Arbor White Paper

Anatomy of a Botnet

How the Arbor Security Engineering & Response Team (ASERT) Discovers, Analyzes and Mitigates DDoS Attacks

ARBOR® NETWORKS

About Arbor Networks

Arbor Networks, Inc. is a leading provider of network security and management solutions for next-generation data centers and carrier networks. Arbor's proven solutions help grow and protect our customers' networks, businesses and brands. Arbor's unparalleled, privileged relationships with worldwide service providers and global network operators provide unequalled insight into and perspective on Internet security and traffic trends via ATLAS—a unique collaborative effort with 100+ network operators across the globe sharing real-time security, traffic and routing information that informs numerous business decisions. For technical insight into the latest security threats and Internet traffic trends, please visit our Web site at arbornetworks.com and our blog at asert.arbornetworks.com.

Table of Contents

The Latest Botnet and DDoS Attack Trends

One of the Internet's great strengths is also one of its weaknesses: with no central governing body, the Internet is a wide-open environment that has allowed a large criminal element to thrive. For evidence, look no further than the escalating growth of botnets and their role in perpetrating a wide array of crimes including identity theft, banking fraud, spam campaigns, malware distribution and Distributed Denial of Service (DDoS) attacks.

Innovation Drives Growth

Fueled by innovations like do-it-yourself botnet construction kits and rent-a-botnet business models, the growth of botnets has skyrocketed and botnet products and services are now brazenly advertised and sold on the Internet. As many as one quarter of all personal computers may now be participating in a botnet, unknown to their owners. At its peak, it is believed the now-defunct Mariposa botnet may have controlled up to 12 million zombies.

One of the major uses of botnets is to launch Distributed Denial of Service (DDoS) attacks, which are simultaneously executed from multiple infected hosts under the command and control of a botherder. The goal is to slow or take down the targeted domain, network infrastructure, web site or application so it can't respond to legitimate requests. An attack may also have a secondary goal of installing malware that can steal valuable information from the infected system or enable further intrusions at a later date.

Crime and Politics Are Favorite Pastimes

The largest share of DDoS attacks are criminally motivated: attackers are seeking financial gain via stolen data or by extorting "protection money" in exchange for a promise not to disrupt a retail, ecommerce or gaming site that generates substantial revenue. In recent years, there has also been a rise in politically motivated DDoS attacks around the globe including in Iran, South Korea, Estonia, Malaysia, China and the U.S.

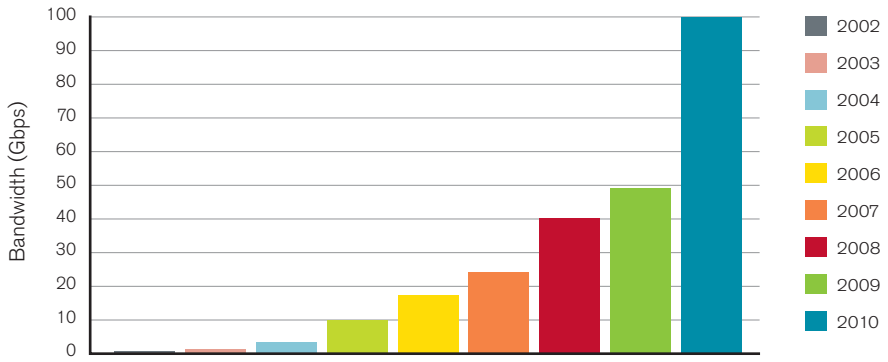
The nature of political attacks are exemplified by the WikiLeaks controversy, where some attacks were launched against WikiLeaks—presumably by parties angered at the disclosure of classified U.S. diplomatic cables—and other attackers, sympathetic to WikiLeaks, targeted hosting providers who had stopped doing business with the site.

Volumetric and Application-Layer DDoS Attacks Continue to Grow

Until recently, most DDoS attacks were volumetric attacks, which seek to overwhelm the network infrastructure with bandwidth-consuming flooding assaults or by targeting servers, load balancers and firewalls with state-exhaustion attacks. In 2010, volumetric attacks grew in size, frequency and complexity; the largest attack reported to Arbor was 100 Gbps, over a 100 percent increase from the previous year.[1] (See the graph, page 3). An attack of this magnitude can easily overwhelm a majority of today's Internet access services.

Application-layer DDoS attacks, which exploit the characteristics of widely used applications—primarily HTTP, DNS, VoIP, and SMTP—also grew. While these attacks consume less bandwidth than volumetric attacks, they are more difficult to detect. In 2010, 77 percent of respondents in Arbor's annual survey experienced application-layer attack, and such attacks represented 27 percent of all attack vectors.

DDoS Attack Size Over Time



Source: Arbor Networks

[1] As reported in Arbor Networks' 2010 annual *Worldwide Infrastructure Security Report.*

ASERT Plays a Leadership Role

As network and hosting operators, enterprises and government institutions grapple with the fast-evolving challenges of DDoS attacks, they often turn to Arbor Networks for insight, guidance and solutions for DDoS attack detection and mitigation.

Arbor is recognized as a worldwide leader in Internet traffic and threat analysis. Members of the Arbor Security Engineering & Response Team (ASERT) blog frequently on emerging threats and are often sought out by the media for their insights on contemporary events, such as the WikiLeaks controversy and governments' suspected role in thwarting network communication during uprisings in Egypt, Libya and elsewhere in the Middle East.

