iBoT: IoT Botnet Testbed

Adam Beauchaine, Miles Macchiaroli, and Mira Yun
Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, MA 02115, USA
{beauchainea, macchiarolim, yunm}@wit.edu

2021 16th International Conference on Computer Science & Education (ICCSE) | 978-1-6654-1468-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCSE51940.2021.9569298

Abstract— Security in Internet-of-Things (IoT) devices has become increasingly relevant in the recent years. One of the driving reasons for the increased attention in this field is the fast-paced propagation of IoT botnet attacks. Botnet cyber-attacks refer to large scale distributed systems that maliciously exploit devices to perform tasks, such as DDoS attacks. IoT devices make easy attack vectors for such a system, given their tendency to be lightweight and insecure. Provided the increasing gravity of the subject, we have proposed a solution to produce a realistic testbed for the study of IoT botnets with low-cost and off-the-shelf devices. This testbed integrates layer 3 connectivity as well as home and business network topology. Special emphasis has been placed upon network and endpoint logging in this design, and testing has been documented with multiple styles of attacks.

Keywords—Botnet, Internet of Things, Testbed, DDoS attacks

I. INTRODUCTION

Internet-of-Things (IoT) devices have become instrumental to the quality of both business and home computing environments. The number of IoT devices is anticipated to rise to over 30 billion by 2030, and these low cost and latency devices often utilize wireless technologies to send and receive data for both operational and administrative purposes [1,2]. With such a sizable installation base, security flaws in IoT devices can prove to be detrimental on a widespread scale. IoT botnets comprise some of the most severe threats to these Ad-Hoc networks, as IoT infrastructure may provide ample computational resources for malicious tasks such as distributed denial-of-service (DDoS) attacks [2]. The *Mirai* botnet attack on *DYN* (now *Oracle*) domain name servers (DNS) of 2016 was one such example where a multitude of IoT devices such as cameras, refrigerators, and thermostats were utilized to bring down a major internet service [3].

The increased targeting of IoT devices should come as no surprise due to the often-insufficient security standards leveraged by manufacturers. Devices often fail to receive security updates, and implementing cryptography often proves too computationally expensive due to hardware restraints [1,2]. In addition, many insecure protocols such as telnet, file transfer protocol (FTP), and simple network management protocol (SNMP) are left enabled for the purposes of remote administration. This has proven to be a major challenge for the security industry due to a lack of adjustment in these development philosophies [1-3].

Despite the growing demand from security professionals for new IoT solutions, it is hard to find a hands-on security program or training for undergraduate students. Since it is clearly shown that hands-on activities promote active learning, and aid to bridge the gap between conventional concept and practice, we propose an IoT botnet testbed, *iBoT,* to enable students and researchers to better understand the process of botnet infection, traffic flows, and common security pitfalls in IoT systems. By leveraging low-cost and off-the-shelf devices, we present instructions for constructing an IoT botnet testbed that includes both home and enterprise configurations. In addition, we display methods to effectively utilize our testbed for the study of botnet attacks.

The rest of this paper organized as follows: Section II presents current methodologies for classifying botnet administrative infrastructure and attack variants, as well as similar efforts to research IoT botnets in a controlled setting. Our *iBoT* design, hardware used, and software configurations are detailed in Section III. Section IV provides the details about an enterprise deployment and logic for its utilization. Finally, we share the documented results of a test attack on our testbed and suggest future improvements to the logging and technical capabilities of this system.

II. IOT BOTNETS AND INFRASTRUCTURE

Botnets broadly refer to a distributed system of networked computers infected and controlled by a single master [4,5]. These installments are typically self-propagating, with individual hosts designed to spread infection across a network. This makes botnets particularly worrisome for larger organizations, as their larger number of assets makes the potential impact of such botnets far more damaging [4]. Typical botnet applications include email spamming, phishing, identity theft, and DDoS attacks. Given the security issues of IoT devices, and their increasing prevalence in home and enterprise networks, they have become ideal targets for botnet attacks, typically dedicated to DDoS attacks [3]. This arrangement is evidenced by the successful incidents of botnet brute force attacks in the past, as well as the simplicity of brute-force attack execution, examples being DDoS and password cracking. These attacks are so effective, even simple designs that lack synchronization methods can be catastrophic [6]. This allows for even the unsophisticated botnets to serve as an easy way for attackers to gain illicit access or bring down resources, as such the increased notoriety botnets are receiving in the security field is well warranted.

