpn.register.com.	60	Malwareurl.com	129/many	Exploits
3	dns03.gpn.register.com.	60	Malwareurl.com	129/many	Click fraud
4	dns04.gpn.register.com.	60	Malwareurl.com	129/many	Spam
5	dns05.gpn.register.com.	60	Malwareurl.com	129/many	
6	dns082.d.register.com.	60	No Report		
7	dns1.wavenet.com.ar.	0	No Report		
8	dns151.a.register.com.	60	Malwareurl.com	1	Malware
9	dns159.c.register.com.	60	Malwareurl.com	1	Malware
10	dns164.b.register.com.	60	No Report		
11	ns.induce.com.	60	No Report		
12	ns1.h69.hvosting.ua.	60	No Report		
13	ns1.hidc.co.kr.	100	Malwareurl.com	10	Malware
14	ns2.h69.hvosting.ua.	60	No Report		
15	ns2.hidc.co.kr.	100	Malwareurl.com	10	Malware

5.4. Conclusion

This study proposes four features for finding malicious authoritative DNS servers. We evaluate the four features using real traffic of cache DNS servers. Our future works include a proposal of comprehensive detection method using the proposed features as well as deriving proper parameters for each feature.

Chapter 6

Detecting Malicious Domains and Authoritative Name Servers Based on Their Distinct Mappings to IP Addresses

Introduction

In this chapter, coordination of passive DNS monitoring with active DNS crawling to detect malicious domains and malicious authoritative name server is explained. We present a novel method for detecting malicious "domains" (noted as d) and malicious "authoritative name servers" (noted as ns-d) based on their distinct mappings to "IP addresses" (noted as IP). Namely, we present three distinct features to detect them; 1) Single ns-d is mapped to many IP, 2) Single IP is mapped to many ns-d, and 3) Single IP is mapped to both ns-d and d. All these three features are more carefully categorized features of domain flux size features explained in Chapter 5.

We evaluate the proposed method in terms of accuracy and coverage in detection of malicious d and ns-d. The evaluation shows that our detection method can achieve significantly low false positive rate in detecting both malicious d and ns-d without relying on any previous knowledge, such as blacklists or whitelists.

6.1. Features

6.1.1. Mappings of d, ns-d and Respective IP

We first explain mappings of d, ns-d and their respective IP with real data example of "google.com" domain. In Figure 7, google.com is d and

ns1.google.com, ns2.google.com, etc., are ns-d. Both google.com and ns1.google.com have respective IP.

In the same way, for a particular d, it may have one or more corresponding ns-d. Both ns-d and d will have corresponding IP. In more detail, IP of ns-d is the IP address of a server running authoritative DNS service and IP of d may be the IP address of the server running other Internet service such as web.

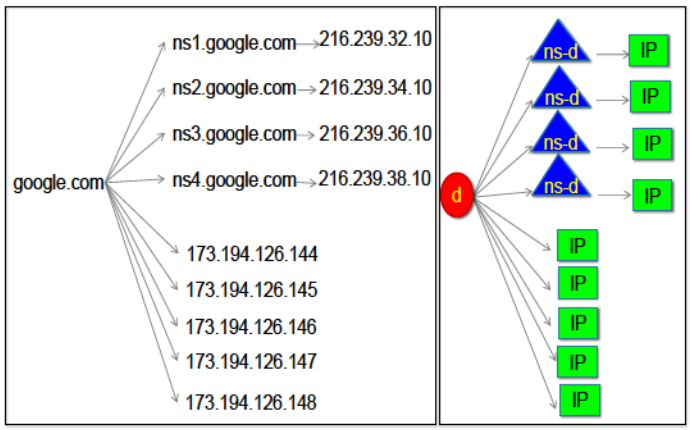


Figure 7 - Mapping of d, ns-d and IP

6.1.2. Feature One: single ns-d is mapped to many IP

As authoritative name server needs reliability for proper zone operation, IP of ns-d should not be changed frequently. On the other hand, attackers try to hide their authoritative name server by changing IP of ns-d. IP fluxing with IP of ns-d is a sign that ns-d is suspicious. Thus if a single ns-d is mapped to more than *Th1* IP addresses, we consider the mappings as a malicious case. The comparison between normal case and malicious case is shown in Figure 8.

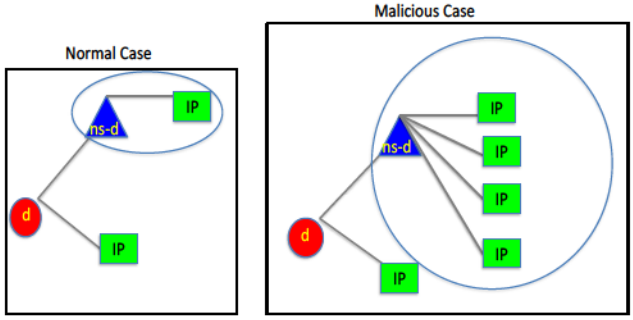


Figure 8 - Feature one

6.1.3. Feature

Two: single IP is mapped to many ns-d

Normally, different ns-d resolves to separate IP. For example, ns1.example.com and ns2.example.com resolve to separate IP. If many different ns-d resolve to single IP we consider the mappings as malicious case. Attacker with limited IP resources can take advantage in controlling his malicious domains with this feature. He can also hide his malicious authoritative name server by setting separate ns-d for each malicious domain. For example, in registering malicious domains, attacker can setup to resolve malicious-1.com, malicious-2.com and malicious-3.com to ns.malicious-1.com, ns.malicious-2.com and ns.malicious-3.com respectively rather than resolving all malicious domains to a particular ns-d. In this way, if one hundred malicious d are registered, there will be one hundred different ns-d. All these ns-d are again setup to resolve to a single IP managed by the attacker so that he can manage all his ns-d with a single IP or a set of IP. That is why, in feature two, if single IP is mapped to more than Th_2 ns-d, we consider the mappings as malicious case. The comparison between normal case and malicious case is shown in Figure 9.

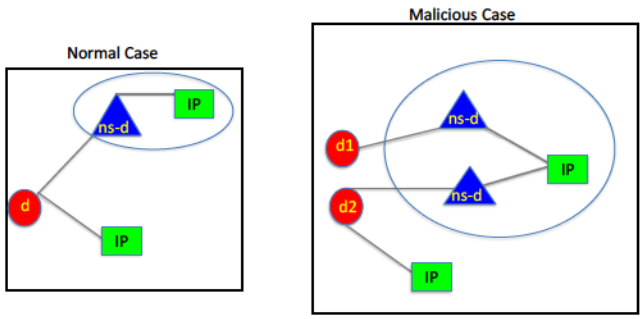


Figure 9 - Feature two

6.1.4. Feature Three: single IP is mapped to both ns-d and d

This feature is based on our finding that ns-d and d share the same IP. That is, DNS services and other malicious services, run in the same server. In the case

of virtual hosting, one IP may be shared by many web sites. But it is practically very rare to share one IP with both DNS service and other service such as web.

As it is technically possible to run both web service and DNS service in the same server, a benign user of small business may install both services in the same server. In such case, the number of ns-d and d sharing the same IP should not be high. Therefore, if the total number of ns-d and d sharing the same IP is more than Th_3, we consider this as a malicious case. The comparison between a normal case and a malicious case is shown in Figure 10.

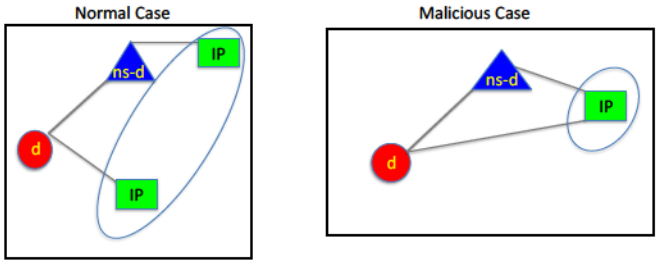


Figure 10 - Feature three

6.2. Approach

6.2.1. The Proposed Method

We propose a method for detecting malicious d and ns-d based on their distinct mappings to IP addresses. Namely, we present three distinct features to detect them; 1) Single ns-d is mapped to many IP, 2) Single IP is mapped to many ns-d, and 3) Single IP is mapped to both ns-d and d. An overview of the proposed method is shown in Figure 11.

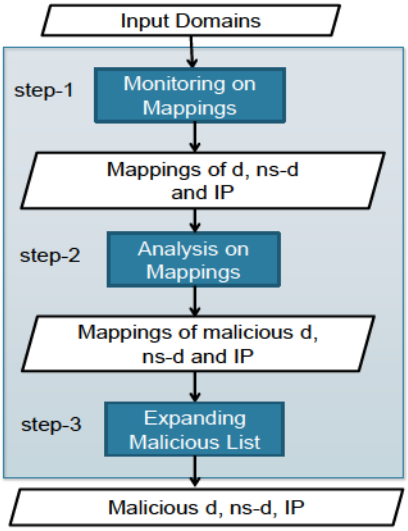


Figure 11 - Overview of proposed method

The proposed method consists of three main steps: monitoring on mappings, analysis on mappings and expanding the malicious list. The input is a set of domains that are not known to be benign or malicious. Step one is monitoring on mappings of d, ns-d and IP. Step two is an important part in which we extract distinct mappings of malicious d, ns-d, and IP using all three features we proposed. In step three, we expand the malicious list and receive a list of malicious d, ns-d and IP as final output. Detail explanations of the three steps are described in the following sections. Analysis procedures and outcomes in each step of the proposed method are shown in Figure 12.

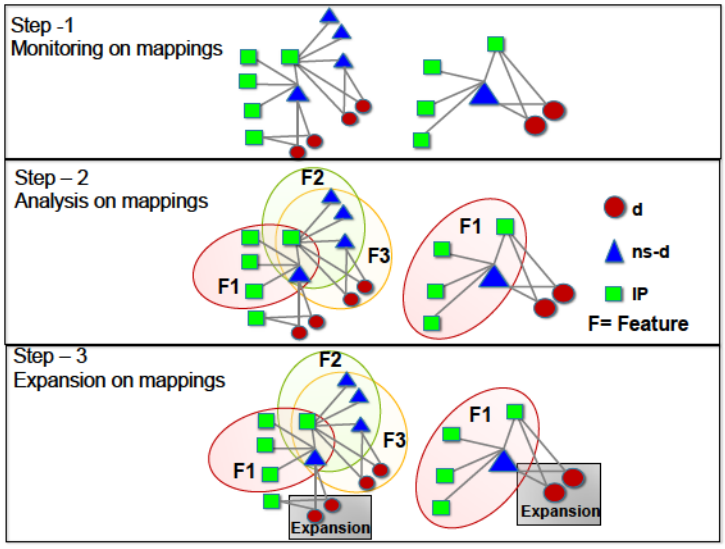


Figure 12 - Analysis procedure in each step of the proposed method

6.2.2. Step One: Monitoring on Mappings

For every input d, we find 1) ns-d of d, 2) IP of ns-d, and 3) IP of d. Figure 13 shows the process of finding mappings between d and ns-d, ns-d and IP and, d and IP.

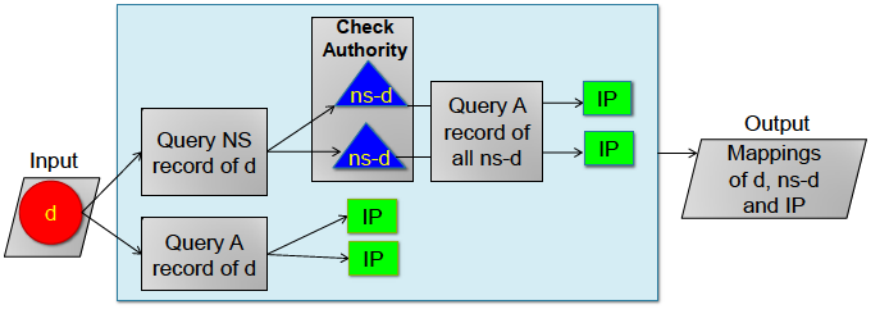


Figure 13 - Finding the mappings

In order to find ns-d of d, we simply query NS RR (Name Server Resource Record) of d. For example, we query NS RR of google.com so that we can get reply as “ns1.google.com” which is ns-d of google.com.

To look for IP of ns-d, we query A RR (IPv4 Address Resource Record) of ns-d. For example, we query A RR of ns1.google.com so that we can get reply “216.239.32.10” which is IP of ns1.google.com. After knowing ns-d and IP of ns-d, we check whether ns-d is really authoritative name server of d or not. For this, we query SOA RR (Start Of Authority Resource Record) of d at ns-d and check reply packet whether aa (authoritative answer) bit is set or not. Only if aa bit is set in reply packet from ns-d, we assume that ns-d as authoritative name server of d.

Finally, to find the corresponding IP of d, we make A RR query of d. For example, we query A RR of google.com so that we can get a reply as “173.194.126.144” which can be one of the web servers of google.com domain.

For all queries, we use our recursive DNS server that query recursively to different levels of name servers in the DNS hierarchy till it reaches a final authoritative name server. For example, while querying A RR of d, our recursive DNS server talks directly to different levels of referral name servers in the DNS hierarchy starting from root servers till it reaches a final authoritative name server in which the corresponding IP of the queried domain is recorded in its zone file. We also set UDP (User Datagram Protocol) time out of queries to 1 second so that our resolver cannot be highly loaded. After finding all mappings of d, we obtain mappings between d and ns-d, ns-d and IP and, d and IP.

Step one is supposed to be continued for some period in order to obtain mappings of d, ns-d, and IP to be examined. In the experiment, we use the mappings obtained from the monitoring period of 214 days.

6.2.3. Step Two: Analysis on Mappings

Mappings obtained by step one are analyzed based on the following three features:

- Single ns-d is mapped to many IP
- Single IP is mapped to many ns-d
- Single IP is mapped to both ns-d and d

The details of features are explained in section 4. We depict the typical structure of features in Figure 14.

Firstly, we check the obtained mappings to see whether any of the three features is met. All three features have separate threshold values (noted as Th_1, Th_2, Th_3). Mappings exceeding threshold values will be considered malicious.

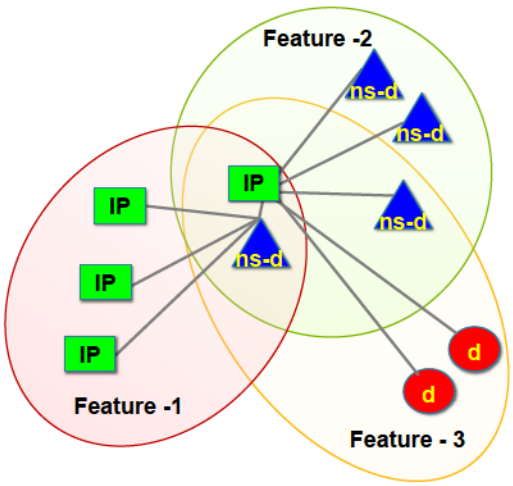


Figure 14 - Typical structure of all three features

Indeed, in order to increase the accuracy of detection, we consider features in combined manners as shown in column 1 and 2 of Table 4. For example, for F1^F2 combination, we look for mappings between ns-d and IP that meet both feature one and two.

Table 4 - Different Combinations of Features

Combining three features	Combining two features	Separate Features
F1 ∧ F2 ∧ F3	F1 ∨ F2	F1 only
(F1 ∧ F2) ∨ F3	F1 ∨ F3	F2 only
F1 ∧ (F2 ∨ F3)	F2 ∨ F3	F3 only
(F1 ∨ F2) ∧ F3	F1 ∧ F2	
F1 ∨ (F2 ∧ F3)	F1 ∧ F3	
F1 ∨ F2 ∨ F3	F2 ∧ F3	
(F1 ∧ F3) ∨ F2		
(F1 ∨ F3) ∧ F2		

In general, feature one and two are mappings between ns-d and IP and only feature three is mappings of d and ns-d to IP. That is why only some combinations that has OR operation with feature three will consist of d in the result. For example, the result of "F1^F2^F3" combination will contain only ns-d and IP while the result of "F1vF2vF3" combination will include not only ns-d and IP but also d. Output of step two will be the mappings of d, ns-d and IP that meet the combined features in Table 4. We consider all these d, ns-d, and IP of output as malicious.

