Radware Security Research

Reverse Engineering a Sophisticated DDoS Attack Bot

Author: Zeev Ravid

July 2015

Introduction

In July 2015, Radware's Emergency Response Team (ERT) noticed a significant increased usage of the **Tsunami SYN Flood attack** against a large customer.

During the investigation of these attacks, Radware Security Researchers obtained a sample of a malware binary which was used to generate these DDoS attacks.

Radware's ERT deployed the malware in an isolated and controlled environment to study its behavior and its different attack vectors.

During a period of 10 days (June 14-23, 2015), Radware's ERT monitored greater than 2000 attacks against more than 60 different targets across 7 different countries.

This report presents an analysis of the attack, the sophisticated attack tool and an inside look into a Tsunami DDoS attack.

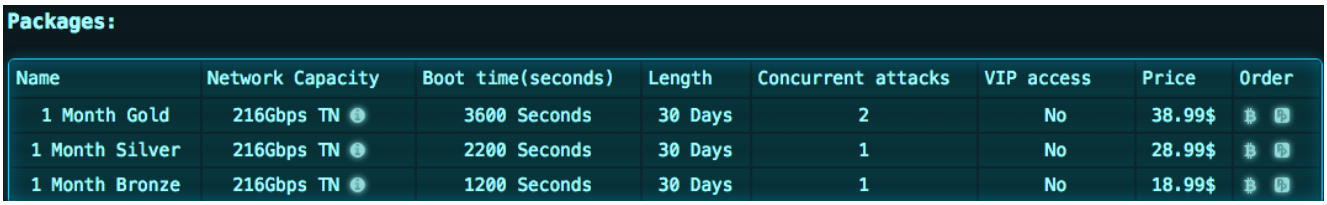
DDoS-as-a-Service

Upon analysis, the attack patterns of this DDoS bot indicates that the attacks were likely initiated through a professional malware operator who provides "DDoS for hire"[1] services, selling its DDoS capabilities to random bidders.

Even if one lacks the relevant expertise, it's easy and inexpensive to launch an attack.

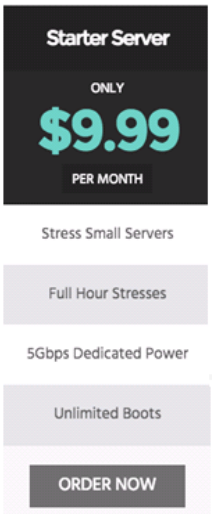
There are increasingly large numbers of "shady" websites providing easy-to-use web interfaces that can be used to launch DDoS attacks. Dozens of websites, referring to themselves as "booter" or "stresser," openly advertise themselves on the Internet and accept payments with Bitcoin or PayPal, some starting as low as $9.99 per month.

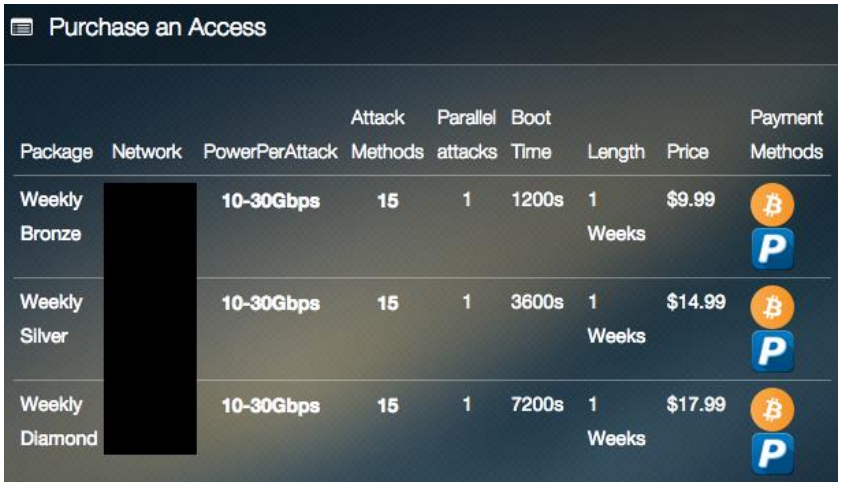
Below are sample screenshots:



