IoTPOT: A Novel Honeypot for Revealing Current IoT Threats

YIN MINN PA PA, SHOGO SUZUKI†1
KATSUNARI YOSHIOKA, TSUTOMU MATSUMOTO†1
TAKAHIRO KASAMA†2,CHRISTIAN ROSSOW†3

We analyze the increasing threats against IoT devices. We show that Telnet-based attacks that target IoT devices have rocketed since 2014. Based on this observation, we propose an IoT honeypot and sandbox, which attracts and analyzes Telnet-based attacks against various IoT devices running on different CPU architectures such as ARM, MIPS, and PPC. By analyzing the observation results of our honeypot and captured malware samples, we show that there are currently at least 5 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.

Keyword: IoT, Honeypot

1. Introduction *

Since years, it is known that many Internet of Things (IoT) devices are vulnerable to simple intrusion attempts, for example, using weak or even default passwords [1]. In 2012, Carna botnet [2] revealed that there were more than 1.2 million open devices that allowed logins with empty or default credentials. In January 2014, an Internet-connected fridge was discovered as a part of a botnet sending over 750,000 spam e-mails [3]. In December 2014, online DDoS services (i.e. booters) knocked down Sony and Microsoft's gaming networks, presumably powered by thousands of compromised IoT devices such as home routers [4]. From an attacker's point of view, IoT devices are attractive playgrounds, as–as opposed to PCs–they are 24/7 online, have no antivirus installed, and weak login passwords give attackers an easy access to powerful shells (such as BusyBox [5]). Seeing these trends, we believe that IoT devices are an important new area of security research.

In this paper, we investigate the threat of IoT device compromises in the masses. We first analyze Telnet-based scans in darknet, revealing that attacks on Telnet have rocketed since 2014. Moreover, by grabbing Telnet banners and web contents of the attackers, we show that the majority of attacks indeed stem from IoT devices.

Motivated by this, we propose IoTPOT, a novel honeypot to emulate Telnet services of various IoT devices to analyze ongoing attacks in depth. IoTPOT consists of a frontend low-interaction responder cooperating with backend high-interaction virtual environments called IoTBOX. IoTBOX operates various virtual environments commonly used by embedded systems for different CPU architectures. During 81 days of operation, we observed 481,521 download attempts of malware binaries from 79,935 visiting IP addresses. We also confirm that none of these binaries could have been captured by existing honeypots that handle the Telnet protocol such as honeyd and Telnet password honeypot because they are not able to handle different incoming commands sent by the attackers.

We manually downloaded 106 distinct malware samples and found out that they run on 11 different CPU architectures. Among 106 collected samples, 88 samples were new to the database of VirusTotal [6] (as of 2015/06/26) showing a gap of capturing utilities for this type of threat. Out of 18 samples in VirusTotal, 2 of them were not detected by any of the 57 antivirus software of VirusTotal (as of 2015/06/26).

In order to analyze these captured malware binaries, we propose IoTBOX, the first malware analysis environment for IoT devices. IoTBOX supports 8 CPU architectures, spanning MIPS, ARM, and PPC. The sandbox analysis of 25 samples by IoTBOX revealed that the samples are used to perform 11 different types of DDoS attacks, port 23 scans and scans on UDP (port 123, 3143) and TCP (port 80,8080,5916).

Finally, combining the observations results of IoTPOT with the sandbox analysis by IoTBOX, we confirm that i) there are at least five distinct malware families spreading via Telnet, ii) their common behavior is performing DDoS and the further propagation over Telnet, iii) some families evolve quickly, updating frequently and shipping binaries for a variety of CPU architectures, even in the limited observation period of 81 days.

Following is the summary of our contributions:

1) We point out a huge increase of Telnet-based attacks and the involvement of IoT devices.
2) To analyze the scope and variety of the attacks, we propose a novel honeypot called IoTPOT, which mimics IoT devices and captures Telnet-based intrusions.
3) We further analyze the threats and propose IoTBOX, which enables us to run the captured malware on 8 different CPU architectures.
4) We reveal that there are at least five DDoS malware families targeting IoT devices.
5) We analyze the architectures of IoT botnets and point out that there are at least 8 different types of botnet

* †1 Yokohama National University,Japan
†2 National Institute of Information and Communications Technology,Japan
†3 Saarland University, Germany.

Part of the results of this paper will be presented at the 9th USENIX Workshop on Offensive Technologies (WOOT'15).

architectures including the worm type botnet.

