Case Study: FastDNS Infrastructure battles Xor Botnet

1.0 / OVERVIEW / Xor, a Trojan malware attackers are using to hijack Linux machines to include within a botnet for distributed denial of service (DDoS) campaigns, appears to be behind an Oct. 13 DNS flood against a customer using Akamai’s FastDNS infrastructure.

This attack campaign started with a DNS Flood of 30 million queries per second and escalated into a Tsunami SYN Flood ramping up to 140 Gbps with over 75 Mpps in total. All attack signatures match with the recently investigated Xor.DDoS Botnet.

Between Oct. 13 and 23rd, the attack was constantly switching on and off. The attack hit multiple destination hosts at the same time.

Below is an outline of the attack characteristics and defensive measures as observed by Akamai’s Security Intelligence Research Team (SIRT).

In the course of this investigation, the SIRT worked with Akamai’s FastDNS team. The FastDNS team noticed considerable attack traffic while XoR was not targeted at us. It’s possible the adversary is employing a multi-vendor DDoS approach and that all the DNS traffic we see is attributable to XoR. That said, we are reasonably certain that Xor is behind all the SYN flood activity.

2.0 / ATTACK CHARACTERISTICS / As explained in Section 3.0 of the Xor Threat Advisory released in late September, SYN and DNS floods generated by the Xor.DDoS Malware have very specific characteristics.

First, a SYN flood always consists of the following 8 static TCP header options.

<table>
<thead>
<tr>
<th>Header Options</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flags</td>
<td>0x00</td>
</tr>
<tr>
<td>Source Port</td>
<td>0x00</td>
</tr>
<tr>
<td>Destination Port</td>
<td>0x00</td>
</tr>
<tr>
<td>Sequence</td>
<td>0x00</td>
</tr>
<tr>
<td>Window Size</td>
<td>0x00</td>
</tr>
<tr>
<td>Checksum</td>
<td>0x00</td>
</tr>
<tr>
<td>Urgent Pointer</td>
<td>0x00</td>
</tr>
<tr>
<td>Options</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Figure 1: Static TCP headers

<table>
<thead>
<tr>
<th>IP Address</th>
<th>15:55:58.842287</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flags</td>
<td>0x00</td>
</tr>
<tr>
<td>Source Port</td>
<td>0x00</td>
</tr>
<tr>
<td>Destination Port</td>
<td>0x00</td>
</tr>
<tr>
<td>Sequence</td>
<td>0x00</td>
</tr>
<tr>
<td>Window Size</td>
<td>0x00</td>
</tr>
<tr>
<td>Checksum</td>
<td>0x00</td>
</tr>
<tr>
<td>Urgent Pointer</td>
<td>0x00</td>
</tr>
<tr>
<td>Options</td>
<td>0x00</td>
</tr>
</tbody>
</table>

TLP: GREEN
11.12.2015
15:55:48.84219 IP xxx.187.0.184 > y.y.y.y.53: Flags [S], seq 2021059849:2021060856, win 65535, length 10071460+ 

Figure 2: Captured sample #1
We see a 100% duplicate payload, confirming the spoofing capabilities explained in section 4 of the Xor advisory.
The two packets are generated by the same machine while spoofing only the last octet of its real public IP address. The payload consists of garbage memory data of what happens to be in the used memory space. The memory can also contain passwords and ssh private keys.

In the DNS Flood attack payloads, we see all distinguishable characteristics as well.

As you can see, the observed DNS query flood is sourcing from the same source IP, with the same source port and the same query ID targeting multiple domains. The structure of all the packets might also look similar. This is because all of the target domains are passed from the C&C to the bots in a single command. From there the infected machines build the same template for all of the target domains.

We also see the Additional Record Count bit set across all packets and the DNSSEC options at the end.

Another not-so-important characteristic we noticed is that sometimes the DNS queries include the RD (Recursion Desired) flag on and sometimes it’s off.

3.0 / RECOMMENDED DETECTION METHODS/ To detect this botnet in your network, one can look for the communications between the bot and C2 using the following Snort rule:

```
alert TCP $HOME_NET any -> $EXTERNAL_NET any ( msg: "Xor-DDoS";
flow: established;
content: "BB2FA36AAA9541F0BB2FA36AAA9541F0";
offset:32; depth: 64;
classification: trojan-activity;
sid: 201500010; rev: 1;)
```

Figure 5: Snort rule matches on the XOR key string within the appropriate offsets in the initial registration between bot and C2
To detect infection of this malware on your hosts you can use the following Yara rule:

```yara
rule XorDDoSv1 {
    meta:
        author = "Akamai SIRT"
        description = "Rule to detect XorDDoS infection"
    strings:
        $st0 = "BB2FA36AAA9541F0"
        $st1 = "md5="
        $st2 = "denyip="
        $st3 = "filename="
        $st4 = "rmfile="
        $st5 = "exec_packet"
        $st6 = "build_iphdr"
    condition:
        all of them
}
```

Figure 6: Yara Rule

Here we used strings noticed in the binary itself:

- string $st0 - ASCII XOR key
- strings $st1 - $st4 are parameters used by functionalities in the binary related to trojan activities.
- string $st5 - function name to execute DDoS flood
- string $st6 - function name to build IP header

To match SYN Flood attack traffic generated by this botnet you can use the following tcpdump filters:

```bash
```

Figure 7: Example tcpdump filters for SYN flood attack traffic

This filter is composed of the following BPF filters:

- 'ip[2:2] > 0x50' - matches on total size field greater than 80 bytes
- 'tcp[13] & 2 !=0' - matches on SYN flags
- 'tcp[14:2] == 0xffff' - matches on TCP Window size 65535
- 'tcp[20:4] == 0x020405b4' - matches on static bytes used for TCP header options
- 'tcp[24:4] == 0x01010402' - matches on static bytes used for TCP header options

To match DNS Flood attack traffic generated by this botnet you can use the following tcpdump filters:

```bash
'udp[10:4] == 0x01200001' and 'udp[14:4] == 0x00000000' and 'udp[19] == 0x0001' and
'ip[((ip[2:2]) - 14):1] = 0x01'
```

Figure 8: Example tcpdump filters for DNS flood attack traffic

The above BPF filter is composed of:

- 'udp[10:4] == 0x01200001' - matches on the specific DNS fields from the outlisted characteristics in Section 3
- 'udp[14:4] == 0x00000000' - matches on Answer Count and NS (Auth) Count fields
- 'udp[19] == 0x0001' - matches on Additional Record Count
- 'ip[((ip[2:2]) - 14):1] = 0x01' - matches on the DNS Query Type field for ‘A’ records
To remove the malware:

- Identify the main PID and the dropped files in /boot and /etc/init.d

```bash
root@ubuntu:/# ls -la /boot | egrep "[a-z]{10}$"
-rwxr-x-- 1 root root 606K Aug 19 11:42 utmhsahoca

root@ubuntu:/# ls -la /etc/init.d/utmhsahoca
-rwxr-x-- 1 root root 28 Aug 19 11:42 /etc/init.d/utmhsahoca

root@ubuntu:/# find /proc/ -name exe 2> /dev/null | xargs -I{} ls -l {} 2> /dev/null | egrep '/boot/[a-z]{10}$'
```

```bash
lrwxrwxrwx 1 root root 0 Aug 19 11:42 /proc/3746/exe > /boot/utmhsahoca
```

Figure 9: Bash script used in the /proc file system process

The above bash script looks through all of the processes in the /proc file system and gets the path of their symlinked source finding all processes started from an executable in /boot with a 10-character name.

Here we have identified that /boot/utmhsahoca is currently running user PID 3746.

The above script in more readable form and commented form:

```bash
# find all executing processes
1. find /proc/ -name exe 2> /dev/null

# get each processes symlinked destination
2. xargs -I{} ls -l {} 2> /dev/null

# find symlinks that match our '/boot/xxxxxxxxxx' signature
3. egrep '/boot/[a-z]{10}$'
```

Figure 10: Additional Bash script

The PIDs 8318, 8321, 8324, 8326, 8327 are short lived and their purpose is to only check if the main process is running. We are going to eliminate them by their start time, which will always be very close to the current time. If this command is run multiple times over the span of a minute the observed PIDs and associated fake commands will constantly change.

3.1 KILLING THE MALICIOUS PROCESS /

```bash
root@ubuntu:/boot# ps -eaf -u root | grep `date +%H:%M` | egrep -v "ps|grep" | grep -v "ps|grep" | awk '{print $2}' | xargs -I{} kill -9 {}; kill -9 3746
```

Figure 12: Bash script in more readable form
```bash
# get processes
1. ps -eaf

# only get root users processes
2. grep 'root'

# only keep processes started within the current minute
3. grep `date +%H:%M`

# remove grep and ps from list from the list
4. egrep -v "ps|grep"

# get the PID of the process
5. awk '{print $2}'

# force-kill the PID supplied for subprocesses
6. xargs -I{} kill -9 {}

# kill the main process
7. kill -9 3746
```

**Figure 13: Additional Bash script**

This script will get all processes running under the root user, it will then grab a subset of those processes that have spawned within the last minute, removing our own `ps` and `grep` from the list. It then extracts the PID for each process and kills it using `-9`, which cannot be blocked, prevented, or ignored. Once it has killed all of the watcher processes it kills off the master process, in this case PID **3746**.

Next we’ll need to delete all of the malware files that have been dropped to disk.

```bash
root@ubuntu:/boot# rm -f /boot/utmhsahoca && rm -f /etc/init.d/utmhsahoca && /lib/udev/udev
```

**Figure 14: Delete the dropped files in /boot and /etc/init.d**

If you failed to kill the running instances of the malware, after step 4 the malware will drop a new copy of itself with a new 10-character randomized name in /boot and new startup script in /etc/init.d. If new files exist repeat steps 1-4 to ensure you’ve killed all running instances. You can also use the above YARA rule (Figure 6) on the whole file system to see if any copies of the malware are still present.
4.0 / CONCLUSION / Akamai’s SIRT expects XOR DDoS activity to continue as attackers refine and perfect their methods.

This will likely result in a more diverse selection of DDoS attack types included in future versions of the malware. XOR DDoS malware is part of a wider trend of which companies must be aware: Attackers are targeting poorly configured and unmaintained Linux systems for use in botnets and DDoS campaigns.

A decade ago, Linux was seen as the more secure alternative to Windows environments, which suffered the lion’s share of attacks at the time, and companies increasingly adopted Linux as part of their security-hardening efforts.

As the number of Linux environments has grown, the potential opportunity and rewards for criminals has also grown. Attackers will continue to evolve their tactics and tools and security professionals should continue to harden their Linux based systems accordingly.