Arbor's collective expertise is captured and widely shared through two large-scale, collaborative initiatives:

- **Annual *Worldwide Infrastructure Security Report:*** Arbor annually surveys 100-plus customers, including ISPs, data center and hosting operators, and large IT organizations—who, collectively, account for as much as 50 percent of all Internet traffic worldwide. Detailed data is gathered on the type, size and frequency of threats an organization has experienced and the tools and strategies it uses to mitigate attacks. The report provides a global perspective on the threat environment and helps providers understand their own experiences in the context of wider trends while learning from their peers' challenges and successes.
- **ATLAS® (Active Threat Level Analysis System):** Peakflow® SP solutions—which are deployed in a majority of the world's ISP networks—include an optional "anonymous statistics" feature. When providers enable this feature, alert and DDoS attack statistics gathered for their benefit are also anonymized and sent to Arbor databases that are monitored by ASERT. In addition, a number of ISPs have authorized the deployment of specially designed probes that gather DDoS attack information on dark-IP networks.

Together, these initiatives provide a rich, unmatched source of intelligence that ASERT team members can draw on in analyzing current and emerging threats.

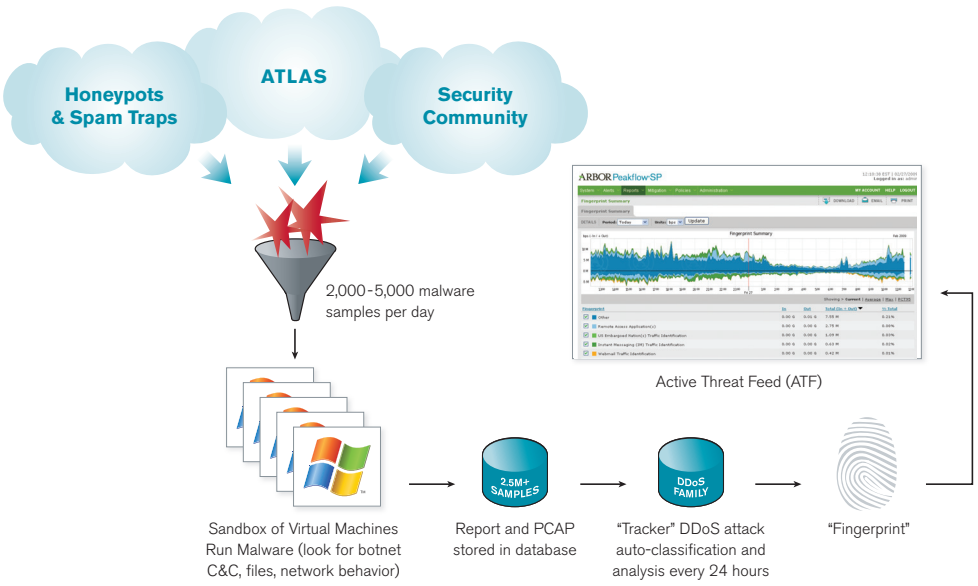
ASERT's Botnet and DDoS Attack Analysis Process

Take "a look under the hood" at the structured, multi-step process ASERT has developed to analyze botnet and DDoS attacks. ASERT collects malware samples daily and executes them in a "sandbox" environment, categorizing each botnet and attack, determining its characteristics, and publishing the information to a database of more than 2.5 million unique threat analyses.

When a new piece of malware is detected or a previously known form mutates, ASERT develops a unique "fingerprint"—a network behavioral pattern that can be used to detect the specific attack—and distributes it to customers of the Peakflow platform.

Arbor Networks is one of several companies and research groups that actively analyze Internet threats. Industry collaboration among these security organizations happens on a daily basis. Such collaboration is paramount to the industry's ability to successfully thwart the daily onslaught of Internet-based threats such as worms, viruses, spam, phishing/pharming, botnets and DDoS attacks.

Because no single entity can analyze and gain expertise in all types of malware or threats, each of these security organizations typically specializes in a specific type of threat. For example, ASERT specializes in the analysis of botnets and DDoS attacks and relies on other expert organizations in the security community for information regarding other types of threats. In return, Arbor openly publishes and shares its analysis and information with these other trusted security organizations.



ASERT's Automated Process for Botnet and DDoS Attack Analysis

The next section of this document outlines the discovery and in-depth analysis process that ASERT performs on a daily basis to protect organizations against botnet and DDoS attacks. It also describes the various forms of output used to alert security vendors, researchers and Peakflow customers of such threats.

1. On a daily basis, ASERT automatically collects 2,000-5,000 malware samples. These samples come from approximately two dozen different sources, such as:

- Anti-virus and other commercial security vendors.
- Public and private research organizations.
- Honeypots and spam traps.
- ATLAS network via specially designed Arbor Network probes deployed in the dark-IP portions of worldwide ISP networks.
- Arbor customers.

2. ASERT executes the malware samples on a number of virtual Microsoft Windows machines, referred to as the "sandbox."

3. After the malware samples are run, they are automatically categorized into different threat types, such as:

- Spam.
- Virus, worm.
- Botnet.
- DDoS bot.