The rest of the paper is organized as follows: Sect. 2 explains our preliminary investigations on Telnet-based attacks. Sect. 3 describes IoTPOT and Sect. 4 IoTBOX. In Sect. 5, we describe the overview of ongoing attacks revealed by our analysis. In Sect. 6, related works are presented. Finally, in Sect. 7 the conclusion and future works are explained.

2. Investigation on Telnet-based Attacks

Until now, there are only anecdotal reports on Telnet-based compromises. In this section, we investigate how the situation of Telnet-based compromises has changed. To this end, we analyze a darknet of NICTER [7], Japan's darknet monitoring system that monitors over 209,000 IP addresses presently.

Figure 1 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 [8], the large peak in the end of 2012 is caused by the activities of the Carna botnet, created by an anonymous hacker for Internet Census by compromising a large number of IoT devices such as routers [2]. Since 2014, even after the deactivation of the 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 [9] 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 the Internet using the HTTP title as keyword 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 the DVR, 16 models of IP Camera, 45 models of wireless routers. Moreover, devices such as a metrological satellite, heat pumps, a parking management system, a fire alarm system, solid state recorders and a TV have scanned our darknet on 23/TCP. Table 1 shows top ten attacking hosts and device models of inferred device types. These results show that various IoT devices are already involved in the ongoing attacks.

3. IoT Honeypot (IoTPOT)

Our preliminary investigation on Telnet-based attacks implies that there are a number of IoT devices being compromised and

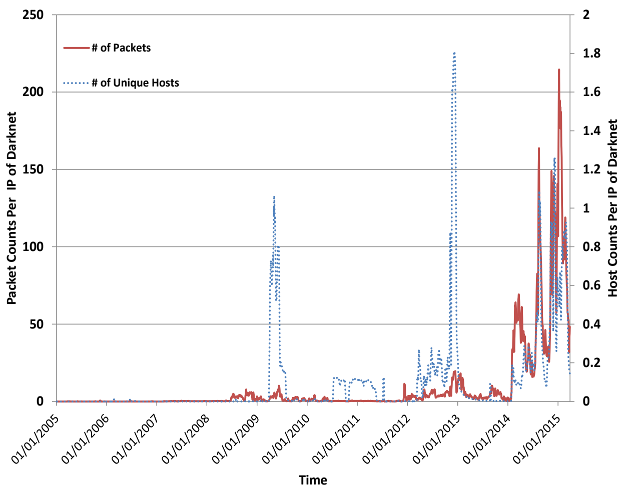

Figure 1 - Packets and hosts on 23/TCP per day per darknet IP

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
Environment Monitoring Unit (EMU) System	9	1
Digital Video Scalar	5	2
Router	4	3

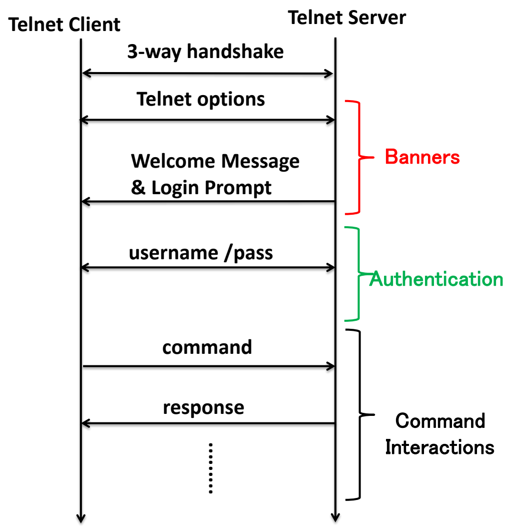

Figure -2 Telnet Protocol

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 the Telnet protocol and a variety of IoT devices.

3.1 Telnet Protocol

Before explaining IoTPOT, we briefly revisit the Telnet protocol [10]. Figure 2 illustrates the interactions between client and the server on Telnet. After the TCP 3-way handshake, the client and the server can exchange Telnet options. Either the Telnet server or the 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 the login prompt. For example, “BCM96318 Broadband Router” as the welcome message and “Login:” as the 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**.

3.2 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 a 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 collected these banners from the Internet by performing Telnet scans with the masscan tool [11]. From all collected banners, we prioritized 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 the 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 a command interaction, the 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 a different CPU architecture, we prepare a set of embedded Linux OS on different CPU architectures to handle the interactions of various devices.

3.3 IoTPOT Implementation

Figure 3 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, a welcome message, and a login prompt. Authentication profiles determine how to respond to incoming authentication challenges. The 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 an 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 a device manual and the maker’s website. If we cannot find explicit CPU information of a particular IoT device, we refer to the list of applications for each CPU architecture [12][13][14][15][16].