Botnet attacks are typically categorized into two methods based on their control systems: peer-to-peer (P2P) and command-and-control (C&C) [3-5]. P2P botnets function in a

non-hierarchical way control is dispersed among all nodes of a botnet network. C&C botnets feature a single point of administration and control for the botnet, typically referred to as the C&C server. C&C has been chosen for simulation in this installment, as C&C botnets are a far more prevalent threat in most organizations today [4]. In addition, C&C botnets offer a simplified model in comparison to most P2P deployments, allowing a better understanding of botnet infrastructure from the perspective of a learner.

There are currently several testbeds that have been utilized for security research including *Emulab*, a virtualized networking testbed often used in distributed systems research [7]. A fully contained testbed for analyzing IoT botnets has been developed based on this platform [3]. This contained testbed provides a toolkit for *Emulab*-enabled testbeds, and includes additional services such as DHCP and DNS, enabling the deployment of botnets that rely on such services. A testbed developed specifically for botnet testing such as a virtualized botnet attack on HTTP services has also been developed [8]. This HTTP-based botnet *(HBB)* testbed operates in a traditional C&C structure and explores various HTTP-based attacks such as HTTP-GET flooding.

Both *Emulab* and *HBB* testbed accurately represent the recent advancements in botnet testbed development, however, to the best of our knowledge, no botnet testbed has demonstrated comprehensive logging capabilities, and a realistic, layer 3 topological design for both home and enterprise networks. By constructing a testbed that successfully executes these prerequisites, our *iBoT* provides a way to thoroughly examine C&C botnet infrastructure at a network and endpoint level. This should ultimately provide a realistic environment in which to study botnet attacks.

In the interest of providing a feature-rich environment to simulate botnet attacks, our *iBoT* features layer 3 connectivity across several network segments to best replicate home and business network configurations. With proper logging, these features allow for easy to access network data to allow researchers to easily identify anomalous botnet traffic, as well as botnet endpoint activities in an IoT environment.

III. *iBoT*

Our initial testbed design, as shown in Fig. 1, featured three separate network segments, each representative of a different part of a botnet system, including a target web server, C&C server, and a home IoT network design for bots. The use of a wireless gateway in conjunction with separate router interfaces for independent nodes provides a comparatively realistic design. It also enables the ability for logging to take place independently on any zone in the system with firewall settings or packet captures during a botnet attack.

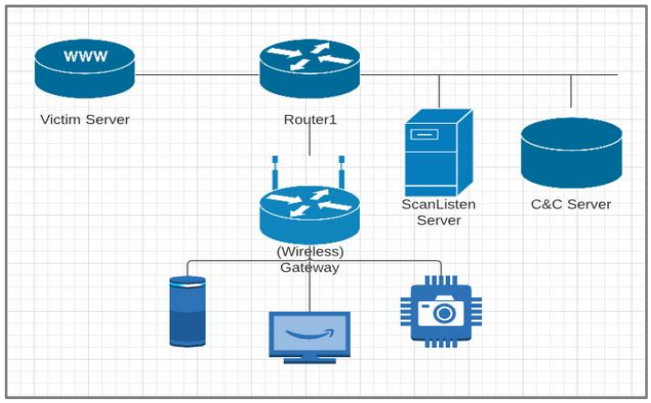

Fig. 1. *Home Testbed Design*

A. *Endpoint Configuration*

IoT devices are typically lightweight, networked machines utilized for simple computational tasks. For our testbed of IoT devices, we opted to use a Raspberry Pi 3, and two Raspberry Pi 2s, as detailed in Table 1. These devices are leveraging a linux-based operating system (OS) for security configurations, as well as wireless connectivity. For accuracy, devices were tested via wireless as it represents typical IoT deployment, although wired connectivity is also available.