4. As previously mentioned, ASERT specializes in and focuses on botnets and DDoS attacks. At this step of the analysis process, ASERT determines the specific characteristics of the botnet and DDoS attacks, such as:

- Address of the command and control (C&C) site.
- Protocol used to communicate with C&C.
- Attack vector.
- Code signatures.

5. The information gathered from running the malware in the sandbox is detailed in a report and packet capture (PCAP) files. These are then stored in a database that has over 2.5 million unique threat analyses.

6. Every 24 hours, an automated process known as "Tracker" runs against the unique threat analysis database to classify the malware into specific botnet or DDoS attack types, such as:

- **Downloader:** a program that downloads and executes other malicious software.
- **IRC bots:** programs connected to an IRC server to participate in a botnet.
- **BlackEnergy:** a specific botnet kit that Arbor has previously characterized.
- The most common attack types include:
  - HTTP request floods (the most commonly implemented and used type of attack).
  - User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) floods.
  - SMTP bots.

On average, in an analysis of 200 new malware samples, 180 are classified into known malware families, and 20 are unclassified. In other words, most attacks stem from common code bases that are shared with, sold to or stolen by the hacker/attacker community.

When a new piece of malware is released in the wild or an old one has changed, ASERT moves quickly to reclassify it and create a new "fingerprint" or profile of the threat.

7. Using a number of software engineering tools, ASERT then analyzes the unclassified samples looking for unique characteristics of the malware. For example, when analyzing an HTTP attack, ASERT looks for unusual HTTP headerfields, unique user-agents, abnormal ordering of the HTTP header fields and other subtle but distinguishing features.

This analysis can lead to the creation of a new classification. Such was the case with the YoYoDDoS Botnet, which will be detailed later in this document.

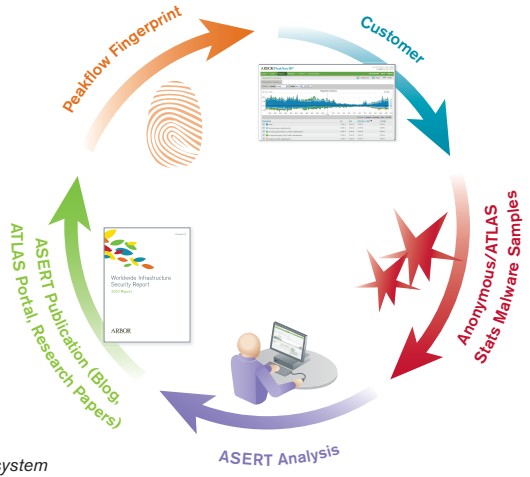
8. After ASERT has classified the malware samples, a risk level of low, medium or high is assigned.

9. In the case of medium- or high-risk levels, ASERT creates a "fingerprint" of the threat for identification in Peakflow solutions. A Peakflow fingerprint can be compared to an anti-virus program's "signature" that looks for a binary bit pattern within files to detect a virus, worm, etc. Instead, a Peakflow fingerprint looks at the network behavioral pattern of the malware. For example, a threat may exhibit the following sequential network behavior:

- Communicate to a known botnet C&C site using a specific protocol.
- Start a file transfer.
- Start a port scanning activity.

This network behavior can be indicative of a known botnet or network worm, virus, etc. In fact, there are times when a Peakflow fingerprint will detect a new variant of a virus or worm before an anti-virus program. For example, a new variant of a virus or worm can be created by simply modifying the virus payload. To detect this new variant, an anti-virus program must rely on a new virus signature. But in many cases, the new virus variant exhibits the same network behavior, which will still be detected by the Peakflow fingerprint.

10. When a Peakflow fingerprint is created, it is distributed to Peakflow customers using Arbor's Active Threat Feed (ATF) system. This completes what Arbor refers to as the "Arbor Threat Analysis Ecosystem," as depicted in the figure below.



Arbor Threat Analysis Ecosystem

The next section of this document overviews the in-depth analysis of an ASERT-discovered set of botnets powered by the YoYoDDoS malware family.

Anatomy of a Botnet: YoYoDDoS

Starting in May 2010, Arbor researchers discovered and processed over 300 specimens of a new family of trojans, dubbed YoYoDDoS. Designed to launch DDoS attacks, the trojans install a backdoor that causes infected systems to join to the YoYoDDoS botnet and establish a connection to a command and control server.

A sizable excerpt from the original analysis of YoYoDDoS describes in great detail its characteristics, behaviors, installation process, types of attacks launched and most frequent targets. The following section contains excerpts of ASERT's YoYoDDoS Botnet analysis published on the ASERT blog. For a more detailed analysis of this botnet, refer to the ASERT blog posting.[2]

A new family of DDoS bots started showing up in ASERT sandboxes in May 2010. The first sample was analyzed on May 7. Since then, ASERT sandboxes have processed over 300 specimens from this family. Further analysis revealed that ASERT had actually started receiving specimens as early as March 2010. ASERT has been using the moniker "YoYoDDoS" to refer to this family (derived from the host name of one of the initially observed C&C servers).