Frontend Responder forwards a response from IoTBOX to the client. Note that the incoming commands forwarded to IoTBOX may cause malware infections or a system alteration. Therefore, we reset the OS image occasionally. Moreover, IoTBOX in IoTPOT is used as a 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 4.

The *Profiler* parses the interaction between *Frontend Responder* and IoTBOX, extracts the incoming command and the 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 on 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 the visitor scan mode, it connects back to hosts who visit our honeypot and grabs 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 [17].

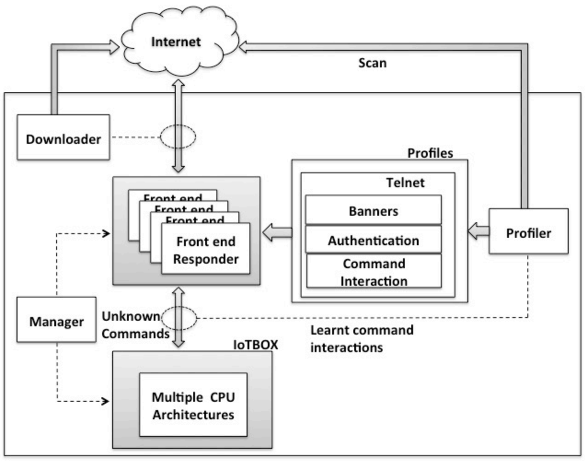

Figure 3 - Overview of IoTPOT

3.4 Observation Results

IoTPOT setup: We operated 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 had tried different configurations, device profiles, and assignment of IP addresses in a 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 deployed IoTPOT on 87 IP addresses, used 29 banner profiles assigning each to three IP addresses. We set authentication profiles to accept any challenges and prepared a single command interaction profile, manually created from one of the most widely exploited DVR brands [18]. The backend IoTBOX contained an embedded Linux OSs of Debian [19] and OpenWrt [20] on 8 different CPU architectures emulated by QEMU [21]. 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 visited IoTPOT. Among them, 130,314 successfully logged in and 79,935 attempted to download external malware binary files. We observed 481,521 download attempts in total. We manually downloaded 106 malware binaries of 11 CPU architectures. Among 106 collected samples, 88 samples were new to the database of VirusTotal (as of 2015/06/26). Out of 18 samples that were in VirusTotal, 2 of them were not detected by any of the 57 antivirus software of VirusTotal (as of 2015/06/26).

General Flow of Telnet Attacks: We observed 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 the intrusion and the infection by IoTPOT and after malware binaries are captured by IoTPOT, we conduct a 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. 5.1.

3.4.1 Stage 1: Intrusion

We recognize two major intrusion behaviors: login attempts with a fixed or a random order of credentials. Table 2 shows the major login patterns observed by IoTPOT. Fixed challenge order, "Fixed Order", in Table 2 means attackers try to login to IoTPOT with a sequence of id and password pairs in a fixed order. For example, in the case of a 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 2 is quite a long list, thus, we show only top sequences. Random challenge order means attackers try to login to IoTPOT with a sequence of id and password pairs in a random order. Thus, in case of "Random Order 1", it is not always true that "root/admin" will come after "root/root".

Table 2 - 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	Random Order	root/root root/toor root/admin root/user root/guest root/login root/changeme ....

3.4.2 Stage 2: Infection

After successfully logged in to honeypot, attackers check and customize the environment to prepare the download of a malware binary by sending a series of commands over Telnet. Table 3 summarizes the 10 major patterns of command sequences observed by IoTPOT. Note that some of the patterns were 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 the characteristic string it contains. For example, the patterns named ZORRO 1, ZORRO 2 and ZORRO 3 all have the common string "ZORRO" in their command sequences. Moreover, we can see the 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 a customized command for downloading an external malware binary file. With this setup, other intruders would have difficulty to abuse the system. A similar intension of attackers can be seen in the case of a pattern named GAYFGT. Although it does not alter the commands, instead it activates iptables [17] to drop incoming telnet connection requests. GAYFGT also has a 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 found is checking whether the 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, the attacker stops the attacks.

Comparison with honeyd: We confirmed that honeyd [22] cannot handle these commands in Table 3 and therefore cannot capture malware binaries observed by IoTPOT. Namely, honeyd failed to respond to the very first few commands such as "cat /bin/sh" in case of the ZORRO family and appropriate reply for the first echo command of GAYFGT, nttpd and KOS family and so the attacker stopped 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 classified 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 4 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 increased 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 the same group if the command sequence from an attacker is similar to the previously categorized 43 samples.