TABLE I. ENDPONT DEVICES

Device	SoC	GPU	RAM	Networking	Storage
Raspberry Pi 3	Broadcom BCM2837	4x ARM Cortex-A5 3, 1.2GHz	1GB LPDDR 2 (900MHz)	10/100 Ethernet, 2.4GHz 802.11n wireless	micro SD
Raspberry Pi 2 x 2	Broadcom BCM2837	900MHz quad-core ARM cortex-A7 CPU	1 GB SDRAM	10/100 Mbit/s Ethernet. USB Wireless Adapter	micro SD

All Raspberry Pi devices were reformatted with Raspberry Pi OS (32 bit) and configured with *XRDP* to allow for remote management via Windows machines natively [10]. Firewall rules were updated to allow for remote connections to be made over port 3389. As shown in Fig. 2, ports 22 and 23 (SSH and Telnet) were left open to simulate the poor port security of many IoT devices in home networks.

```
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
23                         ALLOW       Anywhere
Anywhere                   ALLOW       192.168.0.15 3389
22 (v6)                    ALLOW       Anywhere (v6)
23 (v6)                    ALLOW       Anywhere (v6)
```

Fig. 2. UFW Endpoint Firewall Rules

For the purposes of logging, the honeypot software *Cowrie* is installed on all endpoint devices, which acts as a telnet and SSH enabled endpoint [11]. This allows for superior logging when a botnet is deployed, as all C&C commands are executed via telnet. Fig. 3 details telnet configurations to the endpoints, and the commands executed are saved to a log file. This is particularly useful as any malicious action will be documented to this file when executed. These transactions can be viewed anytime with a built-in play log utility. This configuration process is replicated on each endpoint device.

```
# ====================================
# Proxy Configurations
# ====================================

# real credentials to log into backend
backend_user = root
backend_pass = root

# Telnet prompt detection
#
# To detect authentication prompts (and spoof auth details to the ones the
# login and password prompts, and spoof data to the backend in order to suc
# attackers can only use the real user credentials of the backend.
telnet_spoof_authentication = true

# These regex were made using Ubuntu 18.04; you have to adapt these for the
# from your backend. You can enable raw logging above to analyse data passi
# and identify the format of the prompts you need.
# You should generally include ".*" at the beginning and end of prompts, si
# more data than the prompt.

# For login it is usually <hostname> login:
telnet_username_prompt_regex = (\n|^)ubuntu login: .*

# Password prompt is usually only the word Password
telnet_password_prompt_regex = .*Password: .*

# This data is sent by clients at the beginning of negotiation (before the
# that is trying to log in. We replace that username with the one in "backe
# login after the first password prompt. We are only able to check if crede
# inserted. If they are, then a correct username was already sent and authe
# password to force authentication to fail.
telnet_username_in_negotiation_regex = (.*\xff\xfa.*USER\x01)(.*?)(\xff.*)
```

Fig. 3. Telnet Proxy Configuration

Endpoint devices were connected to a gateway router initially via Category 6 ethernet cables. Wireless connectivity was also implemented and functions similarly at a configuration level. All devices exist logically within the same local network segment and they are always visible to each other. Reserved IPv4 addresses were assigned via DHCP to each endpoint device for consistency and simplification of networking configurations as shown in Fig. 4.

