IoTPOT: Analysing the Rise of IoT Compromises

Yin Minn Pa Pa[†1], Shogo Suzuki[†1], Katsunari Yoshioka[†1], Tsutomu Matsumoto[†1],
Takahiro Kasama[†2], Christian Rossow[†3]
[†1]Graduate School of Environment and Information Sciences/Institute of Advanced Sciences
[†1] Yokohama National University, Japan
[†2]National Institute of Information and Communications Technology, Japan
[†3]Institute of Advanced Sciences, Yokohama National University, Japan and
[†3]Cluster of Excellence, MMCI, Saarland University, Germany

Abstract

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 4 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.

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 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 easy access to powerful shells (such as BusyBox). 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 39 days of operation, we observed 76,605 download attempts of malware binaries from 16,934 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.

We manually downloaded 43 distinct malware samples and found out that they run on 11 different CPU architectures. Among 43 collected samples, 39 samples were new to the database of VirusTotal [5] (as of 2015/05/13) showing a gap of capturing utilities for this type of threat. Out of 4 samples that were in VirusTotal, 2 of them were not detected by any of the 57 A/Vs of VirusTotal (as of 2015/05/13).

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 17 samples by IoTBOX revealed that the samples are used to perform 10 different types of DDoS attacks and port 23 scans.

Finally, combining the observations results of IoTPOT with the sandbox analysis by IoTBOX, we confirm that i) there are at least four distinct malware families spreading via Telnet, ii) their common behavior is performing DDoS and 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 39 days.

The dataset of captured malware, traffic of IoTPOT, list of compromised IoT device types from our darknet analysis are available upon request for interested researchers [6]. We are going to make IoTPOT open source after finishing all documentation processes.

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 captured malware on 8 different CPU architectures.
4) We reveal that there are at least four DDoS malware families targeting IoT devices.

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 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 270,000 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 Carna botnet, created by anonymous hacker for Internet Census by compromising a large number of IoT devices such as routers [2]. 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 [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 DVR (Digital Video Recorder). 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, 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.

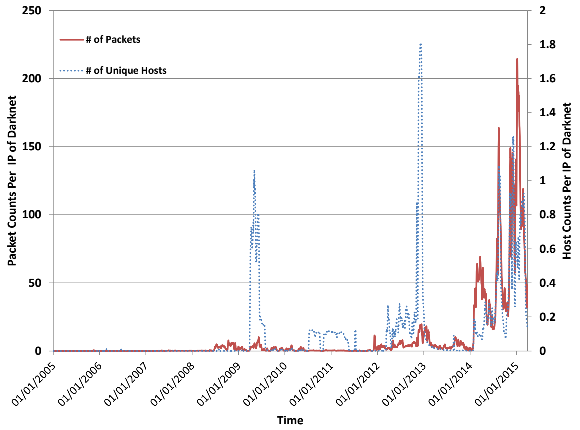

Figure 1 - Packets and hosts on 23/TCP per day per darknet IP

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
EMU System	9	1
Digital Video Scalar	5	2
Router	4	3

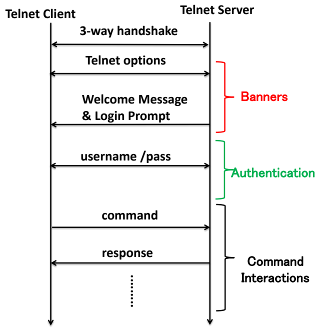

Figure 2 - Telnet Protocol

3. IoT Honeypot (IoTPOT)

Our preliminary investigation on Telnet-based attacks 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.

3.1 Telnet Protocol

Before explaining IoTPOT, we briefly revisit the Telnet protocol [10]. Figure 2 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**.

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 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 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 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, IoTPOT chooses from a set of environments during the command interactions. 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.

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, 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. The detailed explanation of IoTBOX is in Section 4. *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.

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 only connects back to hosts who visit our honeypot.

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, the Manager handles configuration of IoTPOT. Namely, it links IP addresses to specific *Device Profiles*.