Table 3 - 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

Table 4 - Clustering results of collected samples by characteristic strings in the binaries

<table>
<tr><th>Family Name</th><th>Common Strings in Binaries</th></tr>
<tr><td>Bin 1 - Bin 9</td><td>YESHELLO
killattk</td></tr>
<tr><td>Bin 10 to Bin 41</td><td>SCANNER ON | OFF
bin.sh
bin2.sh
bin3.sh
echo -e '\x67\x61\x79\x66\x67\x74'</td></tr>
<tr><td>Bin 42</td><td>sh -c "cd /tmp ; rm -f .nttpd ; wget -O .nttpd http://%d.%d.%d.%d:%d ; chmod +x .nttpd ; ./.nttpd"</td></tr>
<tr><td>Bin 43</td><td>0916.davinci
0923.davinci
0923.8196</td></tr>
</table>

3.4.3 Stage 3 Monetization

IoTPOT can only observe intrusion and infection stages explained in 3.4.1 and 3.4.2. 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 the Appendix. The sandbox analysis results of some of the binaries are described in Section 4.

4. IoT Sandbox (IoTBOX)

IoTBOX is used not only as high interaction systems in IoTPOT but also as a stand-alone multi-architecture sandbox. The design of IoTBOX used for two purposes is the 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 a stand-alone multi-architecture sandbox in section 4.1.

4.1 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 4. To support different CPU architectures, we need a cross compilation environments. We thus choose to run respective platforms (OS) on an emulated CPU using QEMU [21], 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) [20]. 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 a single physical machine. Thus we need a virtual network environment in order to connect a physical interface of host machine with many virtual interfaces of QEMU based virtual machines. The following explains how we create a 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 the bridge and many different virtual interfaces (eth0 of each virtual machine) at the other side of the bridge. In order to create a virtual switch, we first create a virtual interface br0. As we want host only network, we do not bridge br0 with eth0 right now.

Normally, the br0 interface does not need an IP address as it is supposed to function as a 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 an IP address to it. We assign br0 to a local IP address, which will be the gateway of all virtual machines.

We then try to connect br0 with virtual machines so that packets from a virtual machine can reach br0 and vice versa. But, virtual machines' NIC (eth0 in each virtual machine of Figure 4) 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 the 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 a virtual bridge connection of br0 to virtual NICs through tap interfaces by using Linux brctl [24]. We automate all these steps so that the 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 about layer 2 level connections. From the viewpoint of layer 3, the br0 interface will be the same network with all virtual machines and it will be the gateway for all virtual machines. The interface, eth0 of host machine will be on a 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 a stand-alone multi-architecture sandbox: In this case, *Access Controller* controls NAT and outbound traffic from each virtual machine such as C&C communication, the DNS resolution and the attack traffic such as DoS. We block all outgoing DoS traffic from malware except allowing some DNS and HTTP traffic of a maximum of 5 packets per minute. 23/TCP scans are redirected to *Dummy Server*, which is indeed IoTPOT. In this way, we can monitor how the propagation over Telnet is done.

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

IoTBOX as a high interaction system in IoTPOT: In this case, *Access Controller* will accept only an 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 3 is doing is exactly the same as *Access Controller* we have discussed here.

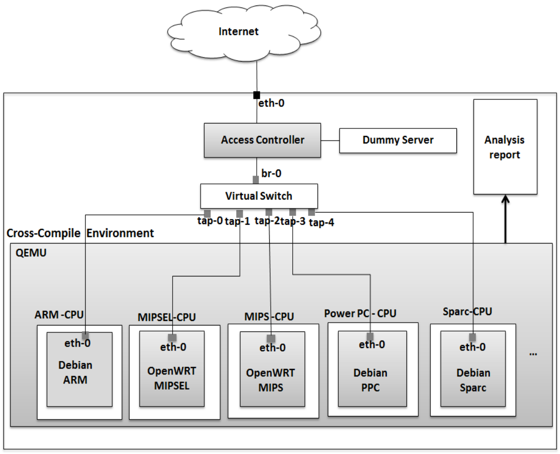

Figure 4 - Overview of IoTBOX

4.2 Analysis Results by IoTBOX

Using IoTBOX, we analyzed 52 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 focused more in analysis. Please refer to Appendix for the information of analyzed malware samples. Red colored samples show analyzed binaries.

We observed 25 of 52 malware binaries performed 11 different types of DoS attacks and 3 different types of scans such as the 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 5. Most attacks we observed were UDP floods and many different types of TCP floods. We also observed UDP floods against multiple destination ports, which seemed to aim at flooding the target network. Interestingly, we also observed a DNS water torture attack [25], SSL attacks [26] and other two unknown DNS based attacks in which a large number of queries to an unknown type of DNS resource records (RR) were sent to an authoritative name server of a popular ISP. Sample Bin 43 exhibits a 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 a 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 Appendix 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 during the infection phase to prevent other attackers from compromising the host.

