BladeRunner: Adventures in Tracking Botnets

Marc Eisenbarth and Jason Jones

Arbor Networks
{meisenbarth,jasonjones}@arbor.net,

Abstract. The problem of tracking botnets is not a new one, but still proves to be an important and fruitful research topic. We have been monitoring numerous botnets for years using an internally built tracking system, which has undergone a number of significant improvements and changes over the years. The basic tenet is a language for implementing botnet command-and-control mechanisms and tracking the resulting infiltrated botnets. This paper will cover the evolution of this system, which offers a vignette of the evolution of the modern day botnet itself. With this historical backdrop, we discuss our current monitoring mechanisms and selected botnet family case-studies, highlighting results we have obtained from our system and conclude with offering pieces of our toolkit to allows others to conduct similar investigations.

Keywords: botnet,ddos,malware

1 Introduction

The topic of tracking botnets is a well-researched area [?] [?] [?] [?], but it is also one that merits revisiting due to modern motivations, command-and-control (CnC) mechanisms and infection vectors [?]. Studying the behavior of botnet CnCs gives researchers insights unachievable simply by studying the malware executable. There are a number of metrics used when studying botnets, such as size, associated malware, geographical location and behavioral intent of the botnet. Arbor Network's Security Engineering and Response Team (**ASERT**) has developed an internal tracking system - codename "BladeRunner" - which solves the botnet tracking problem by taking a behavioral approach to identify and monitor botnets capable of distributed denial-of-service (DDoS). By tracking the behavior and capabilities of these botnets, researchers can glean critical information on botnet targets, classifying them by features such as geographical location, affected industry sector and geo-political motivations. In the latest evolution of BladeRunner, have encountered many challenges in modernizing the code to address modern threats, such as CnC protocols becoming more sophisticated, botmasters maintaining blacklists to prevent researchers from infiltrating the CnC and in some cases broad geographical discrimination.

2 BladeRunner Architecture

The original BladeRunner architecture was a set of pseudo-classes that used Python's `os.system` to call `curl` externally to implement HTTP-based protocols. In addition, the original design relied upon hard-coded lists of CnC servers to be present in related sample binaries. Finally, the code was limited in its ability to only process a fixed number of DDoS-related CnC commands. The upcoming sections will expand on how the old system was redesigned from the ground up to accommodate modern botnet CnC implementations and handle non-DDoS commands such as *update*, *download*, *steal passwords*, etc. that provide useful behavioral clues of the botmaster's intent. The architecture presented depends greatly on the quality of analysis done manually by researchers to get accurate data into the system and to build reliable and undetectable bots. The quality of the protocol reimplementation is extremely important. If a fake bot can be detected, then the resulting data returned from a CnC cannot be trusted and often results in the botmaster blacklisting the researcher's network space.

2.1 Data Models

The architecture of the new system fundamentally addresses these issues. First, data models were created to support storage of CnC information and allow for storing of related, semi-structured data. The main data model for CnC information stores a foreign key to the bot, the CnC host, port, CnC URI, timestamps for success, and how many consecutive failures that been seen for that specific CnC. There is also semi-structured data stored in another table that has a relation pointing to the relevant CnC. This record contains various items, such as encryption keys, specific HTTP headers and unique bot identifiers. This provides the necessary data for both relatively simple CnC protocols such as DirtJumper and also more complex custom protocols such as DarkComet. The storage of consecutive connection failures also allows for aging out CnCs that are suspected to be dead or inactive. The current aging mechanism waits for three days of inactivity and will then attempt a connection once per day for the next two weeks, before moving on to once a week for the next few months. The mechanics of this aging mechanism were based on past experiences of CnCs being dormant for periods of time or even feigning existence by returning a `404 Not Found` error to discourage probing researchers.