Malcode Properties

The YoYoDDoS malcode does not use a packer. It is most frequently encountered in the form of a 37,888-byte executable with various MD5 hash values. Some of the MD5 hash values that ASERT has observed to date include:

```
1f7dd0f7ba97823f1e74324b2171774b
e4a6e9911fe07f6df12c485001df8b2c
a5c29b7b0c77521d961e47c4fdad90b8
```

It is quite common for differences between individual samples to be quite small. For example, three different YoYoDDoS executables were identical except for the byte values at approximately 30 offset locations, some of which are below:

Offset	Sample-A	Sample-B
[0x00000fe4]	O	E
[0x00000fe5]	m	b
[0x00000fe6]	j	z
[0x00000ff2]	R	d
[0x00000ff3]	R	d
[0x000070b4]	S	G
[0x000070b5]	?	?
[0x000070b6]	?	?
[0x000070fd]	?	?

Fortunately, it is not uncommon for several different variants of the bot to have the same PEHash value.

[2] http://asert.arbornetworks.com/2010/08/YoYoDDoS-a-new-family-of-ddos-bots

The lifecycle of the bot is as follows:

1. The bot writes a copy of its own executable file into the C:\Windows\System32\ directory with a randomly generated file name. This file name begins with a capital A, followed by eight to ten random lower-case letters. ASERT has never observed two YoYoDDoS samples using the same name for this file. Examples observed include:

```
Antidzkue.exe
Asunumkzx.exe
Asynheuuf.exe
Antirwzwh.exe
Auuziixbguo.exe
```

The bot also sets the "hidden" and "system" attributes of this file. This is an exact copy of the original malware sample and is 37,888 bytes in size.

2. The bot opens the newly written copy of itself, and appends 53,716,000 bytes of data after the initial 37,888 bytes of data. The exact nature of this data has not been analyzed. The result is that the file's size grows to 53,753,888 bytes. The initial 37,888 bytes remain unchanged. It is not clear whether or not any of the 53 MB of appended data contains reachable instructions.

3. The bot then copies the large (53 MB) version of itself from the System32 directory to the Start Menu\Startup directory of the current user, using a name that starts with the string "360" followed by a random capitalized letter and four random lower-case letters. ASERT has never observed two YoYoDDoS samples using the same name for this file. ASERT has also observed this string being hard-coded within the bot executables. Representative examples observed include:

```
360Icaxv.exe
360Ytqol.exe
360Davsq.exe
360Omkhf.exe
360Kifdx.exe
```

The bot then modifies a single byte located at offset 1262, changing it from a value of 0x26 to 0x00.

4. The bot launches the appended copy of itself that it placed in the System32 directory, as well as the appended and modified version of itself placed in the user's Startup directory.

5. It launches a cmd.exe shell to delete the original copy of itself and then terminates.

6. At this point, the modified copy of itself, placed in the Startup directory, has started executing. It creates a mutex with a pseudo-random name that appears to be derived from the host computer name and possibly other seed information. It is not uncommon for different samples to use the same mutex name. Examples ASERT has observed include:

```
MRSG643GMVXGOLRTGMZDELTPOJTTUMJZHA2A====
NNWGIZDPOM4TOLRTGMZDELTPOJTTUMJQGIYA====
OJXXK2TJFZ2XKZDEN5ZS4Y3PNU5DGNRR
OJXXK2TJFZ2XKZDEN5ZS4Y3PNU5DGNRS
OV2XU2JOOZUXA5LVFZXGK5B2GM3DC===
PFXXK6LPOVSGILRTGMZDELTPOJTTUMRQGEYA====
```

7. The second process (living in the Startup directory) then initiates communications with the C&C master server and goes operational (see details below).

8. Meanwhile, the third process (launched from the copy placed in the System32 directory) creates an MS-DOS batch file that is always named C:\9.bat. This batch file contains the commands for creating a new Windows service associated with the hidden executable that it created in the C:\Windows\System32 directory. The service is configured to start automatically. Once this new service is created, it is started immediately, and the c:\9.bat file deletes itself.

The service is given a name that is more or less gibberish, although it sometimes begins with the word "Media." The display name of the service is generally some derivative of "MS Media Center." Sometimes the service names are enclosed in parentheses. Examples of the service names include:

```
MediaCubmkz
CeaxyaMencle
EnclCtwyme
MediaCwhlbh
```

Examples of service display names include:

```
(MS Media Conblur Center)
MS Mdeaia Congnvm Center Intele
Porconoswd Center
MS Media Conwwsk Center
```

The service is given a description that is some derivative of "support for media palyer. This service can't be stoped." (Note the spelling errors.) Examples include:

```
Prohbdrd support for media palyer. This service can't be stoped.
Support for media palyer. This service can't be stoped.
EnclCtwyme Media palyer. This service can't be stoped.
MediaCwhlbh Proulull support for media palyer. This service can't be stoped.
Meuuziexnzh Proktjbr suuuzit for media palyer. This service can't be stoped.
```

A typical example of the contents of the C:\9.bat file is as follows:

```
@ECHO OFF
SC CREATE CeaxyaMencle binPath= "C:\WINDOWS\system32\Asunumkzx.exe" START= auto
DISPLAYNAME= "MS Mdeaia Congnvm Center Intele" type= own type= interact
SC description CeaxyaMencle "Support for media palyer. This service can't be
stoped."SC START CeaxyaMencle
del %0
```

9. The third process then terminates.

10. Finally, a fourth process is initiated as a consequence of the start of the newly installed service. This process (associated with the copy living in the System32 directory) detects the existing mutex and aborts.