5. Analysis on Attacks

5.1 Overview of Observed Attacks

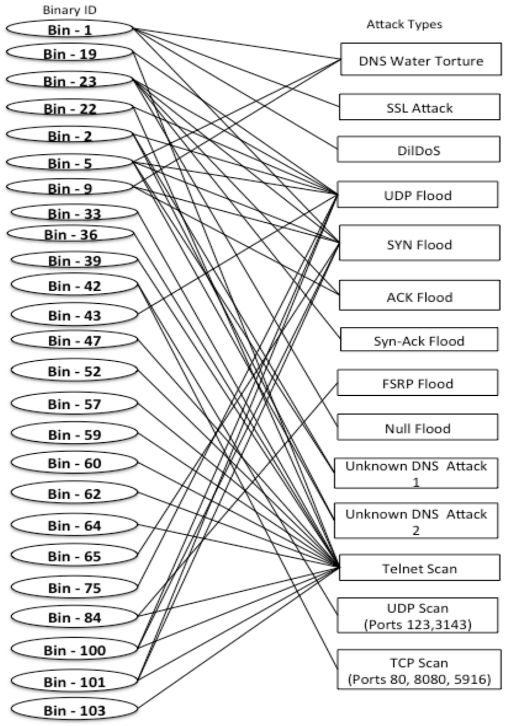

Figure 5 - Observed attacks by IoTBOX

Figure 6 depicts the overview of Telnet-based attacks observed by IoTPOT and IoTBOX. In order to understand the 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, the intrusion pattern "Fixed Order 3", which is shown in Table 2, is always followed by the infection pattern "ZORRO 4", explained in Table 3. 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 a 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 to 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, the 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.

Following are our findings.

1) We have observed five malware families whose intrusion, infection, and malware binaries are independent from each other.
2) 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.

3) Some families seem to spread more aggressively than others. Namely, as in Figure 6, ZORRO, GAYFGT and nttpd familes have updated command sequences twice during the observation period. Also, the GAYFGT family has increased the diversity of binaries to support more CPU architectures.