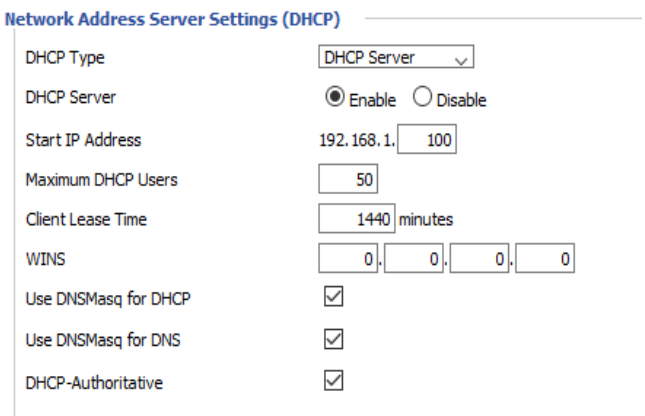

Fig. 4. *Endpoint DHCP Settings*

B. Gateway Configuration

The wireless gateway leveraged for our deployment is a Linksys WRT54GL wireless router operating at 2.4 GHz frequency band, and compatibility with 802.11b/g wireless protocol enabled. An integrated 4-port switch for wired connectivity is also available. The Gateway router is configured using DD-WRT Firmware [12], with major adjustments made that allow wireless access to the administrative portal, as well as the reservation of the 192.168.1.101-103 IP range for static client addresses. Routing configuration is kept to gateway mode, enabling network address translation (NAT) for clients, with port forwarding being an option for deployment if necessary. The stateful firewall was left enabled with default settings, and anonymous WAN requests also allowed for the purposes of testing.

Wireless settings were unchanged, with the sole exception of enabling WPA2-Personal security mode. All access point login requests as well as stateful traffic sessions are logged directly in-router as shown in Fig. 5. Logging capabilities represent the main reason DD-WRT is leveraged, as opposed to the factory firmware. Wide area network connectivity was configured as a static route, with interface and gateway addresses existing as 192.168.5.100 and 192.168.5.1 respectively. Due to a lack of any upstream Internet service in this topology, this change is necessary for the testbed to function.

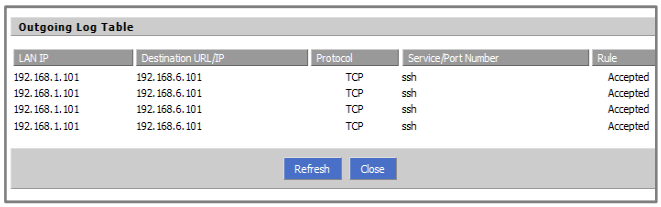

Fig. 5. *SSH Session Logging*

C. Router Configuration

iBoT has layer 3 topological design to allow for IoT devices to exist on separate networks, better representing the typical way in which a botnet is spread. *iBoT* has a Cisco 2911 Integrated Services Router. This device features 3 integrated ethernet ports out of the box, which enables the connectivity of all network segments without the need for additional router expansion modules. All configurations and management were done out of band via the integrated console port.

The configuration of the router was intended to provide simple layer 3 service, while allowing for additional router settings such as access control list (ACL) functionality to be added later. The settings configured are taken from Cisco's recommended base configuration for the device, an enable password is configured, and the hostname was changed from the default. Routing was configured via static routes for simplicity due to only having 3 network segments within the testbed. No ACLs were configured, and all non IoT devices can communicate without hindrance. IP addressing scheme has shown in Fig. 6.