At this point, the malware is installed as both a service (to be automatically started at boot time) as well as a normal file residing in the user's Startup directory (also to be invoked automatically upon log in).

Communication Protocol

After installation, the bot connects to the C&C server and establishes a TCP connection. The C&C server appears to be hard-coded in the bot executable as a host name (not a bare IP address) and port, although the particular location and encoding method used has not been analyzed.

The bot usually initiates communications with its master by sending either a 232-byte (or less frequently, a 228-byte) message in binary format. Qualitatively, the message submitted by the bot client is very similar from specimen to specimen. In some cases, ASERT has observed a single bot specimen attempting to communicate to two C&C servers (presumably for redundancy purposes).

Here is a representative example of a typical 232-byte communication sent from the bot to the C&C:

```
6d 90 8a 99 92 ce 64 cd d6 5e 99 8f 90 ce 6a 71
cd d6 73 66 69 d6 c3 d0 c6 c8 77 6e 7c b6 b6 b6
b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6
b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6
b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6 b6
...
```

Although it has not fully reverse-engineered the communication protocol used by this family, ASERT has been able to determine that it uses a weak obfuscation scheme that appears to be a hard-coded substitution table. When de-obfuscated by applying this hard-coded substitution table, the message contains the following strings:

```
Intel(R) Xeon(TM) CPU 3280GHz
255MB
Win XP SP2
V201008UU
```

It is clear that the bot reports details about the victim host, including the make, model and speed of the processor, along with the operating system and service pack level. The last string (V201008UU in the above example) turns out to be a kind of identification (or version) string that is embedded in each bot variant's binary. The bot includes this string with each message it sends to the C&C. The identifier fails to show up in a static analysis on the original sample, but is easily revealed within the strings of a memory dump of a live YoYoDDoS process. Examples of these identification strings that ASERT has observed include:

```
V201008UU
V20100809
Vip090I03
LuoJ0810
```

The actual substitution table that ASERT has been able to reverse-engineer to date appears to be a modification of a simple substitution scheme that the team has seen previously. The table is constructed by starting with a mapping of the form: C = 0xB6 – P, where P is the plain text ASCII value of a character, and C is its "encrypted" representation. Thus, the null byte 0x00 is encoded as 0xB6, the ASCII value "A" is encoded as 0xB6 – 0x41 = 0x75, etc. However, once this basic layout is created, some block ranges of the table are relocated. For example, the block of (reversed) lower-case characters "z" through "a" is relocated to occupy byte codes 0x84 through 0x9d. Then a few smaller blocks of codes are again relocated within the larger relocated blocks.

DDoS Operation

If the C&C server is alive and responsive, it always sends a 124-byte message. The first four bytes represent a command code indicating the action that the bot client is requested to perform, followed by additional details. ASERT has observed the following different commands:

0x00100000: Initiate HTTP flood attack. The URL to attack is provided in plain text NULL-terminated ASCII starting at byte offset 4. For example:

```
00000000  00 10 00 00 68 74 74 70  3a 2f 2f 77 77 77 2e XX ....http ://www.v
00000010  XX XX XX XX XX XX XX XX  XX 2e 63 6f 6d 2f 69 6e ictimsit e.com/in
00000020  64 65 78 2e 61 73 70 00  00 00 00 00 00 00 00 00 dex.asp. ........
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000060  00 00 00 00 00 00 00 00  50 00 00 00 0a 00 00 00 ........ P.......
00000070  1e 00 00 00 64 00 00 00  c8 00 00 00             ....d... ....
```

In this example, the (anonymized) victim URL is www.victimsite.com/index.asp. The real victim in this particular attack was the Web site of a Chinese automotive parts merchant. The meaning of the remaining non-zero data bytes in the instructions is currently unknown.

In ASERT's observations, the bot generates HTTP traffic against the victim URL at a rate of approximately 20-25 GET requests per second.

0x80040000: Initiate a generic UDP+TCP/SYN DDOS attack. The host name or IP address to attack follows at byte offset 4. For example:

```
00000000  80 04 00 00 31 32 31 2e  31 32 2e 31 30 34 2e XX ....121. 12.104.X
00000010  XX 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 X....... ........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000060  00 00 00 00 00 00 00 00  c5 54 00 00 02 00 00 00 ........ .T......
00000070  1e 00 00 00 00 00 00 00  00 00 00 00             ........ ....
```

In this example, the target is the IP address 121.12.104.XX (last octet has been anonymized).

Upon receiving such a command, the bot immediately attempts to set up two TCP connections to the victim IP address on a port that appears to be random (e.g., 21701). As soon as these initial two SYN packets are sent off to the victim, the bot begins flooding the victim with UDP packets to the same port number.

During one observed attack, the bot sent UDP packets from two different source ports (1048 and 1049) to the same destination port (presumably chosen randomly). The packets sent from port 1048 contained 78 bytes of junk data (all bytes had the value 0x32, or "2"); the packets from port 1049 contained 111 bytes of junk (all bytes had the value 0x53, or "S"). These two 78- and 111-byte UDP data packets were repeated at a rate of approximately 260 packets/second until the victim responded with a SYN/ACK to one of the initial TCP connection requests, at which point the bot transitioned from UDP flooding to TCP SYN flooding. It is not clear whether the size of the UDP packets and/or their content were randomly chosen.