The next consideration in the improved data model was to store information about the CnC itself that was discovered during communication. For DNS names, the IP addresses that they currently point to are stored. Historical "passive" DNS is collected alongside any commands received from the CnCs themselves. Commands are stored in both raw and parsed format so the original encrypted or obfuscated command is available for future analysis. The normalized DDoS targets are stored in yet another table that has a pointer back to the command that ordered attacks on those targets. This structure allows researchers to perform additional manual analysis on unrecognized commands, which hopefully results in more accurate processing at a later date, and allows presentation of these new results based on the original, raw commands. An example where this flexibility has turned out to be extremely useful is when ASERT created new processing workflows around *download* and *update* commands. As we show later in this paper, this allowed for easier tracking of the malware that each CnC and botnet are dropping onto compromised systems and also allows for better tracking of the evolution of the bot code itself, by monitoring and acting upon these *download* and *update* commands.

The changes to the data models allow for the flexibility in designing bots, while working well with the botnets that were tracked in the legacy system. A graphical representation of the models described above is shown in Figure 2.1.

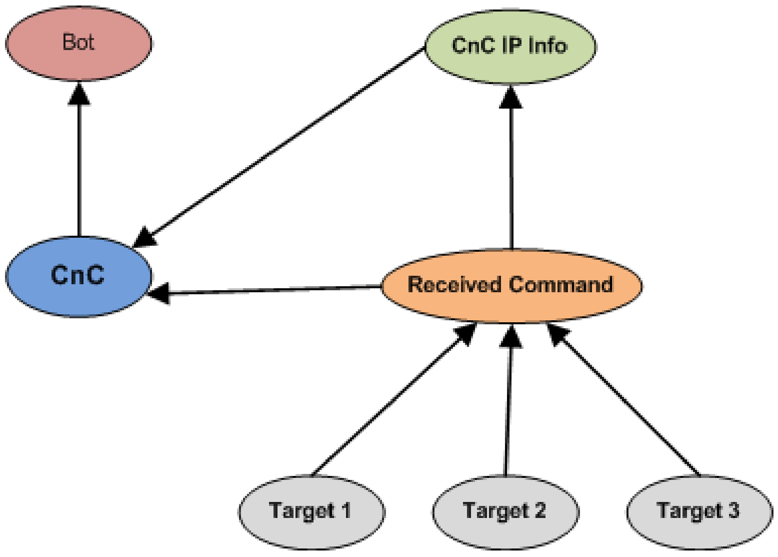


Fig. 1. Diagram of database layout

2.2 Hunting CnC Servers

What good is a "fake bot" if it does not have an active CnC to communicate with? Within malware families, significant amounts of code are shared between different samples. For instance, two separate DirtJumper samples may only vary in the CnC hostname embedded in the executables by the malware. Modern malware often accomplishes this through the use of a *builder* application, which automates creating many unique samples that have embedded one-time use characteristics. Thus, the same communication protocol is used, even though the CnC differs. Thus it is important that BladeRunner can monitor CnCs for which it possess defined communication protocol specifications, but may not necessarily have a sample of the specific malware.

As mentioned earlier, a limitation of the first incarnation of Bladerunner was the use of a hard-coded list of CnC servers, each of which was specified on a per bot basis. Later bots that were written started dynamically querying ASERT's malware analysis system - codename "MCorral", short for Malware Corral - and storing results in a Python Pickle Cache. There was no programmatic way to inspect data stored in this cache and it was not possible to tell which of the entries stored in this cache were active or why they became inactive. The goal of the new harvesting system was to allow for multiple diverse inputs to feed the system. These harvesters have a loose structure with the most important pieces for identifying network traffic being stored as patterns in a database.

Each botnet we track now has a processing module to pull the necessary information from the connection. The processing module is responsible for extracting the CnC information in addition to any bot specific parameters that are needed for communicating with the CnC. Examples of this will be presented later in the Implementation Case Studies section.