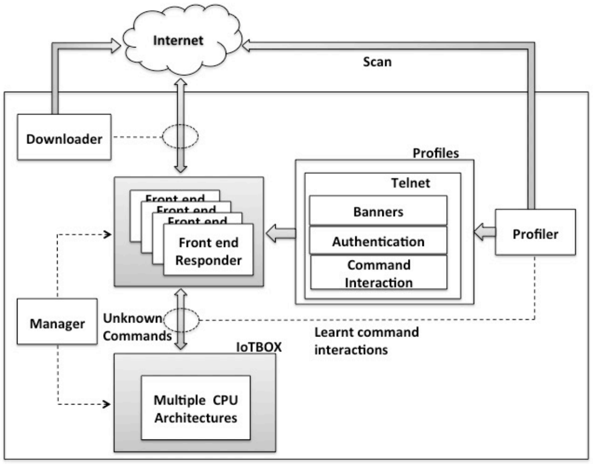

Figure 3 - Overview of IoTPOT

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 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 2015/05/09, we deployed IoTPOT on 165 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 [12]. The backend IoTBOX contained an environment that runs Linux for embedded devices on 8 different CPU architecture created by OpenWRT. Downloader was not fully implemented so we manually downloaded and collected malware binaries.

Summary of Observations: During 39 days of the stable operation, 70,230 hosts visited IoTPOT. Among them, 49,141 successfully logged in and 16,934 attempted to download external malware binary files. We observed 76,605 download attempts in total. We manually downloaded 43 malware binaries of 11 CPU architectures. Among 43 collected samples, 39 samples were new to the database of VirusTotal [5] (as of 2015/05/13). Out of 4 samples that were in VirusTotal, 2 of them were not detected by any of the 57 A/Vs of VirusTotal (as of 2015/05/13).

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. 2) The second stage is **infection**, in which attackers send a series of commands over Telnet to check and customize the environment and download and execute the external binaries. 3) The third stage 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. The following subsections highlight some points noticed for each attack stage.

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 four login patterns observed by IoTPOT. For the fixed login sequences, we can reasonably infer that these challenges are from malware sharing the same implementation of dictionary attacks.

Table 2 - Major log in patterns observed by IoTPOT.

Pattern Name	Challenge Order	Username/Pass	Number of Observed Attacks Per Day (Average)
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	174
Random Order 1	Random Order	root/root root/admin root/12345 root/123456 admin/root admin/admin support/support …	606
Fixed Order 2	Fixed Order	admin/admin admin/362729 admin/m4f6h3 admin/n3wporra admin/263297 admin/fdpm0r admin/1234 root/1234 …	3.2
Random Order 2	Random Order	root/xc3511 root/123456 root/12345 root/root …	3.5

3.4.2 Stage 2: Infection

After successfully logged in to honeypot, attackers check and customize the environment to prepare download of malware binary by sending series of commands over Telnet. Table 3 summarizes the 6 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 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 Bashlite. Although it does not alter the commands, instead it activates iptables to drop incoming telnet connection requests. Bashlite also has functionality to kill other existing malicious processes. All these activities explained above come in a form of commands over Telnet except that Bashlite 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 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 confirmed that honeyd [13] cannot handle these commands in Table 3 and therefore cannot capture malware binaries observed by IoTPOT. Namely, honeyd failed 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 Bashlite, nttpd and KOS family and so the attacker stopped sending any further commands.

3.4.3 Stage 3: Monetization

Finally, the attacker tries to monetize the compromised devices. We thus analyzed the 43 malware binaries collected by IoTPOT. We show the list of samples in Appendix. The sandbox analysis results of some of the binaries are described in Section 4. As none of the collected samples are obfuscated, we classified the binaries based on the hardcoded strings, such as strings for C&C commands. Table 4 summarizes results of manual clustering of the collected samples based on the characteristic strings in the binaries.

Table 3 - Patterns of command sequences observed by IoTPOT

Pattern Name	Pattern of Command Sequence	Set of Command Sequence per Day (Average)
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.	174
	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	1,353
Bashlite	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.	606
nttpd	1. Check whether shell can be used or not by echoing “welcome” 2. Download binary to /tmp directory. 3. Run Binary.	3.2
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	3.5

The column with # in Table 3 indicates that the patterns are observed only in trial period and no credible counts of the patterns can be provided. Steps 1 - 4 of ZORRO 1, ZORRO 2, and ZORRO 3 are done by a group of reconnaissance hosts and Steps 5 - 9 are done by a single intrusion host repeatedly. See Section 3.4.4 for details.