Once the victim responds to one of the initial SYN packets with a SYN/ACK, the bot will cease the UDP flood and initiate a SYN flood. Specifically, it will issue additional SYN packets from sequentially increasing source ports to initiate approximately 3-4 TCP connections per second. The bot executes a typical SYN flood attack. When the victim responds to each initial SYN packet with a SYN/ACK, the bot ignores the response and refuses to send an ACK to complete the three-way TCP handshake. Instead, it just keeps sending more SYN packets (from incrementally increasing source ports) to request more and more TCP connections.

In one observed attack, the bot managed to coerce the victim into partially opening more than 40 TCP connections over the course of about 12 seconds.

0x00000004: Download and launch an executable. The URL to the executable that is to be downloaded is provided at byte offset 4.

0xc1000000: Unknown command. ASERT observed this command, followed by a host name at byte offset 4. The bot then performed a DNS resolution of the specified host name. However, the logging of this sample terminated before any further actions were observed, so ASERT does not know what, if any, kind of attack is associated with this command code.

HTTP Header Signature

As mentioned above, if commanded to launch an HTTP flood attack, the bot will generate a large number of HTTP GET requests to the victim URL. The header fields in these GET requests are as follows:

```
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: g{ip, deflate
Host: www.victimsite.com:80
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Mozilla/4.1 (compatible; MSIE 6.0; Windows 5.1)
Referer: http://www.victimsite.com
Connection: Keep-Alivf
```

Note that the HTTP GET header fields generated by the bot contain several distinctive typos: the “Connection: Keep-Alivf” header is misspelled, and the “Accept-Encoding: g{ip, deflate” header has a typo. Both of these header values are present as hard-coded strings within the bot executable, which suggests that they are consistently present in the HTTP GET requests generated by this family.

Additional characteristics of the bot’s GET header include the User-Agent and Accept-Language fields, which are also hard-coded into the bot and are therefore expected to remain constant:

- Accept-Language: zh-cn
- User-Agent: Mozilla/4.1 (compatible; MSIE 6.0; Windows 5.1)

Furthermore, the bot has been observed to use the host name of the victim Web site for the “Referer” header field. Alternatively, an analysis of hard-coded strings within the bot executable suggests that it may also provide the header “Referer: http://www.google.com” in some cases instead.

Taken together, these distinctive GET header properties would most likely allow a very precise signature to be developed with a low probability of false positives. This signature would be able to identify inbound HTTP flooding traffic at a victim site.

Identification

The following interesting ASCII strings were found hard-coded within the bot executable; note that some of these strings suggest the possibility that the bot could have some limited root kit functionality:

```
\svchoSt.exe
JiangMin
c:\2.eXe
c:\winddk\demo\repairssdt\bin\i386\RepairSSDT.pdb
Possibly KiServiceLimit==%08X
relo type %d found at .%X
strange NtQuerySystemInformation()!
&KiServiceTable==%08X
Dumping ‘old’ ServiceTable:
Can’t find KeServiceDescriptorTabme
Can’t find KiServiceTable...
KeServiceDescriptorTablf
Start beep service ok
\drivers\PCIDump.sys
\\.\Dark2118
```

C&C Servers

To date, we have observed YoYoDDoS samples connected to at least 157 C&C server host names. Most of these C&Cs are located in China, with others in the U.S. and South Korea. The networks on which these C&Cs are located include:

ASN	ASN Name	Country	Network
27645	ASN-DCS-03	US	MANAGED SOLUTIONS GROUP INC
36351	SOFTLAYER	US	SOFTLAYER TECHNOLOGIES INC
3786	LGDACOM	KR	KOREA INTERNET DATA CENTER INC
4134	CHINANET	CN	CHINANET ANHUI PROVINCE NETWORK
4134	CHINANET	CN	CHINANET GUANGDONG PROVINCE NETWORK
4134	CHINANET	CN	CHINANET HUNAN PROVINCE NETWORK
4134	CHINANET	CN	CHINANET JIANGSU PROVINCE NETWORK
4134	CHINANET	CN	CHINANET-ZJ JINHUA NODE NETWORK
4134	CHINANET	CN	DONGGUANSHIWEIYIWANGLUOKEJIYOUX
4134	CHINANET	CN	JINHUA TELECOM CO. LTD IDC CENTER
4134	CHINANET	CN	NINBO LANZHONG NETWORK LTD
4134	CHINANET	CN	NINGBO LIGONG UNIVERSITY(2)
4134	CHINANET	CN	SHENZHENSHILUOHUQUHEPINGLUYIFENGGUANGCHANGCZUO32H
4134	CHINANET	CN	VA OFFICE BRANCH OF CHINA TELECOM CORP
4837	CHINA169	CN	CHINA UNICOM LIAONING PROVINCE NETWORK
4837	CHINA169	CN	ZHENGZHOU GIANT COMPUTER NETWORK TECHNOLOGY CO. LTD