VirusTotal BladeRunner uses the "Malware Hunting" feature of a VirusTotal[**?**] Intelligence account to find new samples to analyze and ultimately new botnets to track. This is accomplished by uploading Yara[**?**] rules to VirusTotal and monitoring alerts for new samples. The hits are polled using the JSON API and if a new sample is found, the VirusTotal behavioral data and associated network PCAP data are compared against the set of identification patterns mentioned above. If any pattern matches, the connection payloads are passed off to the processor engine to parse and store relevant data. If there was no behavioral data or PCAP available, the sample file is downloaded and submitted into MCorral for local static and dynamic analysis.

ASERT MCorral ASERT's MCorral is heavily relied upon for feeding information to BladeRunner. Malware that is passed through the system is run with multiple configurations and is then passed along for post-processing where attempts are made to classify the sample based on its behavior. A visual representation of this process is shown in Figure 2.2. The analysis system allows for tagging specific connections made by a sample if they match patterns or tagging the full sample if Yara rules, mutex patterns, etc. are matched. Using the tags placed on a sample or a connection, the requisite data for communicating with the CnC for most bots can be harvested from the network traffic with little effort.

Configuration Rippers Unfortunately, not all malware can use the tagged sample and/or network traffic to obtain the information needed to communicate with a CnC server. Malware such as DarkComet, MP-DDoS (also known as IP-Killer) and YoYo DDoS use encrypted and/or obfuscated protocols. In these instances, the configuration must be extracted from the binary - typically from a memory dump - and then using the capability of storing the configuration and

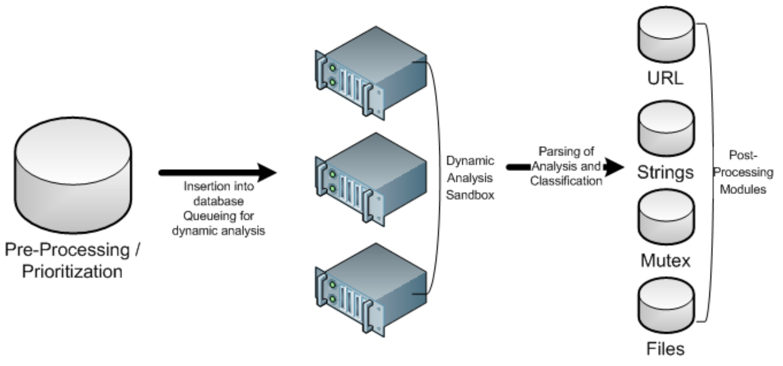

Pre-Processing / Prioritization
Insertion into database Queueing for dynamic analysis
Dynamic Analysis Sandbox
Parsing of Analysis and Classification
URL
Strings
Mutex
Files
Post-Processing Modules

Fig. 2. Life of a Sample in MCorral

keys as unstructured data tied to a specific CnC. The configuration rippers tend to be difficult and very involved to write, and are therefore only used as a last resort for botnets that ASERT has a strong desire to monitor. More information on specific configuration rippers implemented in BladeRunner will be presented later in the Implementation Case Studies section.

3 Implementation Case Studies

In this section, a number of case studies will be presented based on bots that exist in BladeRunner. The case studies will discuss how the protocols were re-implemented and the specific challenges overcome.

3.1 DirtJumper

The DirtJumper[**?**][**?**][**?**] (formerly Russkill) family of DDoS malware is long-lived and still one of the more popular families of DDoS malware that are processed by MCorral. DirtJumper has spawned many variants and has a simple clear-text HTTP-based protocol. DirtJumper uses an HTTP POST with a single parameter named *k* that is either 15- or 32-bytes in length depending on the version and is used by the CnC as the unique bot identifier. The server will respond back with a numeric code for the DDoS attack ordered, information on number of threads, attack timeout, and a set of targets separated by carriage-return-line-feed. An example of this process can be seen in Figure 3.1. Reimplementing this protocol can be done easily and there is minimal chance of detection due to the CnC panel not being very strict on bot uses to phone-home as long as it sends an appropriate value for *k*. This is evidenced by the original BladeRunner system generating a new value for *k* on every phone-home, and likely filling the list of bots in the panel up with a number of old bots on the same IP.

```
POST /dj/ HTTP/1.1
Host: aba.net.ua
Connection: keep-alive
Keep-Alive: 300
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US)
Content-Length: 17
Content-Type: application/x-www-form-urlencoded

k=UfNxfuT1LZCZE8o
```

```
HTTP/1.1 200 OK
Date: Mon, 28 Oct 2013 22:38:56 GMT
Server: Apache/2.2.14 (Ubuntu)
X-Powered-By: PHP/5.3.2-1ubuntu4.21
Vary: Accept-Encoding
Content-Length: 19
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html04|25|60cwdw.24.net
```