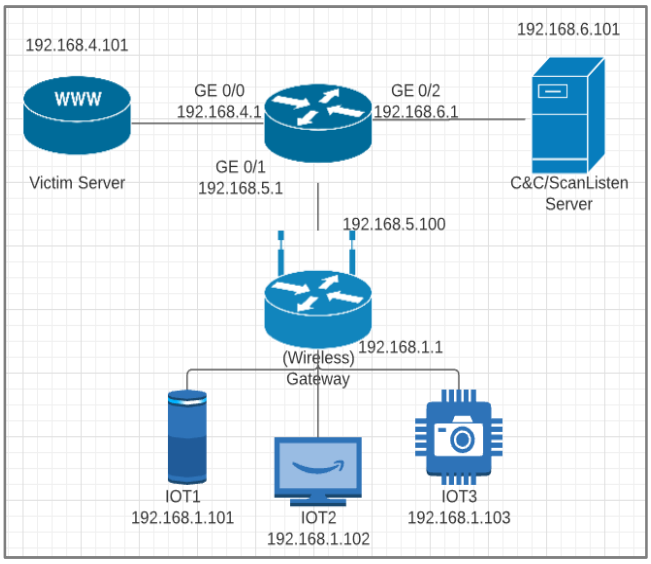

Fig. 6. Fully networked home testbed

D. Web Server Configuration

Our web server was a simple Apache HTTP Server deployed on a MacBook Pro machine running macOS 10.14 Mojave. Configuration was kept explicitly simple; PHP scripting is enabled but not utilized. This is simply due to Apache default configuration, as well as an effort to simplify web deployment. For web server logging, Wireshark was installed locally on the MacBook. No other security measures were implemented, and all network segments could access the test web server. Fig. 7 shows our test webpage viewed from a client.

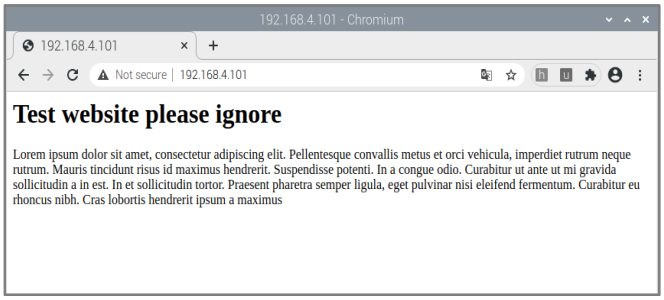

Fig. 7. Test web server page

E. C&C / Botnet Configuration

Our botnet server operates on a virtualized Centos 7 server instance leveraging VirtualBox as a hypervisor. A typical full installation featuring GNOME desktop software was performed, in addition to the VirtualBox guest tool suite. These alterations were performed to simplify configuration and file transfer to the guest instance. A Bridged network adapter was used to fully simulate an independent network device. A separate interface was configured with host wireless NAT connectivity for internet access to install necessary packages.

When selecting which botnets to deploy, ease of configuration and technical relevance were both important factors to consider. As such, the first botnet, *Vivid*, is a newer variant of the older *Mirai* botnet software, that specifically targets Linux machines. Additionally, *UFONet*, a botnet-like tool for DDoS attack testing, was also deployed [13]. *UFONet* is unlike a traditional C&C botnet, in that it customarily targets open redirect vectors on websites to act similarly to more traditional instances. Fig. 8 shows the main page of *UFONet* command center.