Table 4 - 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

3.4.4 Coordinated intrusions

In the trial period, we noticed a coordinated intrusion by ZORRO family, in which reconnaissance and the actual malware infection were 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 without downloading nor executing any malware binary file. After a while, we observed another host who visited our honeypot and successfully logged in with just one challenge implying that it already knew the valid credential from the earlier reconnaissance. This intrusion host then sent series of commands to download and execute external malware binary. The downloaded binary file was indeed of the CPU architecture of the honeypot and so we think that this host knew the CPU architecture of the honeypot from the reconnaissance.

We then set a new login credential and kept observation. We had 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 visited us again with the newly obtained credential and infected the malware. After all, we observed a group of over 100 reconnaissance hosts and only a single intrusion host in coordination. Figure 4 depicts the coordinated attack.

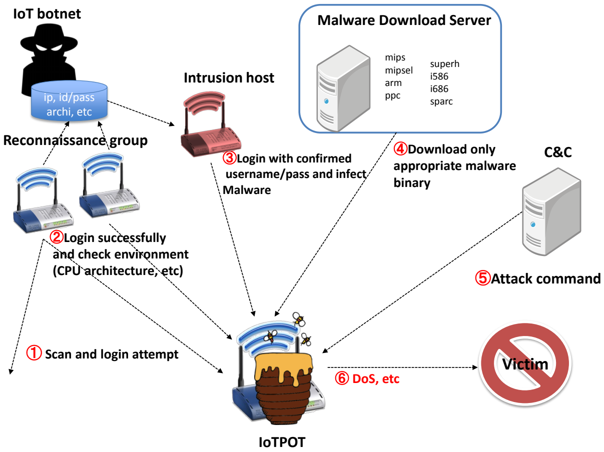

Figure 4 - Coordinated attack of ZORRO family observed by IoTPOT

4. IoT Sandbox (IoTBOX)

IoTPOT has shown that there is a clear rise of Telnet-spreading malware that has already compromised thousands of IoT devices. In this section, we present our multi-architecture sandbox called IoTBOX.

4.1 IoTBOX Design

IoTBOX supports analysis of malware on 8 different CPU architectures, namely as MIPS, MIPSEL, PPC, SPARC, ARM, MIPS64, sh4 and X86. The design of IoTBOX is shown in Figure 5. To run malware binaries of different CPU architectures, we need a cross compilation environments. We thus chose to run respective platforms (OS) on an emulated CPU using QEMU, 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 (typically wireless routers) [14]. 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.

Finally, the *Access Controller* controls all network related operations such as NAT and outbound traffic 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.