Fig. 3. DirtJumper Bot and CnC Communication Example

3.2 Drive2

Drive2[?] is the 2nd version of Drive[?] which appears to be a successor to the just discussed DirtJumper malware family. The first version of Drive had a communication protocol to DirtJumper, but modified the way the DDoS commands looked when sent back from the server. The command is first XOR'd against a static key and then Base64-encoded before being sent to the bot, a pseudo-code representation of this can be seen in Algorithm 1. The CnC URL is also encoded inside the malware sample using the same procedure. Without knowledge of the key, intercepting CnC communications does not reveal much about the botnet. Thus far, all binaries use the same key, but the key can also be brute-forced by programmatically locating the encoded URL inside the malware sample and comparing it to what the sample attempted to communicate with. The key is also located at predictable offsets, so brute-forcing has never been required.

Now that the algorithm for encryption is known, it is a simple task to reverse the process and then parse out the targets of the specified attacks.

Algorithm 1 Drive2 Command Encryption

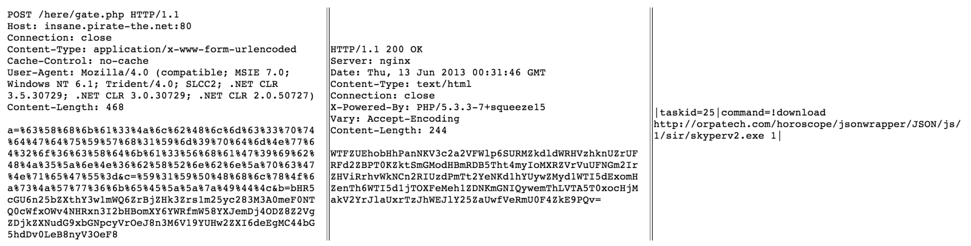
```
function EncryptCommand(commands, key)
    $msg \leftarrow$ ""
    $i \leftarrow 0$
    while $char \in commands$ do
        $msg \leftarrow msg + char \oplus C(key)$
    $msg \leftarrow Base64(msg)$ return $msg$
```

3.3 Athena HTTP

Athena HTTP is a successor to the Athena IRC bot that has been around for a number of years. The migration to HTTP rather than IRC makes it much easier to monitor without detection, since regular connects and disconnects would be readily visible to a botmaster in an IRC channel, but not necessarily as visible through a Web control panel. Athena HTTP sports a number of DDoS features, and also download and execute file, execute shell commands, and directed browsing to sites. Communicating with an Athena HTTP CnC proves to require more finesse. First, the command is generated and put into a format string to send to the client. Next, the command is run through the Base64 algorithm. After that, two 32-byte length random strings are generated and applied to the Base64'd string in a string translation manner. Finally a random 8-byte length string is generated to be used as a data marker by the server when it responds. These values are stored in three POST parameters, the first and last of which are fully URL-encoded even though they contain no characters that require it. The server response prepends the data marker to a Base64-encoded list of commands that have been run through the string translation table used by the client. An example phone-home and response that was intercepted and decoded is shown in Figure 3.3.

Another challenge that is not quite as prevalent in other HTTP-based botnets is that the requested phone-home interval is dynamic, and if a bot does not respond fast enough during that interval, the CnC may blacklist it. To account for this, the phone-home interval requested by the server is extracted and the bot will phone-home using that interval.