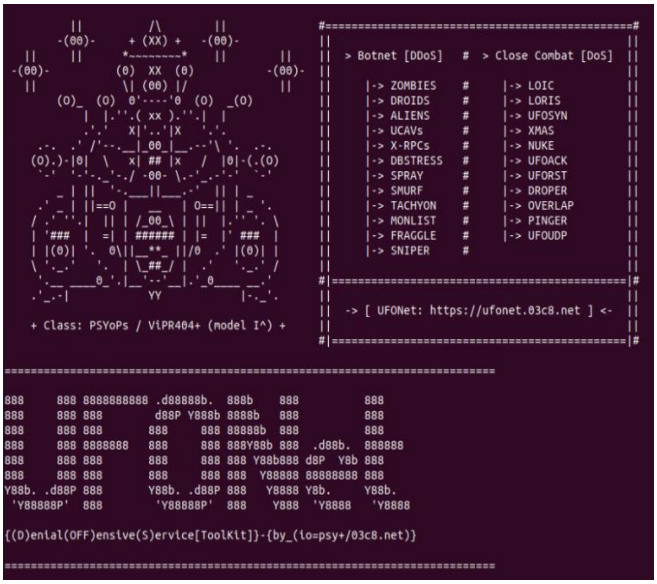

Fig. 8. UFONet Command Center

Both botnet attacks were configured and ran successfully within the test environment, beginning with *Vivid*, and later moving onto *UFONet*. *Vivid*, the *Mirai*-based botnet, is the more traditional of the two in relation to its functionality as well as control structure. Payload text files are configured within the C&C server and sent out to networked devices as linux commands to be ran over open telnet ports and subsequently infect more linux machines. An example can be seen below in Fig. 9.

```
root@CommandControl:~/Loader# cat PAYLOAD.txt
cd /tmp || cd /run || cd /; wget http://192.168.6.101/EkSgbins.sh;
 chmod 777 EkSgbins.sh; sh EkSgbins.sh; tftp 192.168.6.101 -c get
EkSgtftp1.sh; chmod 777 EkSgtftp1.sh; sh EkSgtftp1.sh; tftp -r EkS
gtftp2.sh -g 192.168.6.101; chmod 777 EkSgtftp2.sh; sh EkSgtftp2.s
h; rm -rf EkSgbins.sh EkSgtftp1.sh EkSgtftp2.sh; rm -rf *root@Comm
andControl:~/Loader# c
```

Fig. 9. Set of payload commands run upon infection

Commands may be issued with one of several options in the administrative server console, these include telnet and SSH brute forcing to peers, as well as several types of flood attacks that can be executed. One advantage of *Vivid* over other *Mirai* variants is its non-reliance on DNS services to function properly, all traffic and commands are addressed and called at an IP level. This further simplifies the initial testbed design, while leaving open the possibility for DNS implementation later.

IV. Enterprise Configurations

A. Topological Design

The enterprise deployment of the testbed includes a wireless distribution system (WDS) with four Linksys WRT54GL access points (APs). These systems are operating on DD-WRT 3.0 Build 44715 and they are interconnected to a remote authentication dial-in user service (RADIUS) authentication server on an additional interface to our main router [13]. The IP allocations operate under the 192.164.7.0/24 network, with the RADIUS host assigned 192.164.7.25, and four APs have assigned 192.164.7.50-53 respectfully, as shown in Fig. 10. This expanded topology allows increased flexibility with additional revisions, allowing the system to scale effectively.

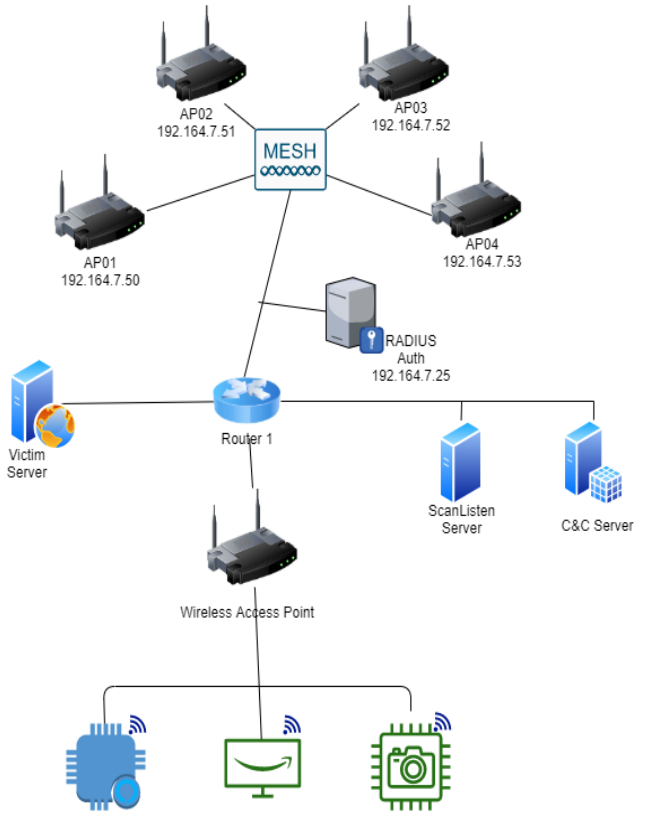

Fig. 10. *iBOT* including enterprise topology

B. Radius Configuration

The RADIUS server chosen for this project is *freeRADIUS*, an open-source implementation, operating on a Raspberry Pi running Raspbian Buster. The RADIUS server is configured as such to allow access to any device residing on 192.164.0.0/16. This ensures all the AP's configured clients can leverage RADIUS authentication.