6.2.4. Step Three: Expanding Malicious List

In step three, for each combination of features, we expand malicious d, respectively. Namely, we consider malicious for all d that are mapped to any of the ns-d or IP that construct malicious mappings identified in step two. Finally, we obtain lists of malicious d, ns-d and IP for each combination of the features.

6.3. Evaluation

6.3.1. Experiment and Results

6.3.1.1. Input Data Set

We collect and combine existing blacklist and whitelist to use it as input to the proposed method. Firstly, as known blacklist, we use malicious domains from

DNS-BH project malwaredomains.com [50] . The total number of malicious domains we could collect within the whole analyzing period is 34,849 domains. Secondly, as known whitelist, we use top 10,000 domains from Alexa domains list [51]. The total number of benign domain we could collect within the whole analyzing period is 15,181 domains. In total, there are 50,030 domains as an input to the proposed method.

6.3.1.2. Step One: Monitoring on Mappings

The monitoring period is from April 1, 2014 to October 28, 2014. Within the whole period, we keep on monitoring all mappings between "d and ns-d", "ns-d and IP" and "d and IP". Table 5 shows number of d, ns-d and respective IP we are able to find in step one.

Table 5 - Numbers of d, ns-d and respective IP

	d		ns-d		IP of d		IP of ns-d	
	Benign	Malicious	Benign	Malicious	Benign	Malicious	Benign	Malicious
	15,101	22,735	18,384	16,543	31,041	25,736	17,721	11,162
Unique total	37,836		32,280		54,754		25,657	

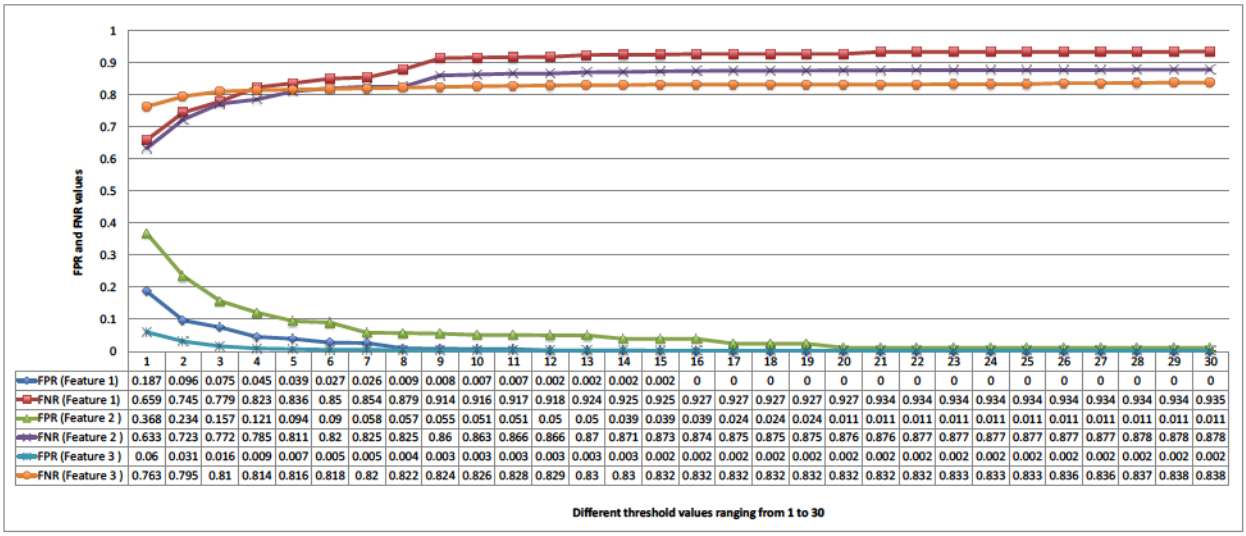
We could only find mappings of 75% of input d. The main problem is because of NXDomain (Non Existence of domain). It is because of the short lifetime of malicious domains. Out of all input d, 17% of d becomes NXDomain in time of query. The rest 8% encounters errors such as ServFail (Server Fail), NoError (No Error), Refused (Query Refuse) and UDP query time out error. ServFail can be because of some failure in DNS service of authoritative name server. Although NoError literally means no error, we did not get any answer back for the query. It is because the RR type of d we are querying is not implemented although other RR type of d exist. For example, in querying NS RR of www.example.com, NS RR type of www.example.com does not exist although A RR type of www.example.com and NS RR of example.com both exist. In such case we receive NoError reply with no answer. Refuse error simply means that our

query is refused. UDP timeout error is because of DNS query exceeding UDP timeout time.

6.3.1.3. Step Two: Analysis on Mappings

In this step, all mappings that meet any of the proposed three features are extracted as distinct mappings of a malicious case. We set value of Th_1, Th_2 *and* Th_3 to "three" as constant threshold value for all features because we would like to compare the strength of each feature and we think that 3 should be the smallest threshold value for detecting malicious domains and authoritative name servers according to many initial studies on malicious and benign domains.

An additional experiment on many different threshold values is conducted. By comparing the FPR and FNR values of different threshold values ranging from 1 to 30 as shown in Figure 15, we would like to recommend 8 as the best threshold value for all features while FPR is as low as 0.004 and FNR is less than 0.9.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
FPR (Feature 1)	0.187	0.096	0.075	0.045	0.039	0.027	0.026	0.009	0.008	0.007	0.007	0.002	0.002	0.002	0.002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FNR (Feature 1)	0.659	0.745	0.779	0.823	0.836	0.85	0.854	0.879	0.914	0.916	0.917	0.918	0.924	0.925	0.925	0.927	0.927	0.927	0.927	0.927	0.934	0.934	0.934	0.934	0.934	0.934	0.934	0.934	0.934	0.935
FPR (Feature 2)	0.368	0.234	0.157	0.121	0.094	0.09	0.058	0.057	0.055	0.051	0.051	0.05	0.05	0.039	0.039	0.039	0.024	0.024	0.024	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011
FNR (Feature 2)	0.633	0.723	0.772	0.785	0.811	0.82	0.825	0.825	0.86	0.863	0.866	0.866	0.87	0.871	0.873	0.874	0.875	0.875	0.875	0.876	0.876	0.877	0.877	0.877	0.877	0.877	0.877	0.878	0.878	0.878
FPR (Feature 3)	0.06	0.031	0.016	0.009	0.007	0.005	0.005	0.004	0.003	0.003	0.003	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
FNR (Feature 3)	0.763	0.795	0.81	0.814	0.816	0.818	0.82	0.822	0.824	0.826	0.828	0.829	0.83	0.83	0.832	0.832	0.832	0.832	0.832	0.832	0.832	0.832	0.833	0.833	0.833	0.836	0.836	0.837	0.838	0.838

Figure 15 - FPR and FNR of different threshold values

In our current experiment, to analyze data by feature one, we check all mappings between ns-d and IP. Then, we extract distinct mappings of a malicious case according to Th_1. As a result, in all mappings that meet feature one, there are 5,340 ns-d and 3,081 IP. In an extreme case, we found ns-d named

"ns2.alfacoma.ru" (colored yellow in Figure 16) that has 200 corresponding IP. We believe that these 200 IP can be IP of compromised hosts. All mappings extracted by feature one are visualized using force-directed graph drawing algorithm. Figure 16 shows one example of mappings with 181 ns-d and 1,479 IP.

In order to analyze data by feature two, again, we check all mappings between ns-d and IP. But, this time, the analysis is focused on IP. For example, according to Th_2, if an IP has more than three corresponding mappings to ns-d, we think of it as a malicious case. As a result, there are 1,908 IP and 9,088 ns-d in all mappings that meet feature two. In an extreme case, to our surprise, we find a single IP related to 2,925 ns-d that are quite similar to each other such as ns1.com-fn41.net, ns1.com-fn62.net, ns1.com-fo30.net, etc. Some of the mappings that meet feature two exhibit similar structure when these are visualized. Figure 17 shows two mappings, both of which consist of exactly 7 IP and 560 ns-d. Although their relational structure is very similar, their actual ns-d and IP are different. This may be an indication of the usage of the same administrative tool for these d and ns-d although a deeper investigation is necessary.

To find mappings that meet feature three, we extract all mappings in which one IP is shared by both ns-d and IP. Then, for each detected mapping, it is checked whether the number of ns-d and d exceeds the threshold Th_3. As a result, there are 3,438 d, 5,477 ns-d, and 522 IP in all mappings that meet feature three. In an extreme case, we notice a single IP shared by 2,892 ns-d and 651 d.

An example of mappings that meet feature three is shown in Figure 18 consisting of 1,444 d, 1,420 ns-d and 70 IP. According to Figure 18, we think that attackers are controlling a large number of d and ns-d with a limited number of IP resources.

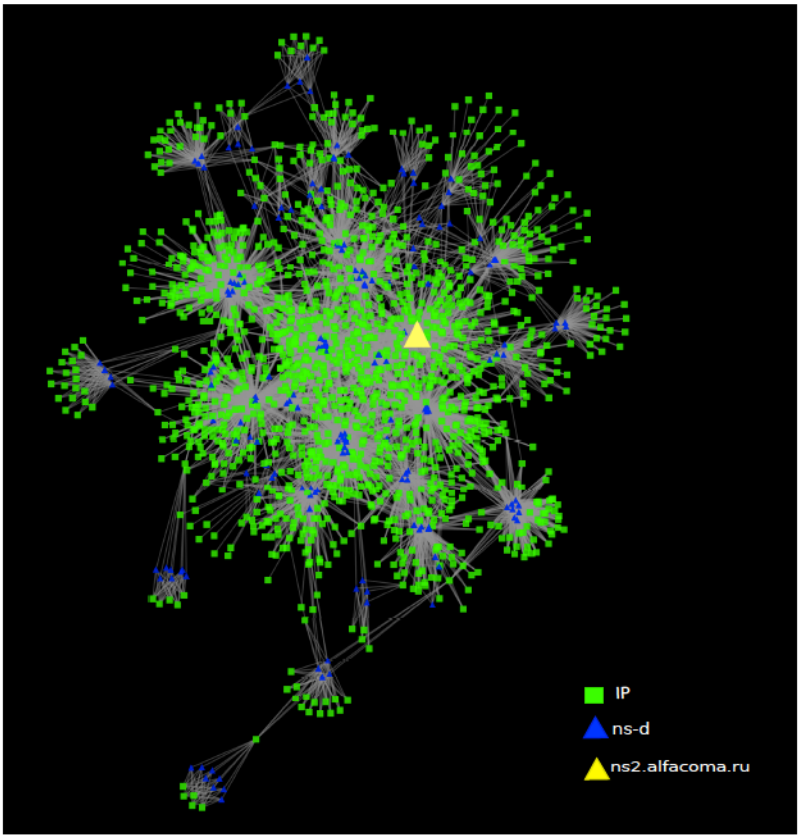


Figure 16 - Example of mapping that meet feature one

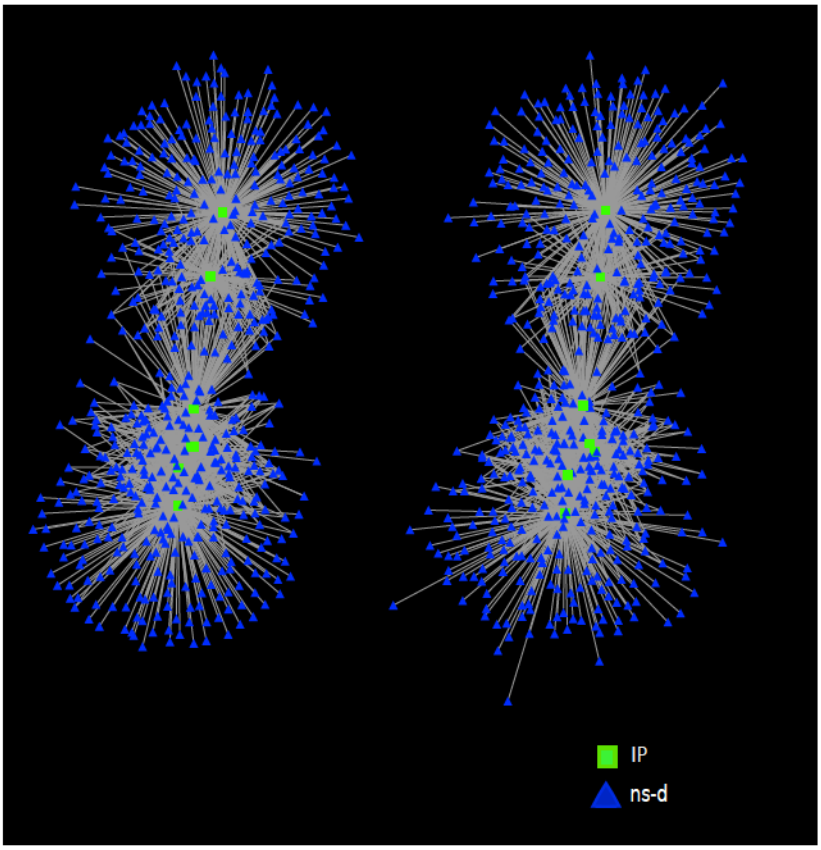


Figure 17 - Two examples of mappings with a similar structure that meet feature two

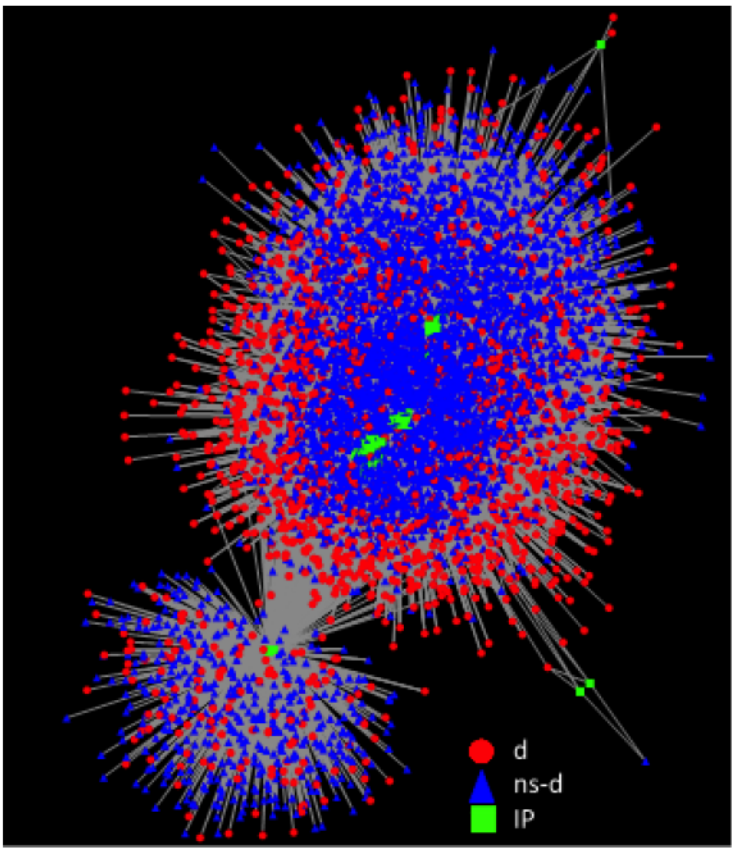


Figure 18 - Example of mapping that meets feature three

After receiving all distinct mappings of a malicious case that meets features separately, we analyze features in a combined manner. The number of d, ns-d and respective IP obtained by different combinations of features are shown in Figure 19.

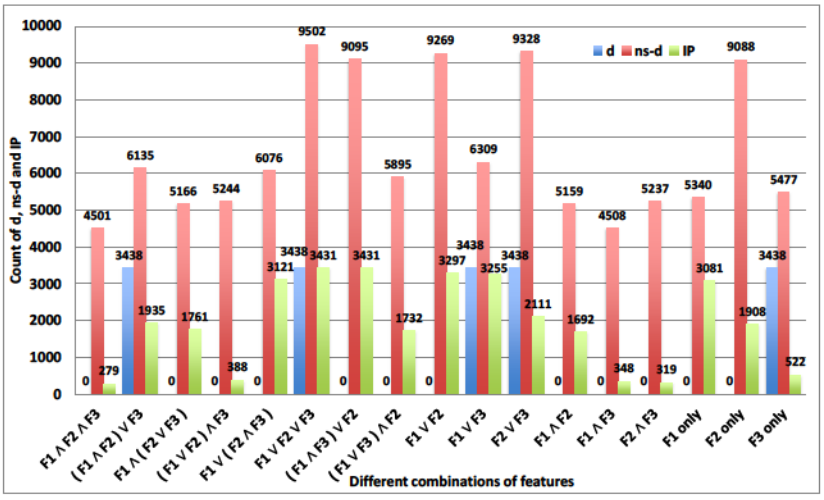


Figure 19 - Number of d, ns-d and respective IP obtained by step two

Numbers of malicious and legitimate instances from output of step two are shown separately in Table 6.