DDoS Victims

To date, ASERT has observed over 3,000 victims of YoYoDDoS attacks, the majority of which are located in China. The breakdown by top 10 countries of the victim IP addresses is as follows:

Country	Victim Count
China	2178
United States	577
South Korea	114
Hong Kong	59
Japan	39
Taiwan	12
Canada	7
Cambodia	7
Germany	6
Australia	5

Anti-Virus Coverage

Approximately 50 percent of the YoYoDDoS samples that ASERT has observed do not exist at all in the Virus Total database. The remaining samples have AV detection results roughly similar to the following:

```
Anti-Virus      Version         Last Update  Result
a-squared       5.0.0.31        2010.07.09   Trojan.Win32.SystemHijack!IK
AhnLab-V3       2010.07.09.01   2010.07.09   Win-Trojan/Agent.53753888
AntiVir         8.2.4.10        2010.07.09   TR/Dropper.Gen2
Antiy-AVL       2.0.3.7         2010.07.09   -
Authentium      5.2.0.5         2010.07.09   W32/QQhelper.C.gen!Eldorado
Avast           4.8.1351.0      2010.07.09   Win32:Agent-AERY
Avast5          5.0.332.0       2010.07.09   Win32:Agent-AERY
AVG             9.0.0.836       2010.07.09   Dropper.Agent.RCT
BitDefender     7.2             2010.07.09   -
CAT-QuickHeal   11.00           2010.07.09   TrojanDropper.Agent.ayqh
ClamAV          0.96.0.3-git    2010.07.09   -
Comodo          5372            2010.07.09
TrojWare.Win32.TrojanDropper.Agent.~CSQ
DrWeb           5.0.2.03300     2010.07.09   BackDoor.Darkshell.96
eSafe           7.0.17.0        2010.07.08   -
eTrust-Vet      36.1.7694       2010.07.09   -
F-Prot          4.6.1.107       2010.07.08   W32/QQhelper.C.gen!Eldorado
F-Secure        9.0.15370.0     2010.07.09   -
Fortinet        4.1.143.0       2010.07.09   -
GData           21              2010.07.09   Win32:Agent-AERY
Ikarus          T3.1.1.84.0     2010.07.09   Trojan.Win32.SystemHijack
Jiangmin        13.0.900        2010.07.09   TrojanDropper.Agent.abzo
Kaspersky       7.0.0.125       2010.07.09   Trojan-Dropper.Win32.Agent.ayqh
McAfee          5.400.0.1158    2010.07.09   BackDoor-DKA
McAfee-GW-Edition               2010.1       2010.07.05
Heuristic.BehavesLike.Win32.Downloader.H
Microsoft       1.5902          2010.07.09   Trojan:Win32/SystemHijack.gen!C
NOD32           5264            2010.07.09   a variant of Win32/Farfli.AY
Norman          6.05.11         2010.07.09   W32/Obfuscated.FA
nProtect        2010-07-09.01   2010.07.09   Trojan-Dropper/W32.Agent.37888.BO
Panda           10.0.2.7        2010.07.08   Trj/Downloader.MDW
PCTools         7.0.3.5         2010.07.09   -
Prevx           3.0             2010.07.09   -
Rising          22.55.04.04     2010.07.09   Trojan.DL.Win32.SBXQ.a
Sophos          4.54.0          2010.07.09   Mal/Agent-AG
Sunbelt         6562            2010.07.09   BehavesLike.Win32.Malware (v)
Symantec        20101.1.0.89    2010.07.09   -
TheHacker       6.5.2.1.311     2010.07.08   Trojan/Dropper.Agent.ayqh
TrendMicro      9.120.0.1004    2010.07.09   -
TrendMicro-HouseCall            9.120.0.1004 2010.07.09  -
VBA32           3.12.12.6       2010.07.09   Backdoor.Win32.Httpbot.ahj
ViRobot         2010.6.29.3912  2010.07.09   Dropper.Agent.31744.I
VirusBuster     5.0.27.0        2010.07.08   -
```

Automated Identification

After manually identifying over 40 samples of YoYoDDoS, ASERT developed a simple automated identification system that classifies malware samples as YoYoDDoS based on the characteristics of the bot-to-controller communications protocol.

ASERT has not yet reverse-engineered the (apparently) obfuscated message sent from bot to C&C. The team has also observed multiple varieties of YoYoDDoS that exhibit a very similar communication pattern, but for which the actual bytes differ significantly. In addition, ASERT is still not sure which of the bytes that appear to be constant are actually variable and just derived from non-changing aspects of ASERT's sandbox environment (e.g., machine name). This made a hard-coded signature-based method for detecting YoYoDDoS communication difficult.