The actual command inside the response sent back by Athena HTTP CnCs mirror the command structure of the IRC version. There is an exclamation point followed by a command and then any parameters - in the case of DDoS attacks, a target or URLs in the case of download commands. Additionally, the CnC expects to be told that the bot is still busy while a DDoS attack is being conducted, so a timer is also implemented to determine whether or not a busy status should be reported back.



```
POST /here/gate.php HTTP/1.1
Host: insane.pirate-the.net:80
Connection: close
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0;
Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR
3.5.30729; .NET CLR 3.0.30729; .NET CLR 2.0.50727)
Content-Length: 468

a=%63%58%68%6b%61%33%4a%6c%62%48%6c%6d%63%33%70%74
%64%47%64%75%59%57%68%31%59%6d%39%70%64%6d%4e%77%6
4%32%6f%36%63%58%64%6b%61%33%56%68%61%47%39%69%62%
48%4a%35%5a%6e%4e%36%62%58%52%6e%62%6e%5a%70%63%47
%4e%71%65%47%55%3d&c=%59%31%59%50%48%68%6c%78%4f%6
a%73%4a%57%77%36%6b%65%45%5a%5a%7a%49%44%4c&b=bHR5
cGU6n25bZXthY3w1mWQ6ZrBjZHk3Zrs1m25yc283M3A0meF0NT
Q0cWfxOWv4NHRxn3I2bHBomXY6YWRfmW58YXJemDj4ODZ8Z2Vg
ZDjkZXNudG9xbGNpcyVrOeJ8n3M6V19YUHw2ZXI6deEgMC44bG
5hdDv0LeB8nyV3OeF8
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 13 Jun 2013 00:31:46 GMT
Content-Type: text/html
Connection: close
X-Powered-By: PHP/5.3.3-7+squeeze15
Vary: Accept-Encoding
Content-Length: 244

WTFZUEhobHhPanNKV3c2a2VFWlp6SURMZkdldWRHVzhknUZrUF
RFd2ZBPT0KZktSmGModHBmRDB5Tht4myIoMXRZVrVuUFNGm2Ir
ZHViRrhvWkNCn2RIUzdPmTt2YeNKd1hYUywZMyd1WTI5dExomH
ZenTh6WTI5d1jTOXFeMeh1ZDNKmGNIQywemThLVTA5T0xocHjM
akV2YrJlaUxrTzJhWEJlY25ZaUwfVeRmU0F4ZkE9PQv=
```

```
|taskid=25|command=!download
http://orpatech.com/horoscope/jsonwrapper/JSON/js/
1/sir/skyperv2.exe 1|
```

Fig. 4. Athena HTTP Phone-Home, Response & Decoded Command

3.4 Madness

The Madness DDoS bot [**?**] presented its own set of challenges when reimplementing its CnC protocol. The phone-home is a simple multi-parameter GET request, but at first glance there are two parameters that do not appear to have any relevance. The server response for Madness is a Base64-encoded set of commands with the attack type as the first value followed by a semi-colon separated set of targets to perform that attack on. Further investigation yielded that these two parameters were counters to track the number of times the bot had phoned-home and the number of attack packets that had been sent. The number of phone-homes is an easy value to continually increment using the unstructured data storage functionality built into BladeRunner, but the number of attack packets sent becomes more problematic as no attack packets are actually sent and the expected attack rate is not known. To determine "reasonable" values for attack packets, the sample was launched in a controlled, host-only VM environment and ordered to conduct fake attacks against the private VM network. Taking that as a baseline and appending a random value to avoid easily identifiable increments, a value was able to be calculated that has helped BladeRunner avoid CnC blacklisting.

3.5 DarkComet

Of the case studies presented in this paper, DarkComet presented the biggest challenge. The problem stems from a wide array of commands used to implement its Remote Access Tool (RAT) capabilities as well as the use of a custom binary protocol with layered encryption. DarkComet has been analyzed many times [?] [?] [?] and these analyses provide a solid foundation for reimplementing its protocol. The first step is to extract the configuration information. To this end, a configuration ripper was created to extract the CnC server, port, encryption key and other information from the binary. Pseudo-code for how this algorithm works is presented in Algorithm 2.