Table 6 - Benign and malicious instances from output of step two

	d		ns-d		IP of d		IP of ns-d		Total(Unique)		
	Benign	Malicious	Benign	Malicious	Benign	Malicious	Benign	Malicious	d	ns-d	IP
Feature 1	0	0	621	4,719	0	0	2,544	1,258	0	5,340	3,081
Feature 2	0	0	1,837	7,251	0	0	1,533	1,123	0	9,088	1,980
Feature 3	157	3,283	597	4,880	180	448	411	244	3,438	5,477	522

5.3.1.4. Step Three: Expanding Malicious List

In this step, d mapping to malicious ns-d and IP obtained by step two are also treated as malicious d in order to expand the malicious domain list. By this way, we obtain d in all combinations of features. Figure 20 shows the output of step three.

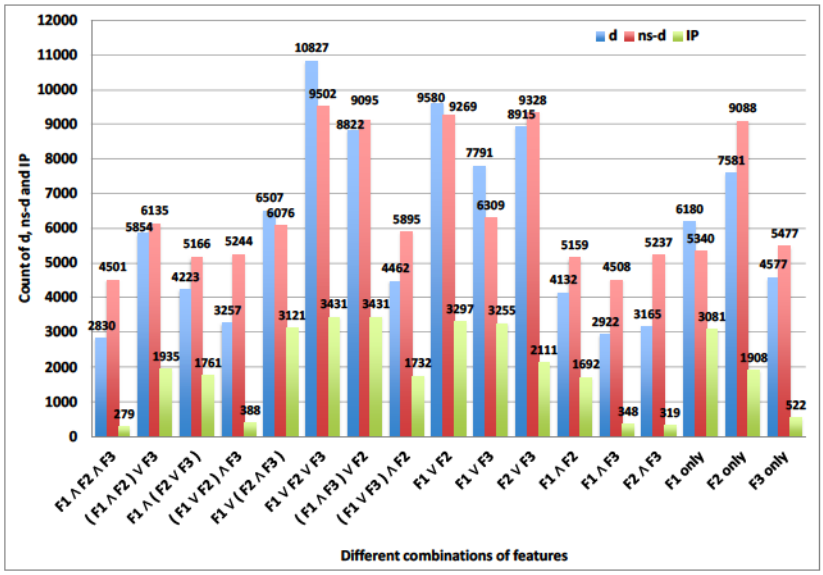


Figure 20 - Number of d, ns-d and respective IP obtained by step 3

6.4. Evaluation Methods and Results

The output of the proposed method is a list of malicious d, ns-d and IP obtained by different combinations of features. We evaluate our method by focusing on d and ns-d.

6.4.1. Evaluation of d

Firstly, the proposed method is evaluated in terms of accuracy and coverage in detecting malicious d. As ground truths, we consider all d in the input blacklist as malicious domains. As there are 323 domains that are in Alexa top 10,000 list and also detected as malicious domains by VirusTotal, we exclude these by utilizing Virus Total database from whitelist and then use the rest of domains in whitelist for evaluation. We determine accuracy by FPR (False Positive Rate). If FPR is low, it means the proposed method detects malicious d accurately. FPR is calculated by FP/N in which FP is the number of false positives d and N is the number of truly benign d. Coverage in detecting malicious d is determined by FNR (False Negative Rate). If FNR is high, it means the proposed method misses to detect a lot of malicious d. FNR is calculated by FN/P where FN is the number of false negatives d and P is the number of truly malicious d.

Figure 21 shows FPR and FNR of the proposed method for each combination of the three features.

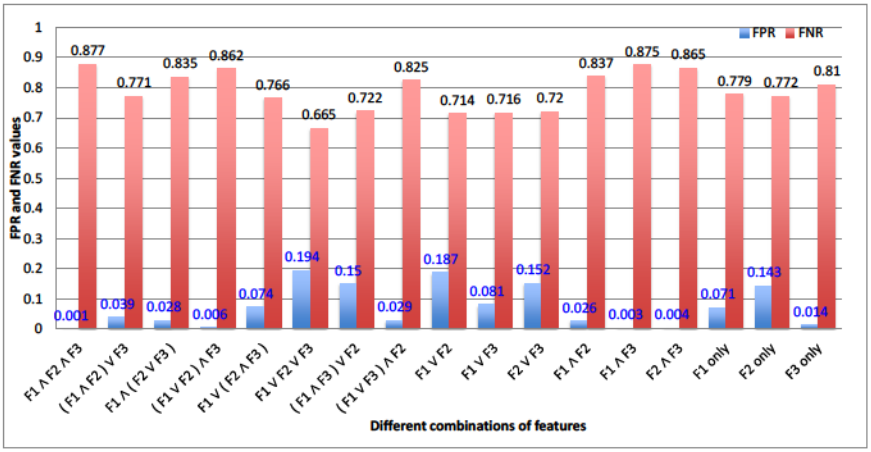


Figure 21 - FPR and FNR of d

By Figure 21, low FPR values show that the proposed method is good in accuracy of detecting malicious d. On the other hand, a high FNR indicates that there are many malicious d we miss to detect. In Figure 21, most FNR is more

than 0.7. It is because the proposed method can detect only malicious domains that meet the features we are looking for and not all malicious d are based on features we used. That is why, in practice, we recommend to use our method in parallel with another method. By comparing results in Figure 21, "(F1∨F2)^F3" is acceptable while FPR is low and FNR is not the highest. When we see features separately, F3 is best for detecting malicious d accurately.

From the point of view of accuracy, the most strict case, namely "F1^F2^F3" combination, shows the lowest FPR of 0.1%.

Our method has a high false negative rate and therefore we should mention that it is not to be used in a single-handed manner. It is indeed to be used on top of an existing detection mechanism. In that sense, we believe that we need to show that what we detect by our method is indeed malicious (i.e. low false positive rate) and different from known malicious domains and IP addresses such as those included in the existing blacklists. We show this in Figure 22.

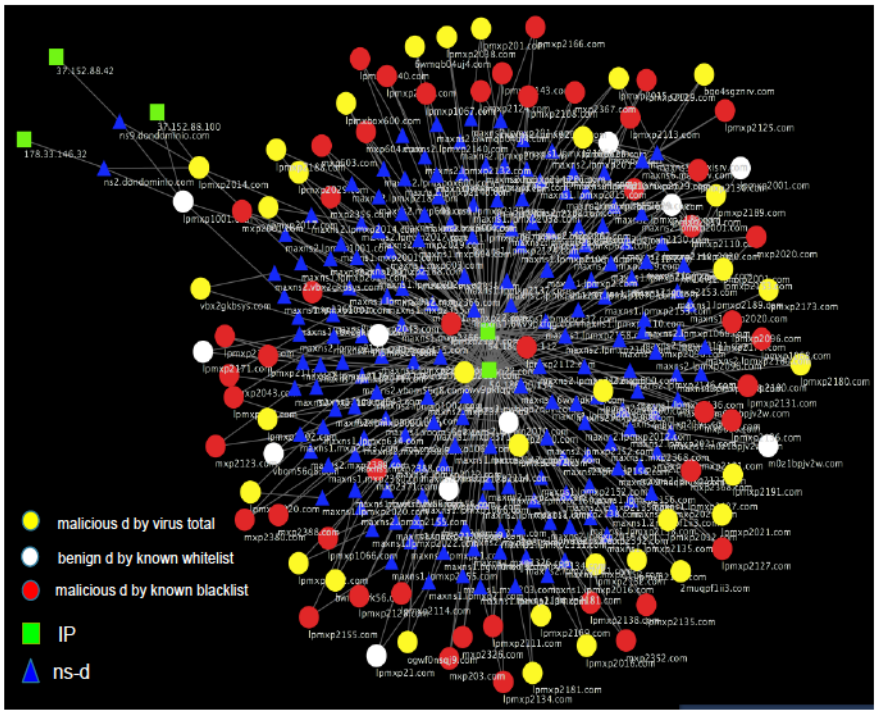


Figure 22 - Example of some evaluation results on d

6.4.2. Evaluation of ns-d

A challenge in evaluating ns-d is that there is no public benign or malicious ns-d list to the best of our knowledge. Thus, we cannot get ground truth for evaluation easily. To face this challenge, we make a malicious ns-d list and a benign ns-d list that we will be using as ground truths. For making a malicious ns-d list, we use two methods. The first one is manually searching ns-d on online web security reports and the second one is programmatically querying ns-d to VirusTotal database.

To search ns-d manually on online web security reports, we group all ns-d according to similarity of names. For example, ns-d such as "ns2.com-zy59.net", "ns3.com-fr26.net" and "ns3.com-gc22.net" are grouped according to their common name, "*.com-.*". Using a common name of each group as keyword, we search web reports and carefully read the reports in order to make sure that at least one ns-d of each group is related to malicious online activity. Then, we label each group according to malicious online activities described in the web report [52][53][54]. With this way, we can group 20% (6,460 ns-d) of all ns-d (32,218 ns-d) into 16 groups and we are able to label their relating malicious activities such as phishing, malicious advertising, drive-by-download, rouge online pharmacies and malware sites. Table 7 shows keywords and malicious activities described in web reports.

In the second method, we query all ns-d (both malicious and benign ns-d) to VirusTotal and check whether any of them are known as malicious by antivirus products in VirusTotal. From this experiment, 18.4 % (5,397 ns-d) of all ns-d are known as malicious.

Finally, we combine both results of two methods to get malicious ns-d list that we will be using as ground truth. As a result of two methods, we get 25.5 % (8,247 ns-d) of all ns-d as malicious ns-d list. Then, the rest of the ns-d not

included in our malicious ns-d list will be treated as benign ns-d. Finally, we receive a malicious ns-d list that includes 8,247 ns-d and a benign ns-d list of 24,034 ns-d. These two lists are used as ground truths in evaluation.

Table 7 - Keywords and type of malicious activities

Group Number	Keyword	Number of ns-d	Type
1	*.com-*.net	3923	Phishing Sites
2	maxns*.*.com	182	Malvertising
3	ns*.allfiles*.com	68	Drive-by-download
4	ns*.*arcinnia*.ru	108	Drive-by-download
5	ns*.*cloudbox*.com	146	Drive-by-download
6	*.cloudsvr*.com	100	Drive-by-download
7	*.*health*.ru	554	Rogue Online Pharmacies
8	*.*pharmacy*.ru		Rogue Online Pharmacies
9	*.*pill*.ru		Rogue Online Pharmacies
10	*.*tablet*.ru		Rogue Online Pharmacies
11	*.*drug*.ru		Rogue Online Pharmacies
12	ns51.*.*	357	Malware Site
13	ns52.*.*	354	Malware Site
14	ns53.*.*	335	Malware Site
15	ns54.*.*	331	Malware Site
16	*.*.orderbox-dns.com	184	Malware Site
	Total	6642	

Using the ground truth data we prepared, the proposed method is evaluated in terms of accuracy and coverage in detection of malicious ns-d. We determine accuracy by FPR (False Positive Rate). If FPR is low, it means the proposed method detects malicious ns-d accurately. FPR is calculated by FP/N in which FP is the number of false positive ns-d and N is the number of truly benign ns-d.

Coverage in detecting malicious ns-d is determined by FNR (False Negative Rate). If FNR is high, it means the proposed method misses to detect a lot of ns-d. FNR is calculated by FN/P where FN is the number of false negatives ns-d and P is the number of truly malicious ns-d.

Malicious ns-d received by different combinations of features are evaluated in terms of FPR and FNR. The results are shown in Figure 23. We also show FPR and FNR of each feature separately in order to compare features.

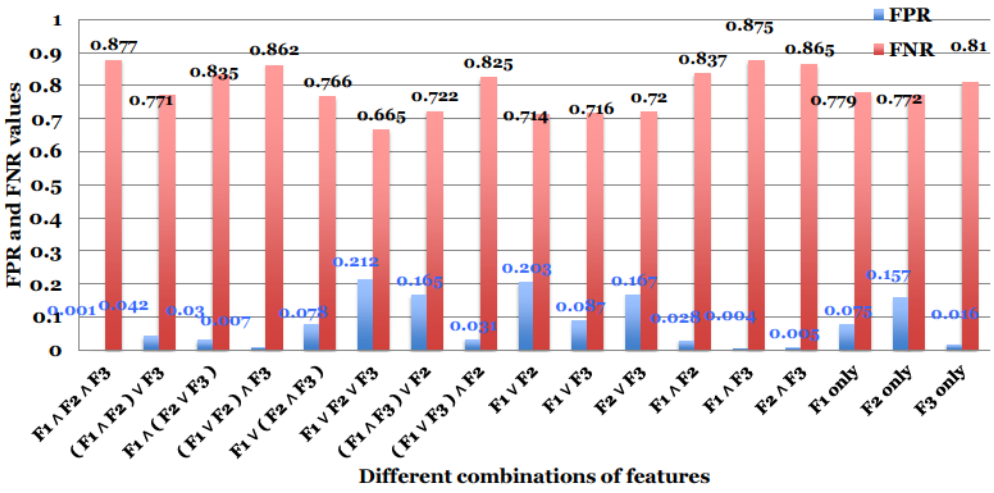


Figure 23 - FPR and FNR of ns-d

According to low FPR values in Figure 23, it shows that the proposed method can detect malicious ns-d accurately. Moreover, FNR values are also not so high. All cases have FNR of less than 0.5 meaning the proposed method can detect more than 50% of malicious authoritative name servers. When we compare, FPR and FNR values of all combinations, we found that combinations that have AND operation with F2 can achieve significantly low FNR. Thus we think F2 is better to detect wide coverage of malicious ns-d comparing with F1 and F3. From the perspective of accuracy, the performance of F1 and F3 is better than F2. From aspect of false positive, in the most strict case, "F1^F2^F3" combination, FPR is 0.8%.

Finally, evaluation of ns-d shows that we can detect malicious ns-d with low FPR and FNR. That is why, we consider that the proposed method is strong enough in practice for detecting malicious authoritative name servers. But, we also

need to notice that FPR and FNR are totally depending on the quality of ground truth data we prepared.

6.4.3. Evaluation on IP

We downloaded 575,147 blacklist IP addresses from public IP blacklists [55][56][46][57][47][58][44]. We match these IP blacklist with IP addresses output by the proposed method. The total number of output IP for all features by the proposed method is 3,431 IP addresses. As result, only 39 IP addresses (out of all 3,341 IP) match with a public blacklist. According to matching results, only 1% of our output IP addresses match with a public IP blacklist. We think that it is because output IP addresses by our proposed method are those of authoritative name servers and the blacklists we downloaded from Internet are not. Although most output IP addresses of the proposed method are not in public blacklist, we think that these IP are really malicious because of their very distinct mappings to ns-d. Some examples of mappings of IP that do not match with a public IP blacklist are shown in Figure 24.