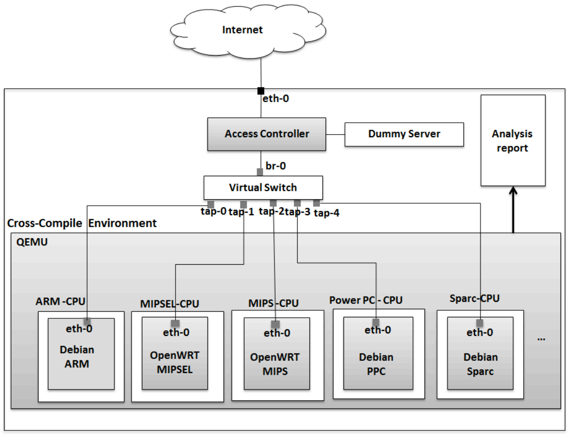

Figure 5 - Overview of IoTBOX

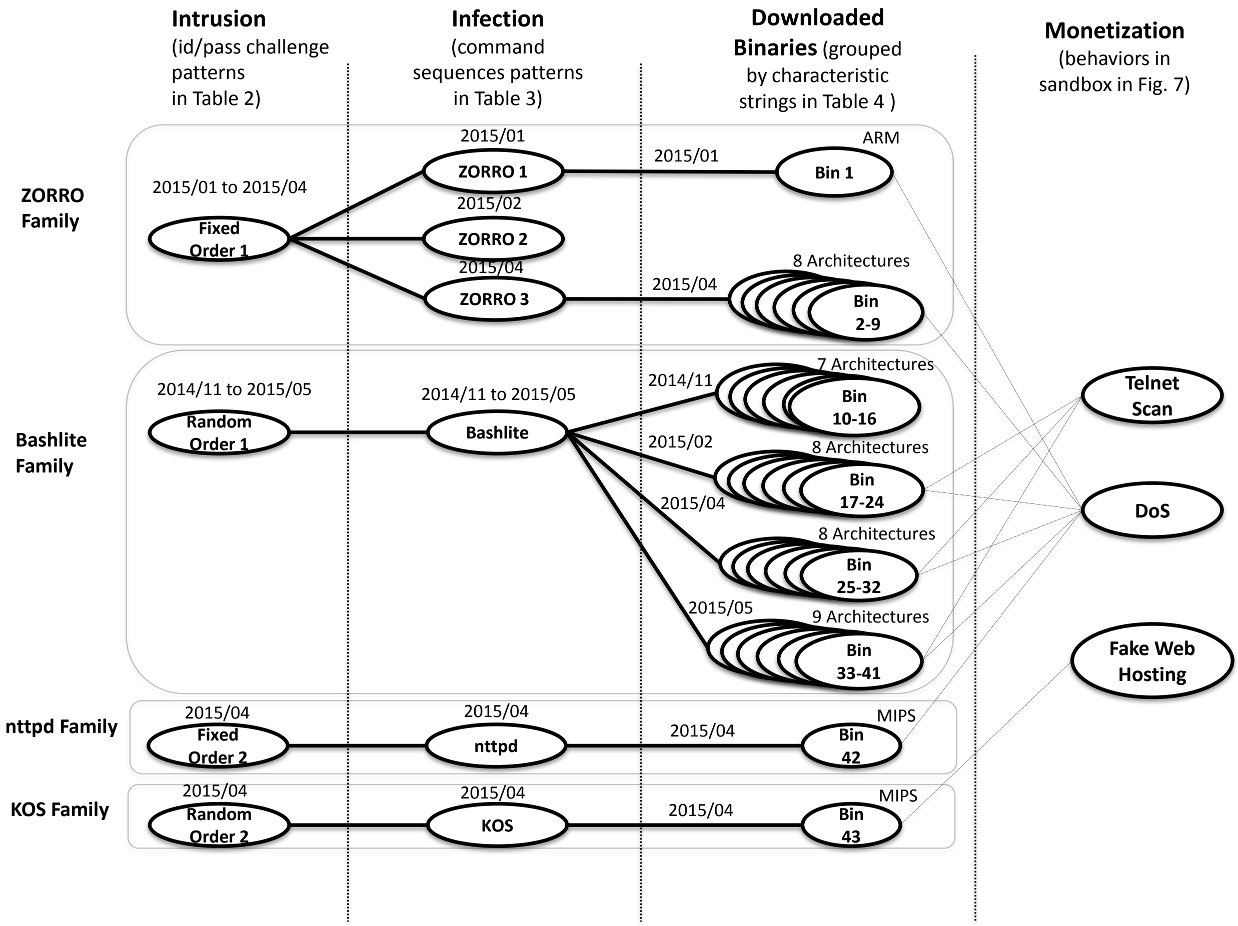

Figure 6 - Overview of Observed Attacks by IoTPOT and IoTBOX

4.2 Analysis Results

Using IoTBOX, we analyzed 17 malware binaries of 8 CPU architectures. We observed 8 of them performed 10 different types of DoS attacks and 2 performed 23/TCP scans. Please refer to Appendix for the information of analyzed malware samples. A summary of the observed attacks is illustrated in Figure 7. 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 target network. Interestingly, we also observed DNS water torture attack [15], SSL attacks [16] and other two unknown DNS based attacks in which a large number of queries to unknown type of DNS resource records (RR) were 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 7 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 [17] during the infection phase to prevent other attackers from compromising the host.