Once the configuration is extracted, the encryption algorithm used to communicate with the CnC must be re-implemented, along with support for all observed CnC commands. Some commands prove more problematic to implement than others which greatly increases the probability that our fake bot will be easily detected. Commands that request control of the desktop, live chat, live streaming from webcams, etc. prove especially problematic. In these cases, terminating the connection and marking the server inactive is one solution. This is not ideal because valuable intelligence may be lost, but it is also not ideal for actors, big or small, to detect that they are being monitored.

Algorithm 2 DarkComet Configuration Extraction

```
function ExtractConfiguration($commands$,$key$)
    $config \leftarrow$ ""
    $resource \leftarrow FindAndLoadResource()$
    $prefix \leftarrow LocateStandardCryptoPrefix()$
    $rsrc\_key \leftarrow LocateCryptoKey()$
    $rsrc\_key\_suffix \leftarrow ComputeKeySuffix()$
    $key \leftarrow rsrc\_key + rsrc\_key\_suffix$
    $config \leftarrow DecryptResource(resource, key)$ return $config$
```

4 Results and Findings

Since March 2013, 1,339 CnC servers have been added to the BladeRunner system across 14 malware families. Of those, 395 have been successfully contacted. Table 4 shows how the distribution breaks down between malware families. During this time, 234,540 attacks have been logged against 2,381 distinct targets with 136 distinct attack types.
BladeRunner has seen 500 download and execute commands and has successfully executed over 150 of the commands to result in a large number of previously unseen malware. Six distinct update commands have been seen resulting in new versions of the malware being monitored.

Family	*Total*	*Active*	*Distinct Targets*
Athena HTTP	55	38	333
DirtJumper	278	98	1097
DarkComet	620	138	
DarkDDoser	145	36	88
Drive	25	17	610
Drive2	22	15	188
Madness	5	4	3
MP-DDOS	74	10	253
Pandora	80	47	79

Table 1. Number of Total and Active CnCs per Family

4.1 Relationships between CnCs

As passive DNS data is gathered over time, interesting relationships between IP addresses and hostnames start to emerge amongst families. Some malware families appear to be more likely to be co-located with other families than others. Multiple instances of DarkComet and DarkDDoser hostnames resolving to the same IP addresses have been witnessed, but no other shared IPs or hostnames with other families. Additionally, these two malware families typically use dynamic DNS services and account for the majority of the IP address flux witnessed in BladeRunner - one hostname even resolved to 267 distinct IP addresses over the time it was being monitored.

A number of Drive and Drive2 CnCs appear to have close relationships as well. Nearly all of the Drive2 CnCs discovered have at some point in the past functioned as a Drive CnC. BladeRunner has also seen Athena HTTP CnCs ordering downloads of both of these families and the downloaded sample used the same hostname as the Athena HTTP CnC. Both Drive and Drive2 CnCs have also been seen targeting large numbers of the same sites which suggests either a relationship between botnet operators or a relationship between the hiring entity and the botnet operators.

One of the most popular sites for continued DDoS attacks that span multiple families is a website that is attempting to expose corruption in the sports world. This site has seen extended DDoS attacks from Athena HTTP, Drive, Drive2, DirtJumper v5, and Pandora CnCs. A number of news organizations have also been targeted in the same manner.

4.2 Political Motivations

Political motivations for DDoS attacks have been discussed at length and it should be no surprise that BladeRunner has witnessed these attacks. It is interesting to note that with many of these attacks the increasingly small amount of time from a news story breaking to an attack being ordered in support or against the parties involved. One such instance that BladeRunner witnessed revolved around the political unrest in Egypt that occurred in the Summer of 2013. The change in leadership saw a DarkDDoser CnC spawn and start ordering attacks against the new government website. A few days later, the United Arab Emirates government pledged aid to the new government and the same DarkDDoser CnC quickly started switching its attack target from the Egyptian government to the United Arab Emirates government. Table 4.2 shows a timeline of the attacks witnessed with the earliest seen headlines of events that are believed to be the cause of the attack.