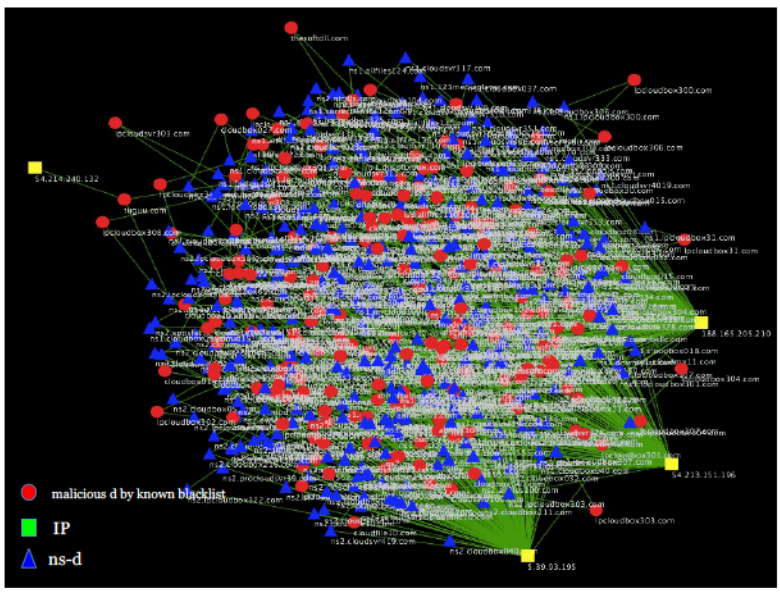


Figure 24 - Some Malicious IP

According to Figure 24, there are only 4 IP addresses that involve with that much domains and authoritative name servers. We think that these IP must be really malicious. But, none of these four IP addresses are matched with a publicly known blacklist. In the same way, five IP addresses involving a lot with many different d and ns-d are not matched with a public IP blacklist although we think it as malicious.

6.5. Discussion on Monitoring Period

We analyze how detection results change according to the length of the monitoring period. The monitoring period for all possible mappings of a particular domain is difficult to determine because it depends on how a domain is managed by the owner. Of course, DNS records of benign domains are more stable than malicious domains. By experiment results of Figure 25 and Figure 26, FPR and FNR values do not have that much difference among results. That is why we think that one month is enough to monitor the change in DNS records of domains thoroughly.

We also analyze data of less than one month such as one day, one week, two weeks and so on. Figure 27 shows how numbers of detected malicious domains are changing in monitoring period of one day, one week, two weeks and one month.

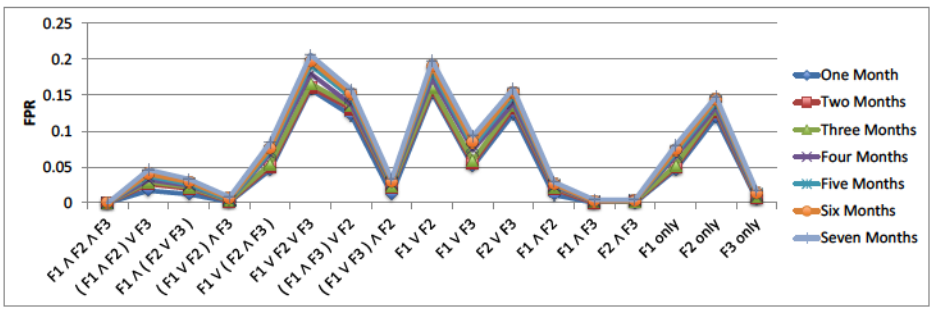


Figure 25 - FPR by each monitoring period (one month, two months and three months, etc)

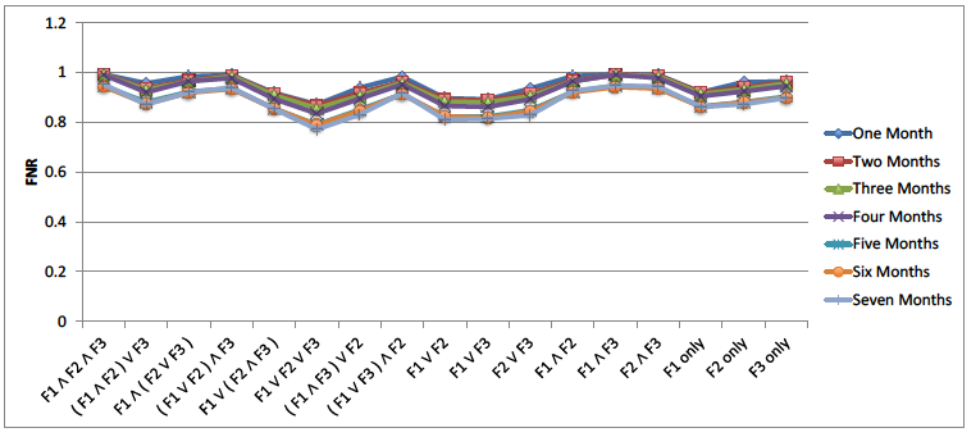


Figure 26 - FNR by each monitoring period (one month, two months and three months, etc)

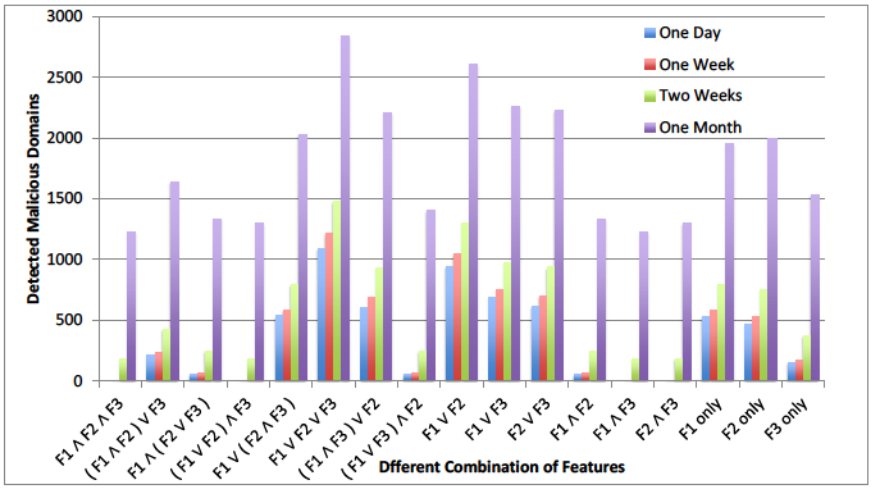


Figure 27 - Number of detected malicious domains in each monitoring period

6.6. Conclusion

We proposed a method for detecting malicious d and ns-d based on their mappings to IP addresses. In the proposed method, we use three distinct features; 1) Single ns-d is mapped to many IP, 2) Single IP is mapped to many ns-d, and 3) Single IP is mapped to both ns-d and d. Detecting malicious d and ns-d includes three steps: 1) Monitoring on mappings 2) Analyzing mappings based on three features and 3) Expanding d according to malicious ns-d and IP found in step two. Finally, we evaluate the proposed method in terms of accuracy and coverage in

detecting malicious d and ns-d. The evaluation shows that the proposed method can detect malicious d and ns-d with a high accuracy. Lastly, we note that our method purely focuses on the mapping of d and ns-d to IP and does not rely at all on any previous knowledge, such as blacklists or whitelists in the detection method.

Chapter 7

IoTPOT: A Novel Honeypot for Revealing Current IoT Threats

Introduction

Our preliminary investigation on Telnet-based attacks by darknet implies that there are number of IoT devices being compromised and misused to search and attack other IoT devices. In order to study these attacks in depth, we propose IoTPOT, a novel honeypot that emulates interactions of Telnet protocol and a variety of IoT devices.

7.1. Telnet Protocol

Before explaining IoTPOT, we briefly revisit the Telnet protocol [59]. Figure 28 illustrates the interactions between client and server on Telnet. After the TCP 3-way handshake, client and server can exchange Telnet options. Either Telnet server or client can initiate a request such as “Do Echo”, a request for echo back and “Do NAWs” a request to Negotiate About Window size (NAWs). After exchanging options, the server sends a welcome message to the client, immediately followed by login prompt. For example, “BCM96318 Broadband Router” as welcome message and “Login:” as login prompt. In this paper, we call the above initial part of interactions banner interactions. Then, the client sends a pair of username/password to log in to the server. We call this part authentication. Finally, if the credentials are valid, the client logs in and instructs the server using various shell commands. We call this part command interactions.

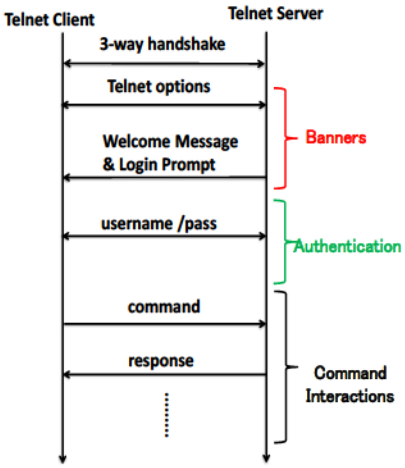


Figure 28 - Telnet protocol

7.2. IoTPOT Design

The Telnet protocol already highlights a few challenges for our honeypot design. First, we need to support options that the attacking clients choose to use. Second, we aim to provide realistic welcome message and login prompt, to deal with situations where an attacker specializes in compromising certain devices only. Third, we want to allow for logins, while we also want to observe characteristics in the authentication interactions (e.g., sequences of usernames/passwords). Finally, independent from the Telnet protocol, our honeypot should support multiple CPU architectures to capture malware across devices. Our honeypot is designed to support these features.

In order to emulate different devices, we collect these banners from the Internet by performing Telnet scans with masscan tool [60]. From all collected banners, we prioritize banners of hosts that have accessed our honeypot. Considering a self-spreading nature of these attacks, these attacking hosts can also be considered as already compromised victims, which should be emulated by our honeypot.

In the next step, during authentication, IoTPOT supports various tactics. For example, it can be configured to reject any authentication credentials to observe login attempts to allow immediate authentication regardless of the login, to accept only certain credentials, or reject the first attempts and eventually accept a

login. Finally, during command interaction, frontend responder of IoTPOT replies known commands from attackers and unknown commands are redirected to backend embedded Linux OSs of different CPU architectures. As each IoT device runs on different CPU architecture, we prepare a set of embedded Linux OS on different CPU architectures to handle the interactions of various devices.

7.3. IoTPOT Implementation

Figure 29 is the overview of IoTPOT. The heart of IoTPOT is *Frontend Responder*, which acts as different IoT devices by handling incoming TCP connection requests, banner interactions, authentication, and command interactions with a set of device profiles.

A device profile consists of a banner profile, an authentication profile, and a command interaction profile. Banner profiles determine the responses of the honeypot for banner interactions, namely Telnet options, welcome message, and login prompt. Authentication profiles determine how to respond to incoming authentication challenges. Command interaction profile determines the responses to incoming commands, consisting of a set of commands and their corresponding responses.

When an incoming command is not known yet, *Frontend Responder* establishes a Telnet connection with a backend IoTBOX and forwards the command to it. IoTBOX is a set of sandbox environments that run Linux OS for embedded devices with different CPU architectures. When an incoming command does not match with any commands in the command interaction profile, thus unknown to *Frontend Responder*, it establishes a Telnet connection with a backend IoTBOX and forwards the command to it. IoTBOX is a set of sandbox environments that run Linux OS for embedded devices with different CPU architectures. Namely, if an unknown command from attacker comes to *Frontend*

Responder with the device profile of some device X assigned, we forward the unknown commands to the sandbox running the CPU architecture of the device X.

As described later, banner profiles are collected by banner grabbing of IoT devices visiting to IoTPOT and their respective CPU architectures are manually chosen by carefully reading device manual and maker's website. If we cannot find explicit CPU information of particular IoT device, we refer to the list of applications for each CPU architecture [61][62][63][64][65].

Frontend Responder forwards a response from IoTBOX to the client. Note that the incoming commands forwarded to IoTBOX may cause malware infections or system alteration. Therefore, we reset the OS image occasionally. Moreover, IoTBOX in IoTPOT is used as high interaction system to reply to commands unknown to the *Frontend Responder* as a component of IoTPOT. We also use IoTBOX independently for analyzing captured malware binaries. The detailed explanation of IoTBOX is in Section 7.5.1.

The *Profiler* parses the interaction between *Frontend Responder* and IoTBOX, extracts the incoming command and corresponding response, and updates the command interaction profile so that *Frontend Responder* can further handle the same command without interacting with IoTBOX. Another important function of *Profiler* is the collection of banners from devices in the Internet. The *Profiler* operates in two banner grabbing modes: active scan mode and visitor scan mode. In active scan mode, *Profiler* scans different networks to collect banners from various devices. In visitor scan mode, it connects back to hosts who visit our honeypot and grab the banners.

The *Downloader* component examines the interactions for download triggers of remote files, such as malware binaries. In particular, we download from all URLs we observed via commands such as *wget*, *ftp*, and *tftp*.

Finally, network communications between *Frontend Responder* and IoTBOX are controlled by *Manager* implemented by iptables [66].

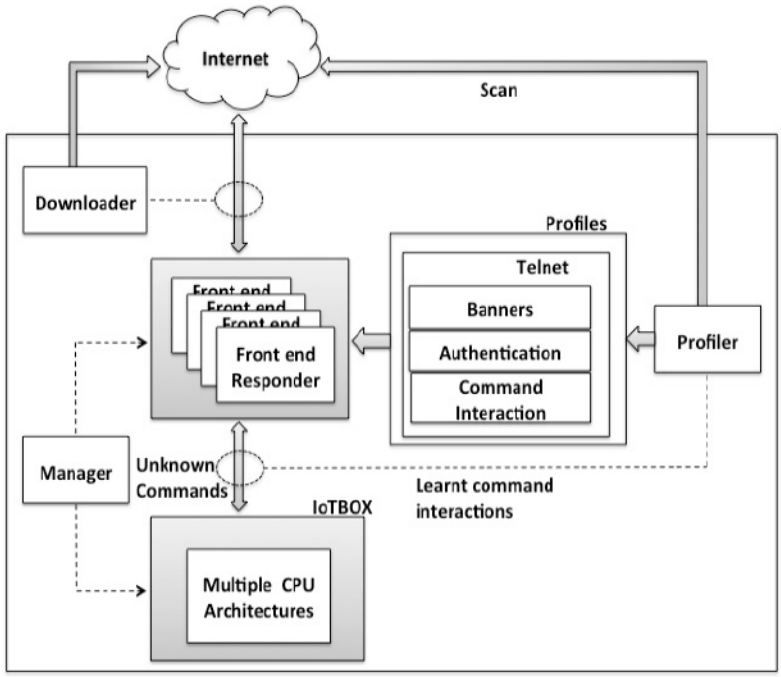


Figure 29 - Overview of IoTPOT

7.4. Observation Results

IoTPOT setup: We operate IoTPOT in two different periods: Trial operation period and stable operation period. In the trial operation period from 2014/11/07 to 2015/03/31, we have tried different configurations, device profiles, and assignment of IP addresses in ad-hoc manner trying to understand the attackers' behavior and discussing the proper setting of the honeypots. In the stable operation period from 2015/04/01 to 2015/06/20, we deploy IoTPOT on 87 IP addresses, used 29 banner profiles assigning each to three IP addresses. We set authentication profiles to accept any challenges and prepare a single command interaction profile, manually created from one of the most widely exploited DVR brands [67]. The backend IoTBOX contains an embedded Linux OSs of Debian [68] and OpenWrt [69] on 8 different CPU architectures emulated by QEMU [70].

Downloader was not fully implemented so we manually downloaded and collected malware binaries.

Summary of Observations: During 81 days of the stable operation, 180,581 hosts visit IoTPOT. Among them, 130,314 successfully log in and 79,935 attempt to download external malware binary files. We observe 481,521 download attempts in total. We manually download 106 malware binaries of 11 CPU architectures. Among 106 collected samples, 88 samples are new to the database of VirusTotal (as of 2015/06/26). Out of 18 samples that are in VirusTotal, 2 of them are not detected by any of the 57 antivirus software of VirusTotal (as of 2015/06/26).

General Flow of Telnet Attacks: We observe three typical steps of compromise: 1) The first stage of attack is intrusion, in which attackers attempt to login to our honeypot. The intrusion normally starts from scanning the targets and then dictionary-based authentication challenges. 2) The second stage after the successful intrusion is infection, in which attackers send a series of commands over Telnet to check and customize the environment, download and execute the external binaries. 3) The third stage after the infection is monetization, in which executed binaries are controlled by the attackers through C&C to conduct the intended malicious activities such as DDoS attacks and spreading of malware. Note that we intend to observe intrusion and infection by IoTPOT and after malware binaries are captured by IoTPOT, we conduct sandbox analysis using IoTBOX. Thus, in this experiment, IoTBOX is utilized in two ways, as a backend component of IoTPOT and as an independent sandbox analysis environment for analyzing the obtained binaries. The following subsections highlight some points noticed for each attack stage. The overall relationships among attacks observed at different stages are summarized in Sect. 7.5.3.1.