Shown below are two representative examples of bot-to-C&C messages sent by two different sub-YoYoDDoS varieties:

Observed March 26, 2010 (using a substitution table based on C = 0xB8 – P):

```
71 8E 84 95 8C D0 6A D1 D8 60 95 8F 8E D0 64 6D    “q.....j..`....dm”
D1 D8 7B 68 65 D8 CB CE C4 C8 77 70 82 B8 B8 B8    “..{he.....wp....”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
CA C5 C5 6D 7A B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “...mz...........”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
67 91 8E D8 60 68 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “g...`h..........”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
F1 F9 3F ED D8 D8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “..?.............”
B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8 B8    “................”
E1 B7 B8 B8                                          “....”
```

Observed August 3, 2010 (using a substitution table based on C = 0xB6 – P):

```
6D 90 8A 99 92 CE 64 CD D6 5E 99 8F 90 CE 6A 71    “m.....d..^....jq”
CD D6 73 66 69 D6 C3 D0 CA C6 77 6E 7C B6 B6 B6    “..sfi.....wn|...”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
C4 C9 C9 71 74 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “...qt...........”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
67 8D 90 D6 5E 66 D6 63 66 C4 B6 B6 B6 B6 B6 B6    “g...^f.cf.......”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
68 8D 86 C6 BD C6 78 C6 CA B6 B6 B6 B6 B6 B6 B6    “h.....x.........”
B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6 B6    “................”
AD BA B6 B6 FF FF FF FF                              “........”
```

Both are approximately the same length (228 and 232 bytes, respectively). Both have a single byte value that occurs very frequently (0xB8 and 0xB6, respectively), which ASERT believes to encode a null byte value (i.e., padding). And both have substantive (non-padding) bytes in approximately the same offset locations, which represent the specifications of the victim machine as well as the aforementioned identification string for the botnet.

Based on these commonalities, ASERT developed a simple algorithm for detecting YoYoDDoS traffic as follows:

- If the length of the message is not between 228 and 232 bytes, classify it as non-YoYoDDoS; otherwise continue.
- Find the most commonly occurring byte (called the "padding byte"). If the percentage of the entire message that consists of "padding bytes" is not between 70 percent and 80 percent, then classify it as non-YoYoDDoS; otherwise continue.
- Enumerate the offset locations of all the "substantive bytes" in the message and compare them to a canonical set of 59 substantive byte offsets observed in manually classified YoYoDDoS traffic. If the number of mismatched offsets is greater than 12, classify the sample as non-YoYoDDoS; otherwise classify it as YoYoDDoS.

So far, these simple heuristics are quite reliable at automatically identifying YoYoDDoS traffic without any false positives of which ASERT is aware.

Automated Tracking

Once ASERT had a mechanism for automatically identifying and tagging YoYoDDoS samples that were analyzed by the team's malware lab, ASERT engineers built a rudimentary YoYoDDoS botnet tracker. It periodically takes the complete set of positively identified YoYoDDoS samples from the malware lab and replays their 232-byte (or 228-byte in some cases) bot-to-C&C messages to their corresponding C&C hosts and ports. Since YoYoDDoS hard-codes its C&C server(s) as host names rather than IP addresses, the tracker checks to see if the DNS record associated with the C&C host name has changed to a different IP since the sample was actually dynamically analyzed. If so, then both the new and original IP addresses are tried.

For each C&C candidate IP, the tracker initiates a socket connection and, if successful, sends the original (replayed) communication message verbatim. It then listens for a response. If no response is received within 15 seconds, it gives up and moves on to the next C&C candidate. On the other hand, if the tracker receives a response, it attempts to parse it based on the command code set observed to date and mentioned above (e.g., 0x00100000 for HTTP flood, etc.).

ASERT's initial statistics with this tracker show that approximately two-thirds of the identified YoYoDDoS C&C servers were still live (in the sense that they are still accepting connections) one month after initial discovery. Of these, the majority tend to be dormant and not actively issuing commands, but ASERT's tracker regularly observes multiple HTTP flood (and occasionally other) attacks being issued from different botnets and/or C&C servers.

ASERT engineers have also experimented with replaying a message from a bot to its C&C, which they had manually modified to replace the bot's true identification string (e.g., V201008UU) with a different string. The motivation was to determine whether the C&C server infrastructure for a particular botnet uses this identification string as a kind of "password" or authentication token. Using the tracking mechanism described above, ASERT located a C&C that was actively engaged in an attack (i.e., issuing flood commands on a victim host in response to the verbatim replay of a message with correct ID string). The team then immediately replayed the modified version of that message to the C&C (with the bogus ID string). ASERT observed no change in the C&C behavior in response to the bogus message; the C&C issued the identical response (including the same attack command) as it issued to the unaltered bot transmission. Based on this experiment, it does not appear that the YoYoDDoS infrastructure attempts to authenticate the individual bot clients that report in to the botnets.

Conclusion

Research reveals that DDoS attacks are increasing in frequency, size and complexity. This is due in large part to the widespread availability of easy-to-use botnets to launch such an attack.

The Arbor Security Engineering & Response Team (ASERT)—a recognized leader in Internet threat analysis—is dedicated to monitoring botnets and DDoS attacks on a 24x7 basis. Using an automated process of discovery, research and analysis—ASERT continuously alerts organizations of emerging botnet and DDoS threats—such as the ASERT-discovered YoYoDDoS botnet—long before they impact business continuity, service availability and integrity.

For more information, visit the ASERT blog at asert.arbornetworks.com

Corporate Headquarters

6 Omni Way
Chelmsford, Massachusetts 01824

Toll Free USA +1 866 212 7267
T +1 978 703 6600
F +1 978 250 1905

Europe

T +44 207 127 8147

Asia Pacific

T +65 6299 0695

www.arbornetworks.com

ARBOR® NETWORKS

WP/AOAB/EN/0612