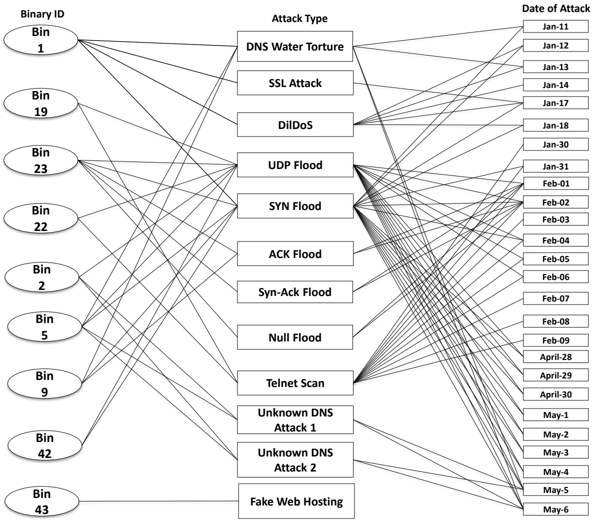

Figure 7 - Observed attacks by IoTBOX

5. Overview of Observed Attacks

Figure 6 depicts the overview of Telnet-based attacks observed by IoTPOT and IoTBOX. Following are our findings.

1) We have observed four 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 Telnet 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 family has updated its command sequences twice during observation period. Also, the Bashlite family has increased the diversity of binaries to support more CPU 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 [13] 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 [18], its main focus is collecting Telnet password and command interactions are not supported. Other popular low interaction honeypots such as Dionaea [19] and Nepenthes [20] do not support Telnet.

We also implemented IoTBOX, the first sandbox that handle to run malware of different CPU architectures. Out of more than 15 surveyed sandbox systems in [21], none supports different CPU architecture such as MIPS, ARM.

7. Conclusion and Future Works

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

Acknowledgement

A part of this was conducted under the auspices of the MEXT Program for Promoting Reform of National Universities and supported by PRACTICE (Proactive Response Against Cyber-attacks Through International Collaborative Exchange) project by the Ministry of Internal Affairs and Communications, Japan.

References:

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

[6] “IoT Security - Research Center for Information and Physical Security” [Online]. Available: http://ipsr.ynu.ac.jp/iot/index.html. [Accessed: 24-May-2015].

[7] M. Eto, D. Inoue, J. Song, J. Nakazato, K. Ohtaka, and K. Nakao, “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 Secure.*

[8] E. L. Malécot, and D. Inoue, “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] “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].

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

[14] “OpenWrt.” [Online]. Available: https://openwrt.org/. [Accessed: 24-May-2015].

[15] Secure64, “Water Torture: A Slow Drip DNS DDoS Attack «Cybersecurity «Cyber Trust Matters.” .

[16] “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].

[17] “netfilter/iptables project homepage - The netfilter.org project.” [Online]. Available: http://www.netfilter.org/. [Accessed: 24-May-2015].

[18] “zx2c4/telnet-password-honeypot · GitHub.” [Online]. Available: https://github.com/zx2c4/telnet-password-honeypot. [Accessed: 24-May-2015].

[19] “dionaea — catches bugs.” [Online]. Available: http://dionaea.carnivore.it/. [Accessed: 24-May-2015].

[20] “Nepenthes - finest collection -” [Online]. Available: http://nepenthes.carnivore.it/. [Accessed: 24-May-2015].

[21] “malware.dvi - malware_survey.pdf"

Appendix

Note that all samples in the table and corresponding honeypot traffic are available for interested researchers upon request [6].

Malware binary files captured by IoTPOT

Familly name	Sample ID	File Name	Hash(md5)	Architecuture	Date of Capture	Existance in VirusTotal	Detection Ratio in VirusTotal
ZORRO	Bin 1	wb.arm	e94f48285ec44e739505889c922def55	ARM	2015/01	yes	0 / 56
	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	
Bashlite	Bin 10	arm	f73da5e1e33762f09d74e2d3d16c5c50	ARM	2014/11	yes	7 / 57
	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
	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
KOS	Bin 43	1225.8196	ec381bb5fb83b160fb1eb493817081c1	MIPS	2015/05	no	