7.4.1. Stage 1: Intrusion

We recognize two major intrusion behaviors: login attempts with a fixed or a random order of credentials. Table 8 shows the major login patterns observed by IoTPOT. Fixed challenge order, "Fixed Order", in Table 8 means attackers try to login to IoTPOT with sequence of id and password pairs in fixed order. For example, in case of pattern name, "Fixed Order 1", the attacker's challenge always starts from "root/root" as user id and password to login to IoTPOT. Then, the pairs, "root/admin", "root/123", "root/12345" come in a fixed order of sequence till it reaches to "admin/admin". Thus, for the fixed login sequences, we can reasonably infer that these challenges are from malware sharing the same implementation of dictionary attacks. "Fixed order 2" in Table 8 is quite a long list, thus, we show only top sequences. Random challenge order means attackers try to login to IoTPOT with sequence of id and password pairs in random order. Thus, in case of "Random Order 1", it is not always true that "root/admin" will come after "root/root".

7.4.2. Stage 2: Infection

After successfully log in to honeypot, attackers check and customize the environment to prepare download of malware binary by sending series of commands over Telnet. Table 9 summarizes the 10 major patterns of command sequences observed by IoTPOT. Note that some of the patterns are observed only in the trial operation period for parameter tuning and we do not have credible counts of these patterns. We believe most infection activities are automated as exactly the same pattern of commands are repeatedly observed and also the intervals between the commands are very short.

We name each pattern by characteristic string it contains. For example, the patterns named ZORRO 1, ZORRO 2 and ZORRO 3 all have common string

"ZORRO" in their command sequences. Moreover, we can see attacker's common intension among them. Namely, all three patterns of ZORRO try to remove many existing commands and files under /usr/bin, /bin/, etc, and prepare customized command for downloading external malware binary file. With this setup, other intruders would have difficulty to abuse the system. Similar intension of attackers can be seen in case of pattern named GAYFGT. Although it does not alter the commands, instead it activates iptables [66] to drop incoming telnet connection requests. GAYFGT also has functionality to kill other existing malicious processes. All these activities explained above come in a form of commands over Telnet except that GAYFGT downloads and executes shell script file to do it. Although there are diversities in attackers' behavior at the infection stage, they all have a common goal of downloading and executing malware binary file. One more common behaviors we find is checking whether shell is usable properly or not by echoing a particular string in all families. If the appropriate reply for the echo command is not received, attacker stops the attacks.

Comparison with honeyd: We confirm that honeyd [36] cannot handle these commands in Table 9 and therefore cannot capture malware binaries observed by IoTPOT. Namely, honeyd fails to respond to very first few commands such as "cat /bin/sh" in case of ZORRO family and appropriate reply for the first echo command of GAYFGT, nttpd and KOS family and so the attacker stops sending any further commands.

Clustering of binaries captured by IoTPOT: Within the first 39 days of operation of IoTPOT (From April 1, 2015 to May 9, 2015), the collected 43 samples are not obfuscated and relatively easy to cluster by checking whether these binaries contain certain characteristic strings or not. Namely, we classify the binaries based on the hardcoded human readable strings contained in the malware binaries such as strings for C&C commands, Linux commands and file names. We

analyze the strings in binaries using the strings command of Linux. Table 10 summarizes results of manual clustering of the collected samples based on the common strings in the binaries.

Within the last 42 days of operation of IoTPOT (From May 10, 2015 to June 20, 2016), the number of captured malware increases more than double (Total 106 samples). Some of the binaries are obfuscated and so the approach to cluster the binaries using just strings command is then difficult. We need to find a better way to cluster these obfuscated binaries. This will be future works for us. Thus, for Bin 44 to Bin 106 of Appendix, samples we newly captured within the last 42 days, we cluster them into same group if command sequence from attacker is similar to previously categorized 43 samples.

Table 8 - Major log in patterns observed by IoTPOT

Pattern Name	Challenge Order	Username/Pass
Fixed Order 1	Fixed Order	root/root root/admin root/1234 root/12345 root/123456 root/1111 root/password root/dreambox root/vizxv root/system admin/admin
Random Order 1	Random Order	root/root root/admin root/12345 root/123456 admin/root admin/admin support/support ...
Fixed Order 2	Fixed Order	admin/admin admin/362729 admin/m4f6h3 admin/n3wporra admin/263297 admin/fdpm0r admin/1234 root/1234 ...
Random Order 2	Random Order	root/xc3511 root/123456 root/12345 root/root ...
Fixed Order 3	Fixed Order	guest/guest guest/12345 admin/ root/root root/admin root/ root/1234 root/123456 root/1111 root/password root/dreambox root/vizxv
Random Order 3		root/root root/toor root/admin root/user root/guest root/login root/changeme ....

Table 9 - Patterns of command sequence observed by IoTPOT

Pattern Name	Pattern of Command Sequence
ZORRO 1	1. Check type of victim shell with command "sh" 2. Check error reply of victim by running non-existing command such as ZORRO. 3. Check whether wget command is usable or not. 4. Check whether busybox shell can be used or not by echoing ZORRO.
	5. Remove various command and files under /usr/bin/, /bin, var/run/, /dev. 6. Copy /bin/sh to random file name 7. Append series of binaries to random file name of step 6 and make attacker's own shell 8. Using attacker's own shell, download binary . IP Address and port number of malware download server can be seen in the command. 9. Run binary
ZORRO 2	1. Check type of victim shell with command "sh" 2. Check error reply of victim by running non-existing command such as ZORRO. 3. Check whether wget command is usable or not. 4. Check whether busybox shell can be used or not by echoing ZORRO.
	5. Remove various command and files under /usr/bin, /bin, var/run, /dev. 6. Copy /bin/sh to random file name 7. Append series of binaries to random file name of step 6 and make attacker's own shell 8. Using attacker's own shell, download binary . IP Address and port number of malware download server cannot be seen in the command because it is hard coded in the attacker's own shell. 9. Run binary
ZORRO 3	1. Check type of victim shell with command "sh" 2. Check error reply of victim by running non-existing command such as ZORRO. 3. Check whether wget command is usable or not. 4. Check whether busybox shell can be used or not by echoing ZORRO.
	5. Remove all under /var/run, /dev, /tmp, /var/tmp 6. Copy /bin/sh to random file name 7. Append series of binaries to random file name of step 6 and make attacker's own shell 8. Using attacker's own shell, download binary. IP Address of malware download server can be seen in the command and port number cannot be seen in the command 9. Run binary
ZORRO 4	1. Check error reply of victim by running non-existing command such as "enable" or "shell". 2. Check type of victim shell with command "sh"
	3. Remove all under /var/run, /dev, /tmp, /var/tmp 4. Copy /bin/sh to random file name 5. Append series of binaries to random file name of step 4 and make attacker's own shell 6. Using attacker's own shell, download binary. IP Address of malware download server can be seen in the command and port number cannot be seen in the command 7. Run binary
GAYFGT 1	1. Check whether shell can be used or not by echoing "gayfgt" 2. Download shell script. 3. Using downloaded shell script, kill previously running malicious process, download malware binaries of different CPU architectures and block 23/TCP in order to prevent other infection. 4. Run all downloaded malware binaries.
GAYFGT 2	1. Check type of victim shell with command "sh" 2. Download shell script. 3. Using downloaded shell script, download malware binaries of different CPU architectures. 4. Run all downloaded malware binaries. 5. Make sure shell is Busybox by echoing binary that will encode into "gayfgt" only in Busybox shell.
*.sh	1. Download shell script using wget command . 2. Using downloaded shell script, download malware binaries of different CPU architectures. 3. Run all downloaded malware binaries.
nttpd 1	1. Check whether shell can be used or not by echoing "welcome" 2. Download binary to /tmp directory. 3. Run Binary.
nttpd 2	1. Check whether shell can be used or not by echoing "welcome" 2. Remove file names, .nttpd and .drop, from /tmp directory. 3. Make new file names, .nttpd and .drop. 4. Append binaries of malware through Telnet commands to .drop file. 5. Run Binary
KOS	1. Check whether shell can be used or not by echoing " $?K_O_S_T_Y_P_E" 2. List /proc/self/exe 3. Check all running process 4. Download malware binary using tftp to /mnt folder 5. Run Malware 6. Check CPU information

7.4.3. Stage 3: Monetization

IoTPOT can only observe intrusion and infection stages explained in 7.4.1 and 7.4.2. Thus, in order to further reveal how attackers are trying to monetize the compromised devices, we analyze the malware binaries collected by IoTPOT using IoTBOX as an independent malware sandbox. We show the list of samples in Appendix. The sandbox analysis results of some of the binaries are described in Section 7.5.2.

Table 10 - Clustering results of collected samples by characteristic strings in the binaries

Family Name	Keywords
Bin 1- Bin9	YESHELLO killattk
Bin 10 to Bin 41	bin.sh bin2.sh bin3.sh echo -e '\x67\x61\x79\x66\x67\x74‘
Bin 42	sh -c "cd /tmp ; rm -f .nttpd ; wget -O .nttpd http://%d.%d.%d.%d:%d ; chmod +x .nttpd ; ./.nttpd"
Bin 43	0916.davinci 0923.davinci 0923.8196

7.5. IoT Sandbox (IoTBOX)

IoTBOX is used not only as high interaction systems in IoTPOT but also as stand-alone multi-architecture sandbox. The design of IoTBOX used for two purposes are same and only routing policies are different for each purpose. So we discuss about IoTBOX design in general first and then explain consecutively how we define routing policies for IoTBOX in IoTPOT and IoTBOX as stand-alone multi-architecture sandbox in section 7.5.1.

7.5.1. IoTBOX Design

IoTBOX supports 8 different CPU architectures, namely as MIPS, MIPSEL, PPC, SPARC, ARM, MIPS64, sh4 and X86. The design of IoTBOX is shown in Figure 30. To support different CPU architectures, we need a cross compilation environments. We thus choose to run respective platforms (OS) on an emulated CPU usin QEMU [70], an open source processor emulator. Then, we use the respective OpenWrt platform to run on the emulated CPU environment. OpenWrt is a highly extensible GNU/Linux distribution for embedded devices of (typically OS of wireless routers) [69]. To install OpenWrt, we use OpenWrt Builtroot, which is a build system for the distribution and it works on Linux, BSD or MacOSX. Next to OpenWrt, IoTBOX also supports Debian Linux.

We design IoTBOX to be able to implement in single physical machine. Thus we need virtual network environment in order to connect physical interface of host machine with many virtual interfaces of QEMU based virtual machines. The following explains how we create virtual networking environment in a single physical machine.

We first create a virtual switch, which is a multiport Linux bridge [23] that connects physical interface (eth0 of host machine) at one side of bridge and many different virtual interfaces (eth0 of each virtual machine) at the other side of bridge. In order to create virtual switch, we first create virtual interface br0. As we want host only network, we do not bridge br0 with eth0 right now.

Normally, br0 interface do not need IP address as it is supposed to function as virtual switch. But, in our case, as we would like to manage our virtual switch to take part in layer 3 routing of IP packets, we assign IP address to it. We assign br0 to local IP address, which will be gateway of all virtual machines.

We then try to connect br0 with virtual machines so that packets from virtual machine can reach to br0 and vice versa. But, virtual machines' NIC (eth0

in each virtual machine of Figure 30) can only process Ethernet frames. In non-virtualized environments, the physical NIC interface (eth0 of host machine) will receive and process the Ethernet frames. It will strip out the Ethernet related overhead bytes and forward the payload (usually IP packets) further up to the OS. With virtualization however, this will not work since the virtual NICs would expect Ethernet frames. We solve this by using tap interfaces. Tap interfaces are special software entities which tell the physical NIC interface to forward Ethernet frames as it is to virtual NICs. In other words, the virtual machines connected to tap interfaces will be able to receive raw Ethernet frames. We manage virtual bridge connection of br0 to virtual NICs through tap interfaces by using Linux brctl [24]. We automate all these steps so that virtual network connection can be done automatically whenever a new virtual machine is added.

Now, br0 is connected to many virtual machines. We have discussed so far are all about layer 2 level connections. From the viewpoint of layer 3, br0 interface will be same network with all virtual machines and it will be gateway for all virtual machines. The interface, eth0 of host machine will be on different network and as we do not bridge it directly with br0, we connect br0 and eth0 through NAT (Network Address Translation) managed by *Access Controller*. *Access Controller* implemented by iptables controls all networking related operations such as NAT and outbound traffic from each virtual machine.

IoTBOX as stand-alone multi-architecture sandbox: In this case, *Access Controller* controls NAT and outbound traffic from each virtual machine such as C&C communication, DNS resolution and attack traffic such as DoS. We block all outgoing DoS traffic from malware except allowing some DNS and HTTP traffic of maximum 5 packets per minutes. 23/TCP scans are redirected to *Dummy Server*, which is indeed IoTPOT. With this way, we can monitor how propagation over Telnet is done.

Analysis Report outputs the results of pcap analysis results for every 24 hours showing total number of packets, start time and end time of packet captures, data byte/bite rate, average packet size and rate and total number of victim IP address for each attack. In addition, summary of commands strings from C&C are summarized if any.

IoTBOX as high interaction system in IoTPOT: In this case, *Access Controller* will accept only incoming connection from *Frontend Responder's* IP addresses and all outbound traffics from high interaction systems except corresponding replies of commands redirected by *Frontend Responder* will be blocked. Please also note that what *Manager* in Figure 29 is doing is exactly the same as *Access Controller* what we have discussed here.

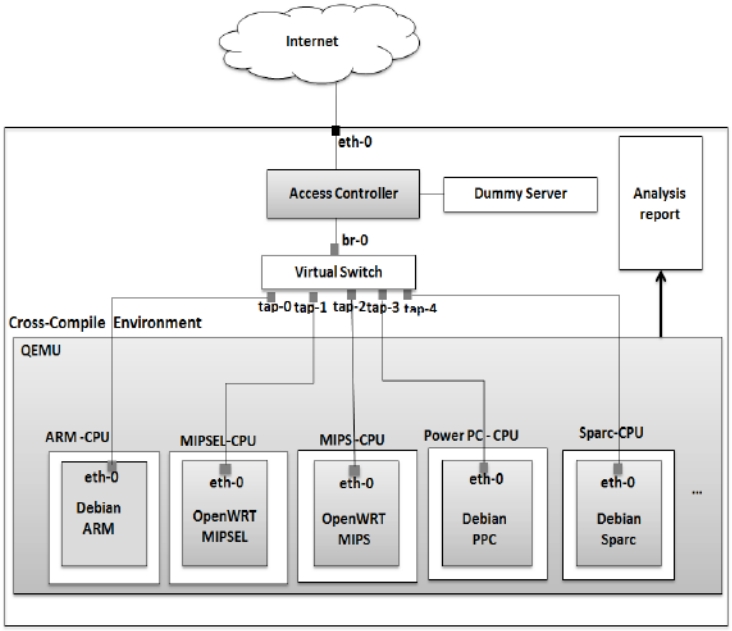


Figure 30 - Overview of IoTBOX

7.5.2. Analysis Results by IoTBOX

Using IoTBOX, we analyze 51 selected malware binaries of 8 CPU architectures. Because of limited resources of IoTBOX, malware binary for popular CPU architectures of embedded devices such as ARM, MIPS and MIPSEL

are focused more in analysis. Please refer to Appendix for the information of analyzed malware samples. Red colored samples show analyzed binaries.

We observe 25 of 50 malware binaries performed 11 different types of DoS attacks and 3 different types of scans such as telnet scan and scans on TCP ports such as 23,80,8080, 5916 and UDP port such as 123, 3143. The 5 samples cannot be executed because of errors.

A summary of the observed attacks is illustrated in Figure 31. Most attacks we observed are UDP floods and many different types of TCP floods. We also observe UDP floods against multiple destination ports, which seemed to aim at flooding target network. Interestingly, we also observe DNS water torture attack [71], SSL attacks [72] and other two unknown DNS based attacks in which a large number of queries to unknown type of DNS resource records (RR) are sent to an authoritative name server of a popular ISP. Sample Bin 43 exhibits unique functionality of a fake web hosting. Namely, it starts hosting a web page that looks like a top page of a popular Chinese search engine "baidu.com". In order to avoid any misuse of the fake web page in real attack, we carefully monitor if any incoming connections appear although nothing has been seen yet. One more point we notice is that Bin 13, 19, and 22 of Figure 31 have a backdoor port 5000/UDP open for further remote control of the compromised host because the initial intrusion route, the Telnet, would already have been blocked by iptables [73] during the infection phase to prevent other attackers from compromising the host.