The WRT54GL APs are pre-loaded with DD-WRT. In the configuration pane, the Global RADIUS Server Settings are configured with the master RADIUS IP Address, and the configured RADIUS Key, as shown in Fig. 11. This step is to be repeated across all APs in the deployment.

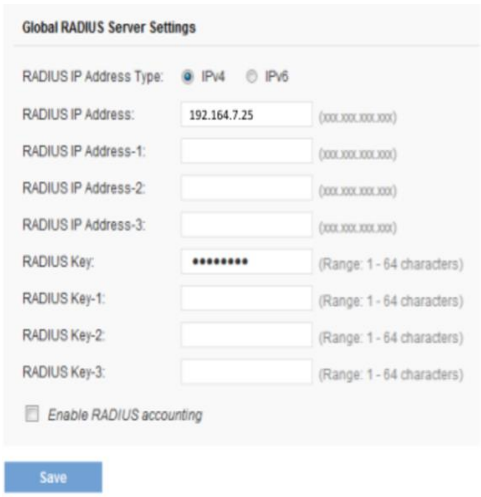

Fig. 11. *Radius Configuration Settings*

C. Enterprise Botnet Application

Despite the differences in topological design, botnet activity across endpoints remains similar to the home network implementation. However, when spreading our endpoints across multiple AP nodes, it can be visualized quickly how botnets can proliferate in enterprise networks. The utilization of services, such as RADIUS for Telnet authentication can provide an efficient, centralized way of stopping such attacks if properly deployed.

The enterprise segment is also an ideal ground to deploy additional services, such as a centralized instance of DHCP, or localized DNS. Both services are leveraged by many botnets today and can help to provide even more realistic depictions of botnet behavior in modern day enterprise networks.

V. Results

Upon installation of all components, testing was performed across multiple IoT devices, with all attempts targeting the deployed web instance. Commands are initiated from the C&C host and propagate to all infected IoT devices behind the wireless gateway. This process is nearly identical for each botnet, and are presented in a list format, alongside other tools for monitoring bots and different methods of attacks.

Our initial tests focused primarily the *Vivid* botnet deployment, due to it being the most like a traditional C&C architecture. Some botnets will install additional programs such as Low Orbit Ion Cannon (LOIC) to perform these attacks. Regardless of method, the target is specified within the command window, and some programs allow for individual bots to be selected for an attack.

The DDoS attack intends to stress the web hosts, and HTTP requests were generated on IoT devices during the attack window. Attacks can be cancelled manually or expire based on amount of time or number of transmissions. Due to the low stress tolerance of the web instance, results were immediately noticeable. Fig. 12 shows average web traffic captured from Wireshark, and Fig. 13 reveals the influx of requests originating

from the 192.168.5.100 address, or the gateway of the home IoT devices conducting the attack.

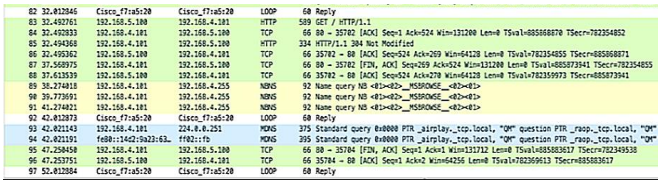

Fig. 12. Packet capture of normal web traffic

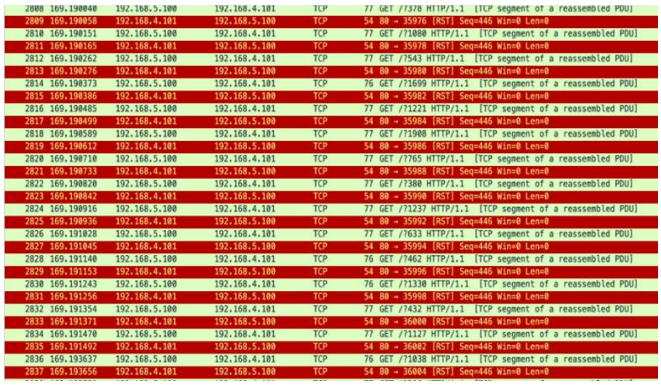

Fig. 13. Packet Capture During DoS Attack

The Web Server was unable to handle additional requests after roughly a minute of DoS requests on each test. No server firewalls were configured, but this could be implemented to further explore in the hardening of this design.