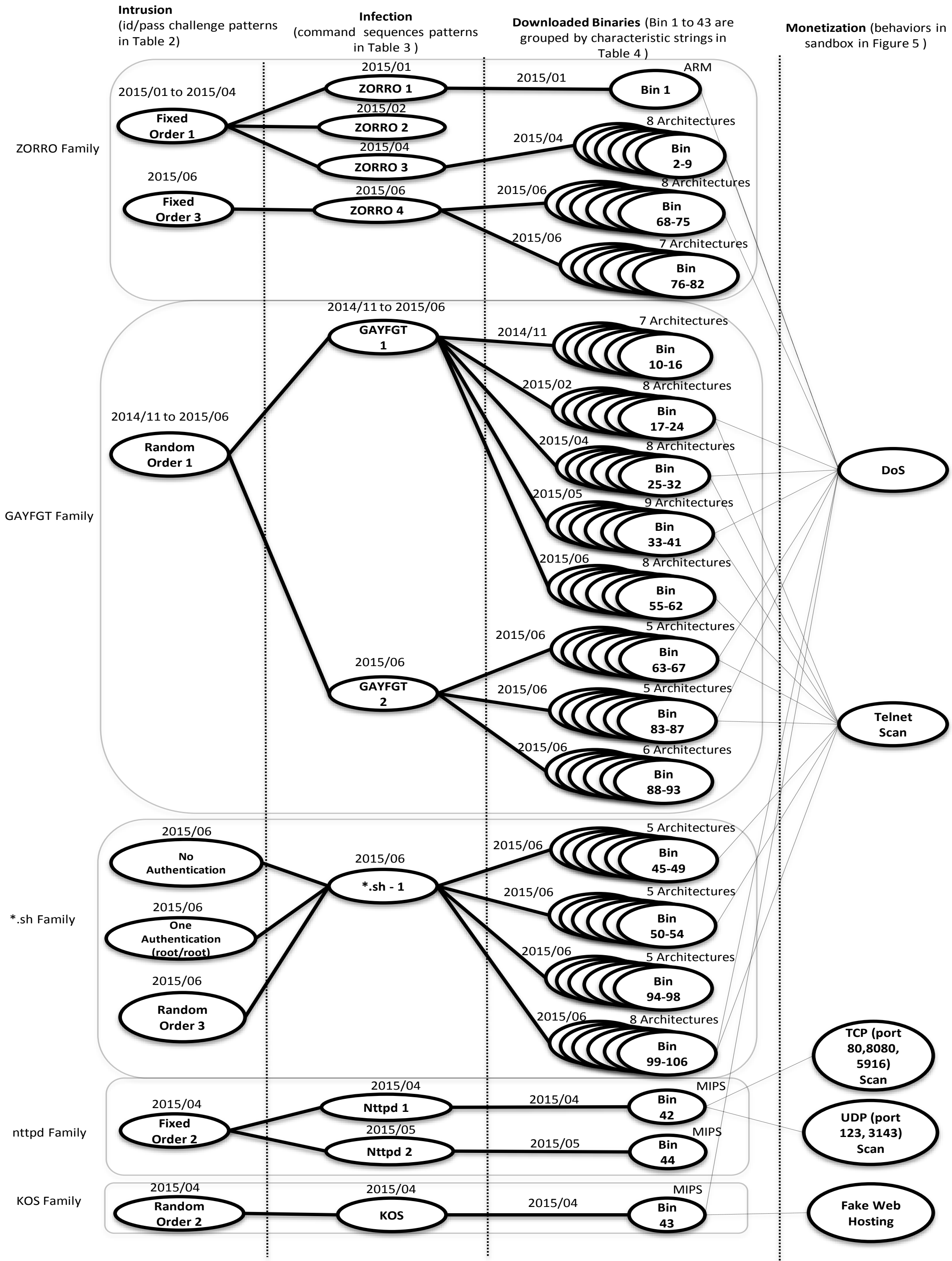

Figure 6 - Overview of Observed Attacks by IoTPOT and IoTBOX

5.2 Overview of an Attacking Botnet

5.2.1 Botnet Architectures

Figure 7 shows the overview of a 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, the intruding host (I) intrudes the victim sending a sequence of commands over Telnet in order to make the victim machine download the malware binary from a malware download server (D). Downloaded binary is run and after the infection, the 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 the 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 follows:

1) Botnet relating to the ZORRO family has many host performing scanning only and few I, D and C of different combinations (B1, B2, B3 of Figure 7).
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 7).
3) The propagation of the nttpd family looks alike warm infection in which the attacking host itself is a scanner, an intruder and a malware download server (B6 in Figure 7). There are also cases in which the scanning and the intruding host make victim infect by sending malware binary over Telnet. In such a case, it is not necessary to download malware binary from a malware download server (B7 in Figure 7).
4) The botnet of KOS family has many hosts performing both scanning and intruding while D and C are separate hosts (B8 of Figure 7). C can be connected by resolving the "s6.kill123.com" domain. In order to resolve the domain, the 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. In this way, attacker set up an authoritative name server as part of his or her botnet.

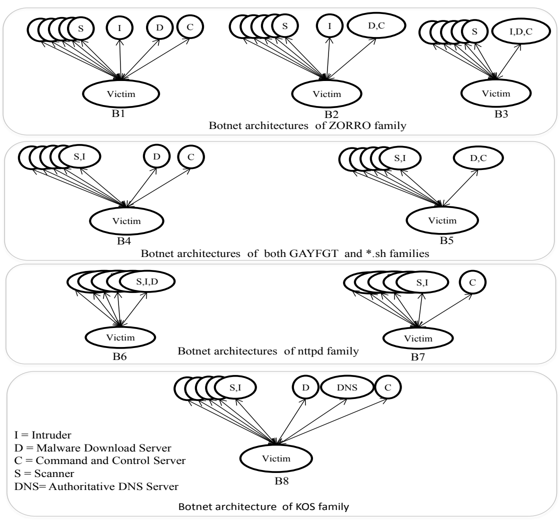

Figure 7 - Botnet Architectures

6. Related Works

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 [22] 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 the Telnet password honeypot [27], its main focus is collecting Telnet password and command interactions are not supported. Other popular low interaction honeypots such as Dionaea [28] and Nepenthes [29] do not support Telnet. Kishimoto et al. [30] propose a novel honeypot that dynamically assigns an IPv6 address to appropriate high interaction honeypots by checking the destination IP address of an incoming NS message which includes the vendor information. SGNET [31] is a honeypot system that has distributed low-interaction sensors to handle known attacks. Its centralized backend high-interaction honeypots 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 to 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 the functionality to perform an automated active scanning of 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 focus on Telnet attacks which are emerging threats according to the recent observations of darknet as explained in Sect. 2, emulate the Telnet services of a large variety of IoT devices to attract attacks, and succeed 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 runs 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.