7.5.3. Analysis on Attacks

7.5.3.1. Overview of Observed Attacks

Figure 32 depicts the overview of Telnet-based attacks observed by IoTPOT and IoTBOX. . In order to understand overview of Telnet attacks observed by our honeypot, we make mappings between different patterns of intrusion and

infection behaviors observed by IoTPOT and monetization behaviors observed by malware analysis with IoTBOX. For example, intrusion pattern “Fixed Order 3”, which is shown in Table 8, is always followed by infection pattern “ZORRO 4”, explained in Table 9. Then, infection pattern “ZORRO 4” ends up downloading one of the binaries from certain clusters of binaries that contain common strings, which will eventually exhibit similar monetization behavior, namely DoS attacks. These mappings reveal that the related patterns and behaviors of attacks can be separated into five major groups, referred as five corresponding malware families. We also notice that some families seem to spread more aggressively than others. Namely, even within one month of operation, ZORRO family has updated its Telnet command sequences twice. This family also has increased the diversity of binaries from 7 architectures to 9 architectures dramatically to support more CPU architectures.

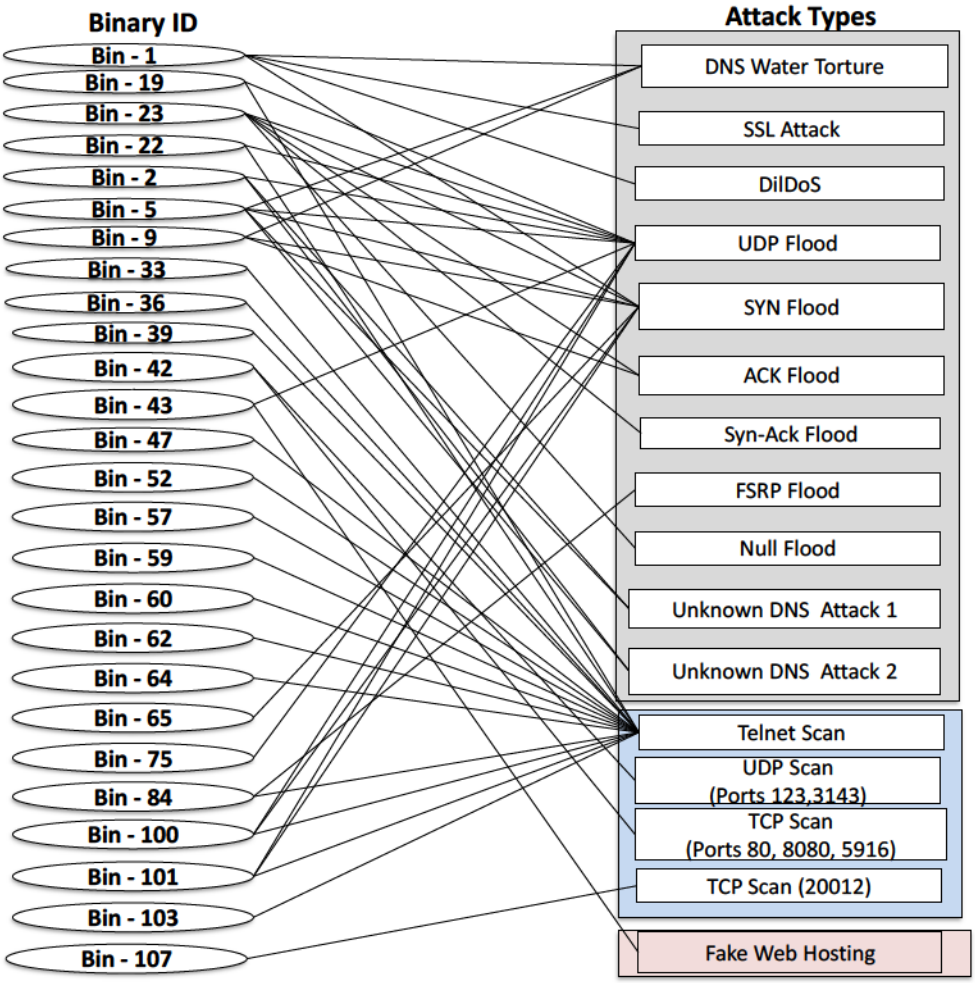


Figure 31 - Overview of attack by IoTBOX

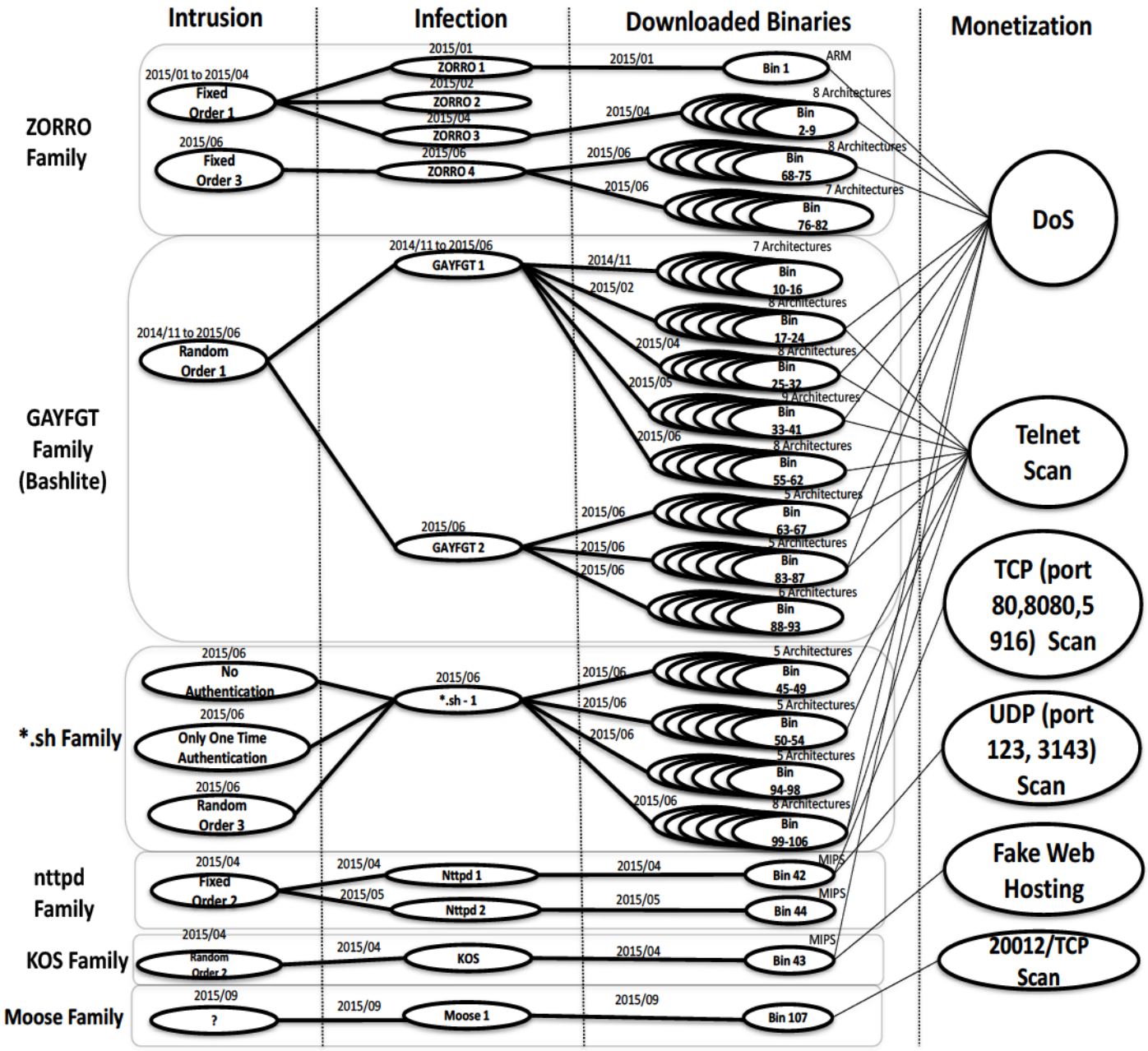


Figure 32 - Overview of observed attacks by IoTPOT and IoTBOX

Following are our findings.

- We have observed six malware families whose intrusion, infection, and malware binaries are independent from each other.
- From viewpoint of monetization, the different families share the same goal of performing DoS attacks and scans. The only exception is Bin 43 that starts to host a fake search engine.
- Some families seem to spread more aggressively than others. Namely, as in Figure 32, ZORRO, GAYFGT and nttpd familes have updated its command sequences twice during observation period. Also, the GAYFGT family has increased the diversity of binaries to support more CPU architectures.

7.5.4. Overview of Attacking Botnet

7.5.4.1. Botnet Architectures

Figure 33 shows overview of botnet attacking IoTPOT. Basically, scanning hosts, we call as Scanners (S), perform Internet wide Telnet scans in order to find hosts listening on Telnet for further infections. After successful Telnet login, intruding host (I) intrudes the victim sending sequence of commands over Telnet in order to make victim machine download the malware binary from malware download server (D). Downloaded binary is run and after infection, victim receives commands from Command and Control Server (C) to perform various DoS attacks and scans. These S, I, D and C can be different hosts or same host. For example, a single host may perform as (S, I, D) or (D and C) are single host while S and I are different hosts. By analyzing S, I, D and C involving IoTPOT, we found 8 different botnet architectures as follow:

- Botnet relating to ZORRO family has many host performing scanning only and few I, D and C of different combinations (B1, B2, B3 of Figure 33). Coordinated instruction of S and I of this family is explained more in section 7.5.4.2.
- Botnet of GAYFGT and *.sh families have many hosts performing both scanning and intruding while D and C are same or separate hosts. (B4 and B5 of Figure 33).
- Propagation of nttpd family looks alike warm infection in which attacking host itself is scanner, intruder and malware download server (B6 in Figure 33). There are also cases in which scanning and intruding host make victim infects sending malware binary over Telnet. In such case, it is not necessary to download malware binary from malware download server (B7 in Figure 33).
- Botnet of KOS family has many hosts performing both scanning and intruding while D and C are separate hosts (B8 of Figure 33). C can be connected by resolving "s6.kill123.com" domain. In order to resolve the domain, authoritative name server IP address of "S6.kill123.com" is hard coded in nttpd malware (bin 44 of Appendix). This authoritative name server is not reachable through normal authoritative name server DNS stacks. With this way, attacker setup authoritative name server as part of his or her botnet.

- Botnet of Moose family is general botnet structure in which same scanning and intruding hosts intrude try to intrude system and malware is downloaded from C&C servers. (B9 of Figure 33)

7.5.4.2. Coordinated Intrusions of Botnet

In the trial period, we notice a coordinated intrusion by ZORRO family, in which reconnaissance and the actual malware infection are done by different hosts in coordination. Namely, we observed a reconnaissance host attempting logins to our honeypot, which had been configured to accept only a single pair of username/ password. Eventually, this reconnaissance host successfully logged in by guessing a valid login, and sent several commands over Telnet for information gathering of the compromised host, including the architecture of CPU it ran. However, it disconnected the session neither downloading nor executing any malware binary file. After a while, we observe another host that visit our honeypot and successfully log in with just one challenge implying that it already knew the valid credential from the earlier reconnaissance. This intrusion host then sends series of commands to download and execute external malware binary. The downloaded binary file is indeed of the CPU architecture of the honeypot and so we think that this host knows the CPU architecture of the honeypot from the reconnaissance.

We then set a new login credential and kept observation. We have a visit of another reconnaissance host and it succeeded to log in and identify the new credential. After a while, the same intrusion host from the previous intrusion visits us again with the newly obtained credential and infected the malware. After all, we observe a group of over 100 reconnaissance hosts and only a single intrusion host in coordination. Figure 34 depicts the coordinated attack.

7.6. Conclusion

We have shown that IoT devices are susceptible to compromises and increasingly are also target for malware on the masses. We identify six malware families, which show worm-like spreading behavior, all of which are actively used in DDoS attacks. As future work, we plan to extend IoTPOT to support more protocols that are likely the target by attacks, such as SSH. Furthermore, we aim to extend the

sandbox with capabilities to stimulate even more architectures and environments that are common on IoT devices.

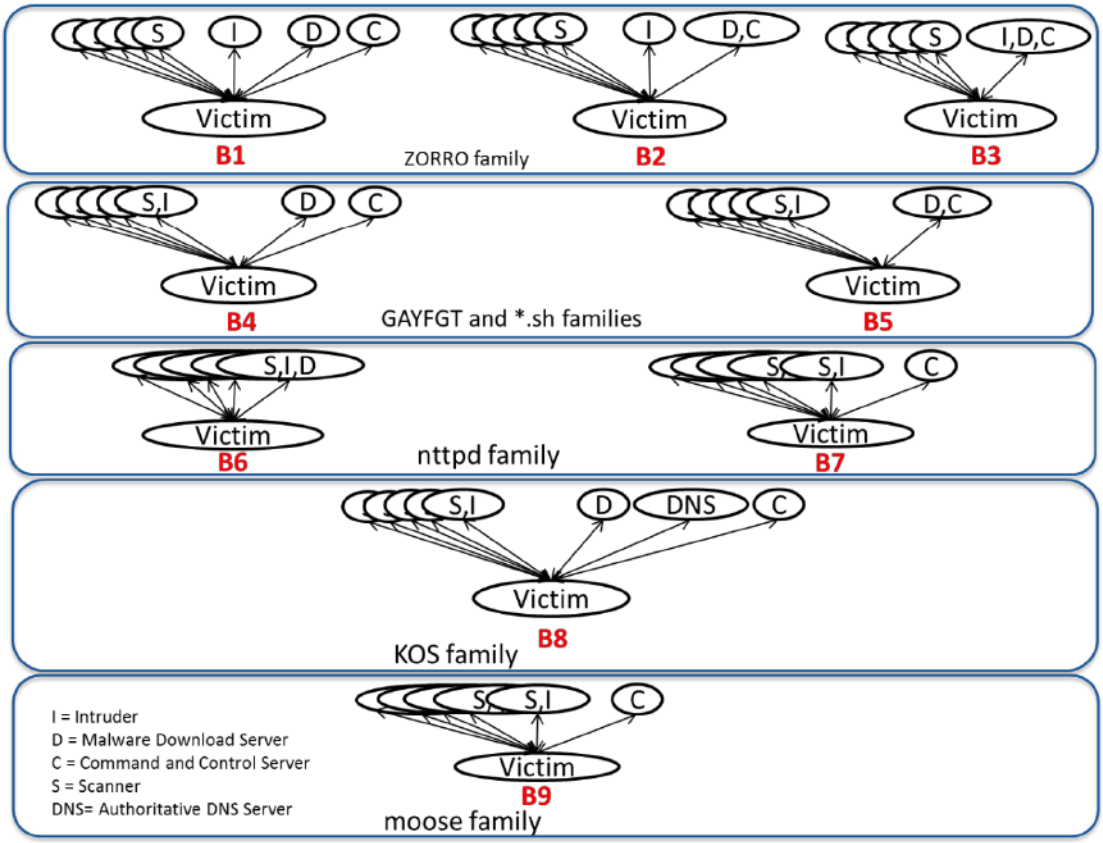


Figure 33 - Botnet architecture

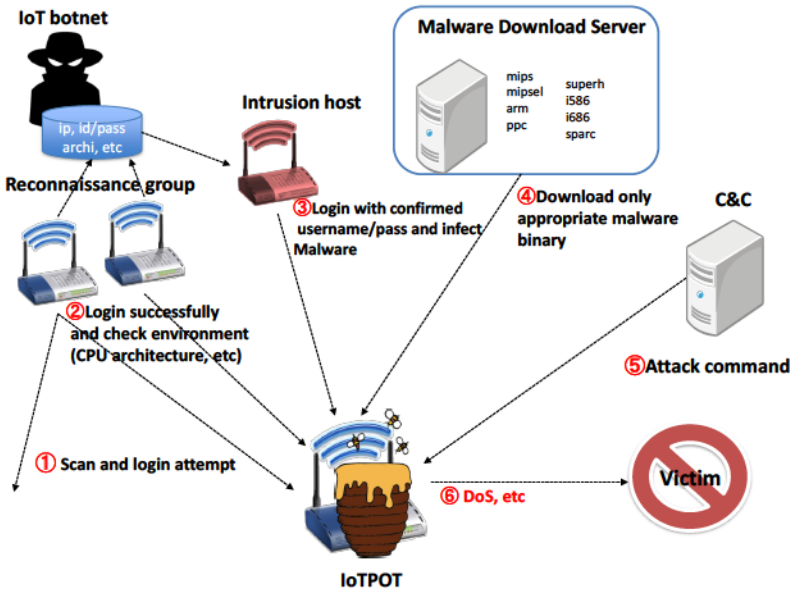


Figure 34 - Coordinated attack of ZORRO family observed by IoTPOT

Chapter 8

Conclusion and Future Works

8.1. Conclusion

Detection of cyber attack resources is an important step for mitigation of cyber attack resources. In this dissertation, we present how to detect malicious authoritative name servers and IoT botnet by efficiently coordinating passive and active monitoring approaches while proposing a framework for coordination.