Both Drive and Drive2 CnCs have also ordered a number of what appear to be politically motivated attacks. These have centered more on the ex-Soviet bloc region and mostly appear targeted towards websites that are not supportive of the current governments. These attacks have centered around news sites in Dagestan, Azerbaijan, Ukraine, and Russia.

Date	*Event*
July 3, 2013	Mohamed Morsi Unseated as President of Egypt
July 5, 2013	DDoS Attacks Launched Against Egyptian Ministry of Interior
July 9, 2013	UAE Pledges Aid to New Egyptian Government
July 9, 2013	DDoS Attacks Launched Against UAE Ministry of Foreign Affairs
July 9, 2013	Hazem al-Beblawi Sworn in as Prime Minister
July 16, 2013	DDoS Attacks Launched Against Egyptian Cabinet

Table 2. Timeline of Events in Egypt and Attacks by DarkDDoser

4.3 Taking out the Competition

A number of attacks on both “underground” forums and so-called “booter” or “stresser” services [?] [?] have also been witnessed. In some cases, this has led to the discovery of underground forums previously unknown to ASERT. Many of these attacks do not appear to succeed, but there have been attacks that have resulted in either the closure or relocation of the forum site. The DirtJumper, Drive, and Drive2 families appear to be the most involved in the forum attacks, while Athena HTTP has been observed as the most active in attacking "booter" and "stresser" services. Many of the Athena HTTP CnC hostnames appear to have affiliation with booter services, so many of these attacks appear to be cases of competitive takeout. Table 4.3 shows a list of hosts that have attacked in the first table and a list of attacked hosts in the second table.

Booter CnC
truboot.org
smokelessbooter.tk
infinitybooter.com
revoltsresser.com
www.mydowncenter.me
downstealer.com
ddos.huarengang.com

Targeted Competitor
www.planetboot.com
www.imistress.ru
fuckav.ru
toxicboot.pw
www.ipstresser.com
takedown.pw
reboot.pw
sovietstresser.com
www.downcenter.me
national-stresser.com
stress-me.net
speedbooter.fr
ifud.ws

Table 3. Likely Booter Services & Targeted Competitor Sites

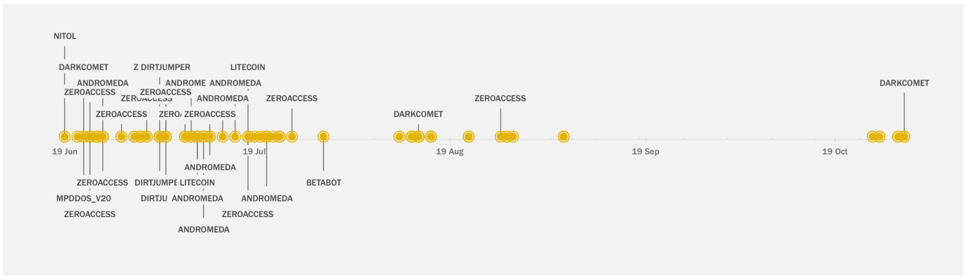


Fig. 5. Timeline of Executables Dropped by Athena HTTP

4.4 Pay-Per-Install

Instead of functioning as a DDoS botnet, Athena HTTP appears to be functioning more in the role of a pay-per-install (PPI) botnet. This is not surprising since it is only able to attack one target at a time, whereas its competitors are able to order attacks against multiple sites and of varying types. The executables dropped on systems have ranged from ransomware to bitcoin miners to more serious malware like Andromeda. A timeline of identified malware is shown in Figure 4.3. One interesting thing to note is that many of these executables are new to services like VirusTotal and in many cases were new to MCorral, so they provide an interesting set of data for ASERT to analyze, which is not available to the security research community at large. Early on, there were a large number of samples downloaded, but that number dropped off in late August 2013. This ended up being due to some small changes in the malware that made our hunting mechanisms miss the samples. Dots without a label were samples that were at the time unrecognized by MCorral's tagging system, and those with lines were ones identified with the names used for those families.