The main differences of the proposed method against existing works are as follow:

1) IoTPOT implements the functionality to perform an automated active scanning of 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.

2) Although the mechanism is similar to existing honeypots, we are the first to focus on a 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 a detailed analysis.

3) We propose IoTBOX, a multi-architecture malware sandbox that is used as a high interaction system as a component of IoTPOT and also independently used as a malware sandbox for analyzing captured binaries.

4) 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 the existing malware database of Virus Total. We also reveal their monetization behaviors and architectures as botnet.

7. Conclusion and Future Works

We have shown that IoT devices are susceptible to compromises and increasingly are also the target of malware on the masses. We identified five 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 of 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.

Reference

[1] A. Cui and S. Salvatore J., "A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan." [Online]. Available: http://ids.cs.columbia.edu/sites/default/files/paper-acsac.pdf. [Accessed: 24-May-2015].

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

[3] "DailyTech - Hackers Use Refrigerator, Other Devices to Send 750,000 Spam Emails." [Online]. Available: http://www.dailytech.com/Hackers+Use+Refrigerator+Other+Devices+to+Send+750000+Spam+Emails+/article34161.htm. [Accessed: 24-May-2015].

[4] "Lizard Stresser Runs on Hacked Home Routers — Krebs on Security." [Online]. Available: http://krebsonsecurity.com/2015/01/lizard-stresser-runs-on-hacked-home-routers/. [Accessed: 24-May-2015].

[5] "BusyBox." [Online]. Available: http://www.busybox.net/. [Accessed: 30-Oct-2015].

[6] "VirusTotal - Free Online Virus, Malware and URL Scanner." [Online]. Available: https://www.virustotal.com/. [Accessed: 24-May-2015].

[7] 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.*

[8] 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.

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

[10] "RFC 854 - Telnet Protocol Specification." [Online]. Available: https://tools.ietf.org/html/rfc854. [Accessed: 24-May-2015].

[11] "robertdavidgraham/masscan · GitHub." [Online]. Available: https://github.com/robertdavidgraham/masscan. [Accessed: 24-May-2015].

[12] "List of applications of ARM cores," *Wikipedia, the free encyclopedia*. 30-Sep-2015.

[13] "List of MIPS microarchitectures," *Wikipedia, the free encyclopedia*. 17-Sep-2015.

[14] "PowerPC applications," *Wikipedia, the free encyclopedia*. 18-Dec-2012.

[15] "SuperH," *Wikipedia, the free encyclopedia*. 30-Sep-2015.

[16] "SuperH RISC engine Family | Renesas Electronics." [Online]. Available: http://www.renesas.com/products/mpumcu/superh/index.jsp. [Accessed: 03-Nov-2015].

[17] "netfilter/iptables project homepage - The netfilter.org project." [Online]. Available: http://www.netfilter.org/. [Accessed: 03-Nov-2015].

[18] "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].

[19] "Index of /~aurel32/qemu/mipsel." [Online]. Available: https://people.debian.org/~aurel32/qemu/mipsel/. [Accessed: 04-Nov-2015].

[20] "OpenWrt." [Online]. Available: https://openwrt.org/. [Accessed: 30-Oct-2015].

[21] "QEMU." [Online]. Available: http://wiki.qemu.org/Main_Page. [Accessed: 30-Oct-2015].

[22] "Developments of the Honeyd Virtual Honeypot." [Online]. Available: http://www.honeyd.org/. [Accessed: 24-May-2015].

[23] "Linux Bridge and Virtual Networking - Blogs by Sriram." .

[24] "brctl." [Online]. Available: http://linuxcommand.org/man_pages/brctl8.html. [Accessed: 02-Nov-2015].

[25] Secure64, “Water Torture: A Slow Drip DNS DDoS Attack « Cybersecurity « Cyber Trust Matters.” .
[26] “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].
[27] “zx2c4/telnet-password-honeypot · GitHub.” [Online]. Available: https://github.com/zx2c4/telnet-password-honeypot. [Accessed: 24-May-2015].
[28] “dionaea — catches bugs.” [Online]. Available: http://dionaea.carnivore.it/. [Accessed: 24-May-2015].
[29] “home [Nepenthes - finest collection -].” [Online]. Available: http://nepenthes.carnivore.it/. [Accessed: 24-May-2015].
[30] 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.
[31] 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.
[32] “malware.dvi - malware_survey.pdf.” [Online]. Available: https://iseclab.org/papers/malware_survey.pdf. [Accessed: 04-Nov-2015].