VI. Conclusion

Given the prevalence of both IoT devices and botnet threats in the modern cybersecurity space, we believe that simple and informative testing of these devices is of the highest priority. Our IoT botnet testbed, *iBoT* has served not only to increase general knowledge about botnet behavior and signatures within a network, but also grant us templates to build upon in the future of botnet research.

There are several improvements that could be pursued with the current iteration of the testbed, increased security appliances could be installed across several network segments, and packet captures could easily be implemented onto both the gateway and enterprise routers for additional data collection for a signature-based prevention strategy. The design itself could also be expanded on a large scale, with hundreds of virtualized devices across multiple network segments.

The enterprise segment could be further expanded upon, providing additional services such as DNS and DHCP, laying a better foundation for additional networked authentication services such as Kerberos. Different networking schemes besides wireless mesh could also be explored, providing further documentation into how botnets propagation differs over unique topological designs.

Despite possible architecture changes, the current iteration of *iBoT* provides ample insight into the functionality of botnets in both a home and enterprise setting. This functionality adds to the list of use cases of botnet education, and other testing scenarios. Given the presence of botnet-based security threats today, this deployment method of testing is a crucial piece in understanding the current, and future developments of botnet technology.

References

[1] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet ofThings: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys Tutorials17(4) (Fourthquarter 2015) 2347–2376

[2] Abdur Razzaq, Mirza & Habib, Sajid & Ali, Muhammad & Ullah, Saleem. (2017). Security Issues in the Internet of Things (IoT): A Comprehensive Study. International Journal of Advanced Computer Science and Applications. 8. 10.14569/IJACSA.2017.080650.

[3] Kumar, Ayush & Lim, Teng. (2019). A Secure Contained Testbed for Analyzing IoT Botnets.

[4] Shetu, Syeda & Saifuzzaman, Mohd & Nessa, Nazmun & Salehin, Md.musfaq-Us. (2019). A Survey of Botnet in Cyber Security. 174-177. 10.1109/ICCT46177.2019.8969048.

[5] A. Woodiss-Field and M. N. Johnstone, "Assessing the Suitability of Traditional Botnet Detection against Contemporary Threats," 2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT), Sydney, NSW, Australia, 2020, pp. 18-21, doi: 10.1109/ETSecIoT50046.2020.00008.

[6] S. Salamatian, W. Huleihel, A. Beirami, A. Cohen and M. Médard, "Why Botnets Work: Distributed Brute-Force Attacks Need No Synchronization," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 9, pp. 2288-2299, Sept. 2019, doi: 10.1109/TIFS.2019.2895955.

[7] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale Virtualization in the Emulab Network Testbed," in USENIX Annual Technical Conference, 2008, pp. 113-128.

[8] E. Alomari, S. Manickam, B. B. Gupta, P. Singh and M. Anbar, "Design, deployment and use of HTTP-based botnet (HBB) testbed," 16th International Conference on Advanced Communication Technology, Pyeongchang, 2014, pp. 1265-1269, doi: 10.1109/ICACT.2014.6779162.

[9] M. Li, Z. Sun and Z. Fang, "Hunting IoT Botnets with Wide-area-network Flow Data," 2019 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), Beijing, China, 2019, pp. 694-698, doi: 10.1109/SDPC.2019.00131.

[10] http://xrdp.org/

[11] https://www.cowrie.org/

[12] https://dd-wrt.com/support/router-database/?model=WRT54GL_1.0/1.1

[13] https://ufonet.03c8.net/