5 Conclusions and Future Work

In this paper, we have presented the extensible and flexible architecture of the BladeRunner botnet tracking framework and we have shown the ability for this system to obtain a extremely valuable and diverse dataset, the likes which have not been previously available. Using BladeRunner, ASERT has gained access to previously unseen malware samples and in some cases new versions of these those samples before others in the security research community. Correlations have also been drawn between attacks ordered and world events to help give a better understanding of geopolitical issues in DDoS attacks.

Future work will focus on adding more supported malware families to the framework with a focus on non-DDoS families and redesigning previous work ASERT has done on estimating botnet size.

Publicly shared code will be available at https://github.com/arbor/.

References

1. Virustotal. http://www.virustotal.com/.
2. Yara. https://code.google.com/p/yara-project/.
3. Prolexic threat advisory: Dirt jumper v3. http://www.prolexic.com/pdf/ProlexicThreatAdvisoryDirtJumper.pdf, 2012.
4. Jeff Edwards. It's not the end of the world: Darkcomet misses by a mile. http://ddos.arbornetworks.com/uploads/2012/03/Crypto-DarkComet-Report.pdf, 2012.
5. Jeff Edwards. Reversing the wrath of khan. http://www.arbornetworks.com/asert/2012/03/kahn/, 2012.
6. Jeff Edwards and Jose Nazario. A survey of chinese ddos malware. http://www.virusbtn.com/pdf/conference_slides/2011/Edwards-Nazario-VB2011.pdf, 2011.

7. Felix C. Freiling, Thorsten Holz, and Georg Wicherski. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In *In Proceedings of 10 th European Symposium on Research in Computer Security, ESORICS*, pages 319–335, 2005.
8. Shanw Denbow & Jesse Hertz. Pest control: Taming the rats. http://www.matasano.com/research/PEST-CONTROL.pdf, 2012.
9. Georg Wicherski & Thorsten Holz. Catching malware to detect, track and mitigate botnets. http://www.blackhat.com/presentations/bh-jp-06/BH-JP-06-Wicherski-Holz.pdf, 2006.
10. Jason Jones. Dirtjumper drive shifts into a new gear. http://www.arbornetworks.com/asert/2013/08/dirtjumper-drive-shifts-into-a-new-gear/, 2013.
11. Jason Jones. Dirtjumpers ddos engine gets a tune-up with new drive variant. http://www.arbornetworks.com/asert/2013/06/dirtjumpers-ddos-engine-gets-a-tune-up-with-new-drive-variant/, 2013.
12. Jeremy Linden Jose Nazario. Botnet tracking techniques and tools. http://techedu.cu.cc/hacking/Botnet%20Tracking%20-%20Tools,%20Techniques,%20and%20Lessons%20Learned-paper.pdf, 2006.
13. Kafeine. Meet madness pro or few days rise of a ddos botnet. http://malware.dontneedcoffee.com/2013/10/meet-madness-pro-or-few-days-rise-of.html, 2013.
14. Brian Krebs. Ddos services advertise openly, take paypal. http://krebsonsecurity.com/2013/05/ddos-services-advertise-openly-take-paypal/, 2013.
15. Brian Krebs and Lance James. Spy-jacking the booters. https://www.blackhat.com/us-13/archives.html#Krebs, 2013.
16. Adam Kujawa. You dirty rat! part 1 - darkcomet. http://blog.malwarebytes.org/intelligence/2012/06/you-dirty-rat-part-1-darkcomet/, 2012.
17. Jose Nazario. Ddos attacks in russia. http://ddos.arbornetworks.com/2012/02/ddos-attacks-in-russia/, 2012.
18. Andre' M. DiMino & Mila Parkour. Dirt jumper ddos bot - new versions, new targets. http://www.deependresearch.org/2011/10/dirt-jumper-ddos-bot-new-versions-new.html, 2011.