Yin Minn Pa Pa received B.E. in Information Technology in 2006 from Mandalay Technological University, Myanmar and M.Phil. in Infrastructure Management in 2013 from Yokohama National University, Japan. She is currently Ph.D. candidate of Information Media and Environment Science Course of Graduate School of Environment and Information Sciences, Yokohama National University. She is going to finish her Ph.D. in March 2016. Her research interest is network security. She received best paper award in Asia-JCIS, 2013 and best technical report award of the year by Information and Communication System Security (ICSS) 2013, Japan.

Shogo Suzuki is currently second year Master student of Information Media and Environment Science Course of Graduate School of Environment and Information Sciences, Yokohama National University. He is going to finish his M.E in Computer Engineering in March 2016. His research interest is network security.

Takahiro Kasama received his B.E. and M.E. and Ph.D. degrees in Computer Engineering from Yokohama National University in 2009 and 2011, 2014, respectively. He is currently a researcher at the National Institute of Information and Communications Technology, Japan. His research interest covers a wide area of network security including network monitoring and malware analysis. He received the Best Paper Award at the Computer Security Symposium 2010 (CSS2010), and the IPSJ Yamashita SIG Research Award in 2011.

Katsunari Yoshioka received the B.E., M.E. and Ph.D. degrees in Computer Engineering from Yokohama National University in 2000, 2002, 2005, respectively. From 2005 to 2007, he was a Researcher at the National Institute of Information and Communications Technology, Japan. Currently, he is an Associate Professor for Division of Social Environment and Informatics, Graduate School of Environment and Information Sciences, Yokohama National University. His research interest covers a wide range of information security, including malware analysis, network monitoring, intrusion detection, and information hiding. He was awarded 2009 Prizes for Science and Technology by The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology.

Tsutomu Matsumoto is a professor of the Graduate School of Environment and Information Sciences, Yokohama National University and directing the Research Unit for Information and Physical Security at the Institute of Advanced Sciences. He received Doctor of Engineering from the University of Tokyo in 1986. Starting from Cryptography in the early 80’s, he has opened up the field of security measuring for logical and physical security mechanisms. Currently he is interested in research and education of Embedded Security Systems such as Smartcards, Network Appliances, Mobile Terminals, In-vehicle Networks, Biometrics, and Artifact-metrics. He is serving as a program officer of the JSPS Research Center for Science Systems, the chair of Japanese National Body for ISO/TC68 (Financial Services), and a core member of the Cryptography Research and Evaluation Committees (CRYPTREC). He was a director of the International Association for Cryptologic Research (IACR) and the chair of the IEICE Technical Committee on Information Security and served as an associate member of the Science Council of Japan (SCJ). He received the IEICE Achievement Award, the DoCoMo Mobile Science Award, the Culture of Information Security Award, the MEXT Prize for Science and Technology, and the Fuji Sankei Business Eye Award.

Dr. Christian Rossow graduated in Computer Science in 2013 at the VU Amsterdam, The Netherlands. Since 2014, he leads the "System Security" Research Group at Saarland University, Germany. He also holds a Guest Associate Professorship at YNU University, Japan. His research focuses are binary analysis, malicious software and network security.

Familly name	BinaryID	Filename	Hash(md5)	Architecuture	Date of Capture	Existance in VirusTotal	Detection Ration in VirusTotal	First sub.	Last sub.
ZORRO	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			
	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			
GAYFGT	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	c58e25360794355fc77c18b1688d4d01	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	3fc4bdb902e086e3e5681798036207e7	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			
	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			
*.sh	Bin 45	armp	dec3bf949c3b107dc3a973015269edd6	ARM	2015/06	NO			
	Bin 46	mipselp	67abda7e85c838448ca1f7915dfc6b17	MIPSEL	2015/06	NO			
	Bin 47	mipsp	de31e34c2e5f6198026354704ac00e54	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
	Bin 50	armm	1c435276ffabe48d753527cccfe398a4	ARM	2015/06	YES	6 / 56	6/1/2015 7:48	6/1/2015 7:48
	Bin 51	mipselm	fe1e5c05fb6abe21f9075a13ea0bec79	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
GAYFGT	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			
	Bin 61	racps	b39b75d52dee457ccc825749226ec8e3	SPARC	2015/06	NO			
	Bin 62	sipm	5f68776702514580793aac478aadb811	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	33899bf41499403c3a53cd3b44d7a844	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			
ZORRO	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			
	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			
GAYFGT	Bin 83	a	bcb8f09861002f322e56697d1e1eb5f2	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			
	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			
*.sh	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
	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	a9c066dbb2205e12a69854f668a391ba	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