In Chapter 3, we propose framework for the detection of cyber attack resources. Using the proposed framework, we proposed two novel detection methods for the detection of malicious authoritative name server and IoT botnet.

In Chapter 5, we analyze passive DNS traffic and try to extract out features for detection of malicious authoritative name servers. Using extracted features, we show how to find malicious authoritative name server by extracting domains from DNS traffic. This is preliminary study for Chapter 6 and this preliminary study, we find out that domain flux size feature is quite strong for detection of malicious authoritative name servers. Thus, more specific and carefully categorized features of domain flux size feature are studied and propose a comprehensive detection method explained in Chapter 6.

In Chapter 6, we present a novel method for detecting malicious "domains" (noted as d) and malicious "authoritative name servers" (noted as ns-d) based on their distinct mappings to "IP addresses" (noted as IP). Namely, we present three features to detect them; 1) Single ns-d is mapped to many IP, 2) Single IP is mapped to many ns-d, and 3) Single IP is mapped to both ns-d and d. We evaluate proposed method in terms of accuracy and coverage in detection of malicious d and ns-d. The evaluation shows that our detection method can achieve significantly low false positive rate in detecting both malicious d and ns-d without relying on any previous knowledge, such as blacklists or whitelists.

In Chapter 7, we reveal current IoT threats proposing IoTPOT, which is a honeypot system in which both active and passive monitoring approaches are coordinated. While honeypot portion of IoTPOT captures malware as passive

monitoring system, the scanner portion of IoTPOT performs active probe of infected IoT devices in order to detect attacker's IoT botnet.

In conclusion, this dissertation contributes following;

We propose novel method for detection of malicious authoritative name servers and malicious domains by coordination of passive and active monitoring approach. The proposed method achieves low false positive rate in detecting both malicious d and ns-d without relying on any previous knowledge, such as blacklists or whitelists.

We propose novel method of revealing current IoT threats. The main contributions of this study are as follows.

- The study point out a huge increase of Telnet based attacks and the involment of IoT devices.
- To analyze the scope and variety of IoT related attacks, a honeypot system, which mimics IoT devices and captures Telnet based intrusion, is proposed.
- To analyze captured malware, a sandbox system for analyzing malware of 8 different CPU architectures such as ARM, MIPS, MIPSEL, PPC, X86, etc., is implemented.
- Analysis by sandbox reveals that there are at least six DDoS malware families targeting IoT devices.
- We share our samples and traffic with more than 11 international organizations.

8.2. Future Works

The scope of study for detecting cyber resources by active and passive monitoring is broad. In this study, we focus on detection of important attack resources relating to DNS protocol and Telnet protocol introducing a framework for coordination of passive and active monitoring. We think that coordination of passive and active monitoring approaches can significantly make improvement in detection of cyber attack resources. Thus, future works should focus on how to improve existing methodologies for countermeasure of detected cyber attack resources.

Bibliography

[1] "HOW FAST-FLUX SERVICE NETWORKS WORK | The Honeynet Project." [Online]. Available: http://www.honeynet.org/node/132. [Accessed: 19-Oct-2015].

[2] "WP Botnet Communications Primer (2009-06-04).pdf." .

[3] "sac-025-en.pdf." .

[4] "Morto worm sets a (DNS) record | Comunidad de Symantec Connect." [Online]. Available: http://www.symantec.com/connect/blogs/morto-worm-sets-dns-record. [Accessed: 19-Oct-2015].

[5] E. Masashi, I. Daisuke, S. Jungsuk, N. Junji, O. Kazuhiro, and N. Koji, "nicter: a large-scale network incident analysis system: case studies for understanding threat landscape," *BADGERS 11 Proc. First Workshop Build. Anal. Datasets Gather. Exp. Returns Secur.*

[6] M. E.L. and T. L. D., "The Carna Botnet Through the Lens of a Network Telescope," *Proc. 6th Int. Symp. Found. Pract. Secur. FPS 2003 Oct. 2013*, Oct. 2013.

[7] "Internet Census 2012." [Online]. Available: http://internetcensus2012.bitbucket.org/paper.html. [Accessed: 24-May-2015].

[8] "p0f v3." [Online]. Available: http://lcamtuf.coredump.cx/p0f3/. [Accessed: 24-May-2015].

[9] "rep/dionaea," *GitHub*. [Online]. Available: https://github.com/rep/dionaea. [Accessed: 20-Oct-2015].

[10] "Specialized Honeypots for SSH, Web and Malware Attacks." [Online]. Available: https://zeltser.com/honeypots-for-malware-ssh-web-attacks/. [Accessed: 20-Oct-2015].

[11] "Blogs | The Honeynet Project." [Online]. Available: http://www.honeynet.org/. [Accessed: 20-Oct-2015].

[12] "Glastopf Honeypot Project Page." [Online]. Available: http://glastopf.org/. [Accessed: 20-Oct-2015].

[13] "desaster/kippo," *GitHub*. [Online]. Available: https://github.com/desaster/kippo. [Accessed: 20-Oct-2015].

[14] L. Krämer, J. Krupp, D. Makita, T. Nishizoe, T. Koide, K. Yoshioka, and C. Rossow, "AmpPot: Monitoring and Defending Against Amplification DDoS Attacks."

[15] "Conpot." [Online]. Available: http://conpot.org/. [Accessed: 20-Oct-2015].

[16] "SCADA HoneyNet Project: Building Honeypots for Industrial Networks." [Online]. Available: http://scadahoneynet.sourceforge.net/. [Accessed: 20-Oct-2015].

[17] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: analysing the rise of IoT compromises," *EMU*, vol. 9, p. 1, 2015.

[18] "Honeytoken," *Wikipedia, the free encyclopedia*. 20-Sep-2013.

[19] "Nmap: the Network Mapper - Free Security Scanner." [Online]. Available: https://nmap.org/. [Accessed: 20-Oct-2015].

[20] "ZMap · The Internet Scanner." [Online]. Available: https://zmap.io/. [Accessed: 20-Oct-2015].

[21] “robertdavidgraham/masscan,” *GitHub*. [Online]. Available: https://github.com/robertdavidgraham/masscan. [Accessed: 20-Oct-2015].
[22] “p0f v3.” [Online]. Available: http://lcamtuf.coredump.cx/p0f3/. [Accessed: 20-Oct-2015].
[23] “Category:Vulnerability Scanning Tools - OWASP.” [Online]. Available: https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools. [Accessed: 20-Oct-2015].
[24] “dns-zone-transfer NSE Script.” [Online]. Available: https://nmap.org/nsedoc/scripts/dns-zone-transfer.html. [Accessed: 20-Oct-2015].
[25] Y. M. P. Pa, K. Yoshioka, and T. Matsumoto, “Search Engine Based Investigation on Misconfiguration of Zone Transfer,” in *2013 Eighth Asia Joint Conference on Information Security (Asia JCIS)*, 2013, pp. 56–62.
[26] “Free OpenSSL Heartbleed Vulnerability Scanner | Rapid,” *Rapid7*. [Online]. Available: http://www.rapid7.com/resources/free-security-software-downloads/openssl-heartbleed-vulnerability-scanner.jsp. [Accessed: 20-Oct-2015].
[27] “Thug - Python low-interaction honeyclient.” [Online]. Available: https://buffer.github.io/thug/. [Accessed: 20-Oct-2015].
[28] Y. K. Mitsuaki Akiyama, “MARIONETTE: Client Honeypot for Investigating and Understanding Web-based Malware Infection on Implicated Websites,” 2009.
[29] “Shodan.” [Online]. Available: https://www.shodan.io/. [Accessed: 31-Jan-2016].
[30] “Internet-Wide Scan Data Repository.” [Online]. Available: https://scans.io/. [Accessed: 31-Jan-2016].
[31] “Censys,” *Censys*. [Online]. Available: https://censys.io/. [Accessed: 31-Jan-2016].
[32] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, “Building a Dynamic Reputation System for DNS.,” in *USENIX security symposium*, 2010, pp. 273–290.
[33] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, “EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis.,” in *NDSS*, 2011.
[34] S. Hao, N. Feamster, and R. Pandrangi, “Monitoring the initial DNS behavior of malicious domains,” in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 269–278.
[35] X. Hu, M. Knysz, and K. G. Shin, “Measurement and analysis of global IP-usage patterns of fast-flux botnets,” in *2011 Proceedings IEEE INFOCOM*, 2011, pp. 2633–2641.
[36] “Developments of the Honeyd Virtual Honeypot.” [Online]. Available: http://www.honeyd.org/. [Accessed: 06-Jan-2016].
[37] “zx2c4/telnet-password-honeypot · GitHub.” [Online]. Available: https://github.com/zx2c4/telnet-password-honeypot. [Accessed: 24-May-2015].
[38] “dionaea — catches bugs.” [Online]. Available: http://dionaea.carnivore.it/. [Accessed: 24-May-2015].
[39] “home [Nepenthes - finest collection -].” [Online]. Available: http://nepenthes.carnivore.it/. [Accessed: 24-May-2015].
[40] K. Kishimoto, K. Ohira, Y. Yamaguchi, H. Yamaki, and H. Takakura, “An adaptive honeypot system to capture ipv6 address scans,” in *Cyber Security (CyberSecurity), 2012 International Conference on*, 2012, pp. 165–172.

[41] C. Leita and M. Dacier, “SGNET: a worldwide deployable framework to support the analysis of malware threat models,” in *Dependable Computing Conference, 2008. EDCC 2008. Seventh European*, 2008, pp. 99–109.
[42] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda, “Fire: Finding rogue networks,” in *Computer Security Applications Conference, 2009. ACSAC’09. Annual*, 2009, pp. 231–240.
[43] A. Nappa, Z. Xu, M. Z. Rafique, J. Caballero, and G. Gu, “Cyberprobe: Towards internet-scale active detection of malicious servers,” in *Network and Distributed System Security Symposium*, 2014.
[44] “ZeuS Tracker :: Home.” [Online]. Available: https://zeustracker.abuse.ch/. [Accessed: 20-Oct-2015].
[45] “Malware Domain List.” [Online]. Available: http://www.malwaredomainlist.com/mdl.php. [Accessed: 20-Oct-2015].
[46] [Online]. Available: http://malc0de.com/bl/IP_Blacklist.txt. [Accessed: 20-Oct-2015].
[47] “MDL.” [Online]. Available: http://www.malwaredomainlist.com/. [Accessed: 20-Oct-2015].
[48] “KnujOn.com - The Internet Buck Stops Here.” [Online]. Available: http://knujon.com/. [Accessed: 20-Oct-2015].
[49] “MalwareURL.” [Online]. Available: http://www.malwareurl.com/. [Accessed: 20-Oct-2015].
[50] “DNS-BH – Malware Domain Blocklist.” .
[51] “Alexa - Actionable Analytics for the Web.” [Online]. Available: http://www.alexa.com/. [Accessed: 20-Oct-2015].
[52] “Is This Website Safe | Website Security | Norton Safe Web.” [Online]. Available: http://safeweb.norton.com/. [Accessed: 20-Oct-2015].
[53] “Spam404,” *Spam404*. [Online]. Available: http://www.spam404.com/. [Accessed: 20-Oct-2015].
[54] “VirusTotal - Free Online Virus, Malware and URL Scanner.” [Online]. Available: https://www.virustotal.com/. [Accessed: 20-Oct-2015].
[55] “OpenBL.org - Abuse Reporting and Blacklisting.” [Online]. Available: http://www.openbl.org/. [Accessed: 20-Oct-2015].
[56] “Blacklist IP Addresses Live Database 2014. Blacklist Check.” [Online]. Available: http://myip.ms/browse/blacklist. [Accessed: 20-Oct-2015].
[57] [Online]. Available: http://www.unsubscore.com/blacklist.txt. [Accessed: 20-Oct-2015].
[58] “Perishable Press | WordPress, Web Design, Code & Tutorials.” [Online]. Available: https://perishablepress.com/. [Accessed: 20-Oct-2015].
[59] “RFC 854 - Telnet Protocol Specification.” [Online]. Available: https://tools.ietf.org/html/rfc854. [Accessed: 24-May-2015].
[60] “robertdavidgraham/masscan,” *GitHub*. [Online]. Available: https://github.com/robertdavidgraham/masscan. [Accessed: 06-Jan-2016].
[61] “List of applications of ARM cores,” *Wikipedia, the free encyclopedia*. 30-Sep-2015.
[62] “List of MIPS microarchitectures,” *Wikipedia, the free encyclopedia*. 17-Sep-2015.
[63] “PowerPC applications,” *Wikipedia, the free encyclopedia*. 18-Dec-2012.
[64] “SuperH,” *Wikipedia, the free encyclopedia*. 30-Sep-2015.

[65] “SuperH RISC engine Family | Renesas Electronics.” [Online]. Available: http://www.renesas.com/products/mpumcu/superh/index.jsp. [Accessed: 03-Nov-2015].
[66] “netfilter/iptables project homepage - The netfilter.org project.” [Online]. Available: http://www.netfilter.org/. [Accessed: 03-Nov-2015].
[67] “Remote Code Execution in Popular Hikvision Surveillance DVR | Threatpost | The first stop for security news.” [Online]. Available: https://threatpost.com/remote-code-execution-in-popular-hikvision-surveillance-dvr/109552. [Accessed: 24-May-2015].
[68] “Index of /~aurel32/qemu/mipsel.” [Online]. Available: https://people.debian.org/~aurel32/qemu/mipsel/. [Accessed: 04-Nov-2015].
[69] “OpenWrt.” [Online]. Available: https://openwrt.org/. [Accessed: 30-Oct-2015].
[70] “QEMU.” [Online]. Available: http://wiki.qemu.org/Main_Page. [Accessed: 30-Oct-2015].
[71] Secure64, “Water Torture: A Slow Drip DNS DDoS Attack « Cybersecurity « Cyber Trust Matters.” .
[72] “DDoS Attacks on SSL: Something Old, Something New.” [Online]. Available: http://asert.arbornetworks.com/ddos-attacks-on-ssl-something-old-something-new/. [Accessed: 24-May-2015].
[73] “netfilter/iptables project homepage - The netfilter.org project.” [Online]. Available: http://www.netfilter.org/. [Accessed: 24-May-2015].
[74] “WP Botnet Communications Primer (2009-06-04).pdf.” .
[75] “Microsoft Word - SAC025-fastfluxversion1dot0.doc - sac-025-en.pdf.” .
[76] G. Ollmann, “Botnet communication topologies,” *Retrieved Sept.*, vol. 30, p. 2009, 2009.

List of Papers

Reviewed Papers in Journals

J-1) Yin Minn Pa Pa, Katsunari Yoshioka, Tsutomu Matsumoto "Detecting Malicious Domains and Authoritative Name Servers Based on Their Distinct Mappings to IP Addresses", Journal of Information Processing, Japan, Vol.23, No.5, pages 623-632, September 2015.

J-2) Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Christian Rossow "IoTPOT: A Novel Honeypot for Revealing Current IoT Threats", Journal of Information Processing, Japan, Vol.24, No.3. March 2016.

Reviewed Papers in International Conference Proceedings

I-1) Yin Minn Pa Pa, Katsunari Yoshioka, Tsutomu Matsumoto "Search Engine Based Investigation on Misconfiguration of Zone Transfer", Asia Joint Conference on Information Security (Asia-JCIS), Seoul, Korea, 2013 July.

I-2) Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Christian Rossow "IoTPOT: Analysing the Rise of IoT Compromises", 9th Usenix Workshop on Offensive Technologies (WOOT' 2015), Washington D.C, United States, 2015 August.

Technical Reports

T-1) Yin Minn Pa Pa, Daisuke Makita, Katsunari Yoshioka, Tsutomu Matsumoto "Finding Malicious Authoritative DNS server", Information and Communication System Security (ICSS), Yokohama, Japan, 2013 March.

T-2) Daisuke Makita, Yin Minn Pa Pa, Katsunari Yoshioka, Tsutomu Matsumoto "A Method for Detecting Malware Infected Hosts with Similarity of Name Resolution Behavior", Symposium on Cryptography and Information Security (SCIS), Kyoto, Japan, 2013 January.

T-3) Katsunari Yoshioka, Yin Minn Pa Pa, Shogo Suzuki, Naoki Watanabe, Sou Nakayama, Toshiya Shimura, Haoyuan Xu, Junji Shikata, Tsutomu Matsumoto, Koji Nakao, Takeo Hariu, Makoto Iwamura, Takeshi Yagi, Mitsuaki Akiyama, Masato Terada, Shigeyoshi Shima, Masafumi Watanabe, Takahiro Kakumaru, Masaru Kawakita, Masahiro Yamada, Daisuke Inoue "Collection and Analysis of Cyber Security Data by Active and Passive Monitoring", Computer Security Symposium (CSS), Nagasaki, Japan, 2015 October.

T-4) Shogo Suzuki, Yin Minn Pa Pa, Yuta Ezawa, Sou Nakayama, Ying Tie, Katsunari Yoshioka, Tsutomu Matsumoto, "Improving IoTPOT for Observing Various Attacks Targeting Embedded Devices", Information and Communication System Security (ICSS), Kyoto, Japan, 2016, March.

List of Poster

P-1) Yin Minn Pa Pa, Katsunari Yoshioka, Tsutomu Matsumoto " Search Engine Based Investigation on Misconfiguration of Zone Transfer", International Workshop on Security (IWSEC-2012), Fukuoka, Japan, 2012 November.

Available Data Set

The following datasets are available upon request for interested researchers.

1. Malware binaries
2. IoTPOT traffic
3. List of compromised IoT devices visiting our IoTPOT.
4. Source Code

http://ipsr.ynu.ac.jp/iot/index.html#datasets

Appendix

Malware Binary List

Familly name	BinaryID	Filename	Hash(md5)	Architecuture	Date of Capture	Existance in VirusTotal	Detection Ration in VirusTotal	First sub.	Last sub.
	Bin 1	wb.arm	e94f48285ec44e739505889c922def55	ARM	2015/01	YES	0 / 56	1/12/2015 23:50	1/12/2015 23:50
	Bin 2	telnet.arm	4101d096094fa7f3b35a14cee8c5d6bb	ARM	2015/04	NO			
	Bin 3	telnet.m68k	2d4c6238ad43bfcc4668467ef6846196	M68K	2015/04	NO			
	Bin 4	telnet.mp	5c091a1c1311aa37443027a315b663f5	MIPS	2015/04	NO			
ZORRO	Bin 5	telnet.mps	acb79b0810aeb8e1db298cd678b33840	MIPSEL	2015/04	NO			
	Bin 6	telnet.ppc	8e654a673d4bdd8ac16c39f7a4654e1b	Power PC	2015/04	NO			
	Bin 7	telnet.sh4	60ee95389061b1c8ce0cf8b6f748c8a6	SH4	2015/04	NO			
	Bin 8	telnet.sparc	9918dba3e5737d25424b05b9f10b16c0	SPARC	2015/04	NO			
	Bin 9	telnet.x86	792d38b6fdd89d65d35d1b01cd1c2ba7	x86	2015/04	NO			
	Bin 10	arm	f73da5e1e33762f09d74e2d3d16c5c50	ARM	2014/11	YES	7 / 57	1/14/2015 18:30	1/14/2015 18:30
	Bin 11	i586	66113dc9a53866702ec0ca68a9a546b8	i586	2014/11	NO			
	Bin 12	i686	6d9f7123e8692087bdb2822e44854eef	x86	2014/11	NO			
	Bin 13	mips	c58e253607943555fc77c18b1688d4d01	MIPS	2014/11	YES	6 / 57	3/10/2015 8:41	3/10/2015 8:41
	Bin 14	mipsel	a265bab2443e0635a4adfe7f47e06974	MIPSEL	2014/11	NO			
	Bin 15	sparc	738db9f6b9debd08976eaa91bbf16117	SPARC	2014/11	NO			
	Bin 16	superh	a12e7f584177fb5d229707c5c7f7fa72	Super H	2014/11	NO			
	Bin 17	arm	06b2fbee4e7ae5c1370753543b7d2e21	ARM	2015/04	NO			
	Bin 18	i586	b7b299fdffbbaabd184ab4d8e69a4d98	x86	2015/04	NO			
	Bin 19	i686	4061432ae8b37171af033d5185b31659	x86	2015/04	NO			
	Bin 20	mips	3fc4bdb902e086e3e568179803620 7e7	MIPS	2015/04	NO			
	Bin 21	mips64	feb53f2aec98e96c1321a6811ac05a18	MIPS64	2015/04	NO			
	Bin 22	mipsel	94b2e00fc4c11abd77fb76fd5815d1dc	MIPSEL	2015/04	NO			
	Bin 23	ppc	06940d099751304c704f7a31c2459fb8	Power PC	2015/04	NO			
	Bin 24	sparc	d76cf4f0f37395906df4d2c0defcd923	Super H	2015/04	NO			
	Bin 25	arm	1549aed9b818b6a994dc5fb6c4a57fa2	ARM	2015/04	NO			
GAYFGT	Bin 26	i586	daab490a0a0a0a2b2528b18dacbf66ed	x86	2015/04	NO			
	Bin 27	i686	8a2b06d4ba8b88cab092801fbcbfd8b4	x86	2015/04	NO			
	Bin 28	mips	61f32f7a0d4b7643fb03da75cf5a1329	MIPS	2015/04	NO			
	Bin 29	mips64	ee7d764767c25d4c54be44f18a5aa47d	MIPS64	2015/04	NO			
	Bin 30	mipsel	490968447a603c3664186164c99c14be	MIPSEL	2015/04	NO			
	Bin 31	ppc	2695e6d6930fc3e5b3345f8cd811d693	Power PC	2015/04	NO			
	Bin 32	sparc	132c5605752c9cfcc3f746b8451c7fe6	Super H	2015/04	NO			
	Bin 33	arm	032ec8869e235bfa8a8dfe7b125a02b6	ARM	2015/05	NO			
	Bin 34	i586	86f9fc4e914d358d05bd5d1d93a0d673	x86	2015/05	NO			
	Bin 35	i686	c1ef1dd4232e14c45661e0a8a976867e	x86	2015/05	NO			
	Bin 36	mips	a41867fbf8e2358ba5551509907b288c	MIPS	2015/05	NO			
	Bin 37	mips32	77b73b0fe4a79dfc284fce55bf3cbe8b	MIPS32	2015/05	NO			
	Bin 38	mips64	d31261199d16b7ad82e0f87094de6e07	MIPS64	2015/05	NO			
	Bin 39	mipsel	c652fe5e53cba8c450ee6f7307408c8c	MIPSEL	2015/05	NO			
	Bin 40	ppc	52f9bd74d63888182fbab15443b70898	Power PC	2015/05	NO			
	Bin 41	sparc	be35cd9d4c6047e940e6c58a96fbf0b8	SPARC	2015/05	NO			
nttpd	Bin 42	nttpd	bbf1327c1a5213b41a4d22c4b4806f7c	MIPSEL	2015/05	YES	0 / 57	2/18/2015 17:24	3/20/2015 15:17
KOS	Bin 43	1225.8196	ec381bb5fb83b160fb1eb493817081c1	MIPS	2015/05	NO			
nttpd	Bin 44	nttpd	d97972cbfdf4207e5cb3a1615c6e4306		2015/06	NO			
	Bin 45	armp	dec3bf949c3b107dc3a973015269edd6	ARM	2015/06	NO			
	Bin 46	mipselp	67abda7e85c838448ca1f7915dfc6b17	MIPSEL	2015/06	NO			
	Bin 47	mipsp	de31e34c2e5f61980263547 04ac00e54	MIPS	2015/06	YES	2 / 57	6/2/2015 19:44	6/2/2015 19:44
	Bin 48	ppcp	4dcfba3c38863e647162ff81f37e8eb8	PPC	2015/06	YES	2 / 57	6/2/2015 19:40	6/2/2015 19:40
	Bin 49	shp	afcda120ec94869329e2b27a9c0e61fc	SH4	2015/06	YES	4 / 57	6/2/2015 19:35	6/3/2015 6:59
*.sh	Bin 50	armm	1c435276ffabe48d753527cccfe398a4	ARM	2015/06	YES	6 / 56	6/1/2015 7:48	6/1/2015 7:48
	Bin 51	mipselm	fc1e5c05fb6abc21f9075e13ca0bcc79	MIPSEL	2015/06	YES	3 / 56	6/1/2015 7:48	6/1/2015 7:48
	Bin 52	mipsm	1616d1cca4ccbca38f8948a42c99239c	MIPS	2015/06	YES	7 / 57	6/1/2015 7:49	6/5/2015 8:34
	Bin 53	ppcm	ac86a5a187f38d9d19c482bbbf24f148	PPC	2015/06	YES	2 / 56	6/1/2015 7:48	6/1/2015 7:48
	Bin 54	shm	d0173b706f9c65c1f011d4683a68217d	SH4	2015/06	YES	4 / 56	6/1/2015 7:47	6/1/2015 7:47
	Bin 55	568i	6bb6edd07979e547dc528a2143a9bf4f	x86	2015/06	NO			
	Bin 56	668i	3ead0f86731993fc8cf4f94159805990	x86	2015/06	NO			
	Bin 57	elimps	b5665875ae7eb40809384146a8bb6784	MIPSEL	2015/06	NO			
	Bin 58	husper	f17a8106fa6129c5aa7f374bed6f9276	Super H	2015/06	NO			
	Bin 59	mar	270307434ef97c688b831bc280671886	ARM	2015/06	NO			
	Bin 60	pcp	129b0be5bf9008095939db8da7c34d4e	Power PC	2015/06	NO			
GAYFGT	Bin 61	racps	b39b75d52dee457ccc825749226ec8e3	SPARC	2015/06	NO			
	Bin 62	sipm	5f6877670251458079 3aac478aadb811	MIPS	2015/06	NO			
	Bin 63	a	f47b27ed72f1a84f43d154399c04aca6	ARM	2015/06	YES	10 / 57	6/13/2015 15:16	6/13/2015 15:16
	Bin 64	m	33899bf414499403c3a53cd3b44d7a844	MIPS	2015/06	NO			
	Bin 65	mi	16679aa6674968494ae32f45fe2025e3	MIPSEL	2015/06	NO			
	Bin 66	p	0d52132275d204363df8b29eb379a2ea	Power PC	2015/06	NO			
	Bin 67	s	ffea6ec00f8ab522ee1e73ab8d4a936b	SH4	2015/06	NO			
	Bin 68	ayy.arm	112baeed64abe8f73e22664c53d30f40	ARM	2015/06	NO			
	Bin 69	ayy.m68k	6f35aefa8cd78b2c9ded814e0129bfd3	M68K	2015/06	NO			
	Bin 70	ayy.mp	20fb9b23986c922856d256f6321d2670	MIPS	2015/06	NO			
	Bin 71	ayy.m	70f75280ba31f993229db3ce1d06e698	MIPSEL	2015/06	NO			
	Bin 72	ayy.ppc	40c3e23080e1ad32c44118336e325d84	Power PC	2015/06	NO			
	Bin 73	ayy.sh4	e6ca89e393a6570a4a4c4208c36641f3	SH4	2015/06	NO			
	Bin 74	ayy.sparc	13ae92a808394938811e3711b2e9d5b4	SPARC	2015/06	NO			
ZORRO	Bin 75	ayy.x86	7df780f115cecd3219e7b0a55239abd4	x86	2015/06	NO			
	Bin 76	scanner.arm	14b32dd3d4dc8927c812c2eee6baa21e	ARM	2015/06	NO			
	Bin 77	scanner.m68k	63ecd54306c26d8f471bdb0a3ac0a651	M68K	2015/06	NO			
	Bin 78	scanner.mp	b147c04245d701669c89d6836a240c33	MIPS	2015/06	NO			
	Bin 79	scanner.mps	73ad21e470abad3da2acb39f621f6683	MIPSEL	2015/06	NO			
	Bin 80	scanner.ppc	56b0feec4e28276141ec0b93b6f21aaa	Power PC	2015/06	NO			
	Bin 81	scanner.sh4	493cb7e94f7073786b13ed0d93de0f4f	SH4	2015/06	NO			
	Bin 82	scanner.x86	fcc3292ffe2dc796573229b0d8d6d939	x86	2015/06	NO			
	Bin 83	a	bcb8f09861002f322e58697d1e1eb5f2	ARM	2015/06	NO			
	Bin 84	m	f81a141beed4f2ad86f96e6e9d219407	MIPS	2015/06	NO			
	Bin 85	mi	4062fd5532d6ece299ea33ddb3a9311d	MIPSEL	2015/06	NO			
	Bin 86	ppc	e6e8790bfccdb567b5713a8d2786c079	PPC	2015/06	NO			
	Bin 87	sh	2c514d5adb35d266b8b4774853f74021	SH4	2015/06	NO			
GAYFGT	Bin 88	armv6l	bec309444d23c6b2b6f3ced0bcd4b272	ARM	2015/06	NO			
	Bin 89	i686	e04781bd52095450259e0f3a3f986460	x86	2015/06	NO			
	Bin 90	mips	470a70b8dd9aa3b0f1ec36435abe96b7	MIPS	2015/06	NO			
	Bin 91	mipsel	2ef109f1b12493a3c4f6bb18f9c62784	MIPSEL	2015/06	NO			
	Bin 92	sh4	0310bf0e72f90c33838e0f0505b62758	SH4	2015/06	NO			
	Bin 93	x86_64	3f4dbbddbf3e1cb64ca43e55bb2027c1	x86	2015/06	NO			
	Bin 94	armm	0c2f8d1015101ac6fd7c3dc13bfdfe57	ARM	2015/06	NO			
	Bin 95	mipselm	ffa457c5a61bccb07ad5f8d0eae3b701	MIPSEL	2015/06	NO			
	Bin 96	mipsm	654ff5d3b63141a03176683ff753819d	MIPS	2015/06	NO			
	Bin 97	ppcm	b6dbd4429c86915af58fa414bbf5fc02	PPC	2015/06	NO			
	Bin 98	shm	3ebc1586ae4b91a537b5df84dd7d4a6c	SH4	2015/06	NO			
	Bin 99	niggerarm	fb7cefb47be606690c9d24708db7e435	ARM	2015/06	YES	5 / 57	6/22/2015 21:12	6/22/2015 21:12
*.sh	Bin 100	niggeri686	0e54692eed81cfc4435d52e2a60805e7	x86	2015/06	NO			
	Bin 101	niggermips	a0e8dae911ce7a8bcfcfe7c3d534573b	MIPS	2015/06	NO			
	Bin 102	niggermips64	761227176c4397dabc8763ded16c194d	MIPSEL64	2015/06	YES	1 / 57	6/22/2015 21:13	6/22/2015 21:13
	Bin 103	niggermipsel	a9c066dbb2205e12a69854f868a391ba	MIPSEL	2015/06	YES	5 / 56	6/22/2015 21:12	6/22/2015 21:12
	Bin 104	niggerppc	fcd714d5b9e099079b5bb3c17e76dcb1	PPC	2015/06	YES	3 / 57	6/22/2015 21:12	6/22/2015 21:12
	Bin 105	niggersh	566beee2814168801ee3662e53929624	Super H	2015/06	NO			
	Bin 106	niggerx86	8b0dbd88c7d90266f2db744adba668de	x86	2015/06	YES	3 / 55	6/26/2015 3:08	6/26/2015 3:08