Packages:

Name	Network Capacity	Boot time(seconds)	Length	Concurrent attacks	VIP access	Price	Order
1 Month Gold	216Gbps TN	3600 Seconds	30 Days	2	No	38.99$	
1 Month Silver	216Gbps TN	2200 Seconds	30 Days	1	No	28.99$	
1 Month Bronze	216Gbps TN	1200 Seconds	30 Days	1	No	18.99$	





Purchase an Access

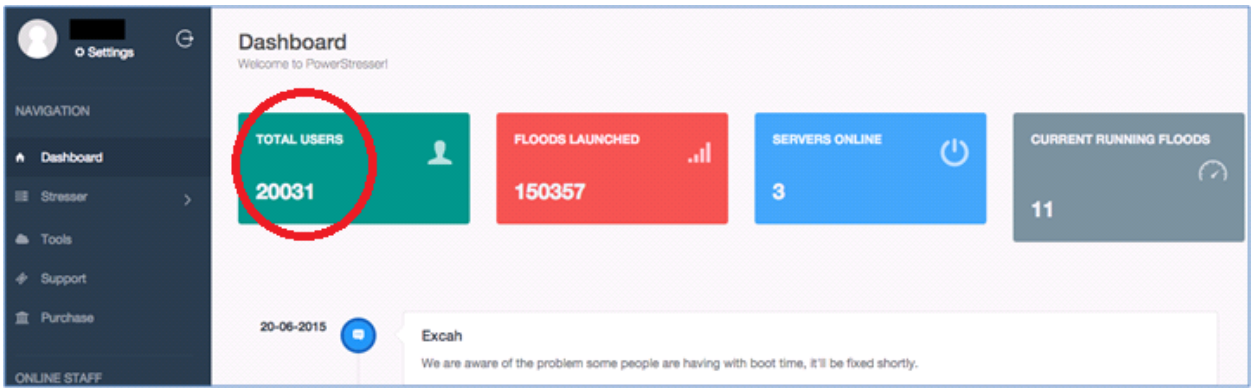
Package	Network	PowerPerAttack	Attack Methods	Parallel attacks	Boot Time	Length	Price	Payment Methods
Weekly Bronze	[redacted]	10-30Gbps	15	1	1200s	1 Weeks	$9.99	
Weekly Silver	[redacted]	10-30Gbps	15	1	3600s	1 Weeks	$14.99	
Weekly Diamond	[redacted]	10-30Gbps	15	1	7200s	1 Weeks	$17.99	

[1] http://krebsonsecurity.com/category/ddos-for-hire/

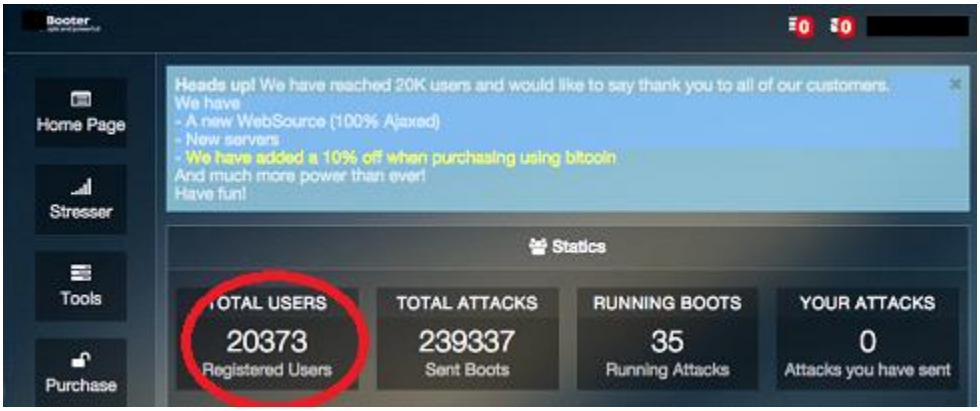
To launch attacks with a DDoS for hire service, registered users gain access to the admin panel and additional features like attack history, network statistics, and a ticket system for opening customer support incidents.

Many of those services have more than 20,000 registered users:

Panel 1

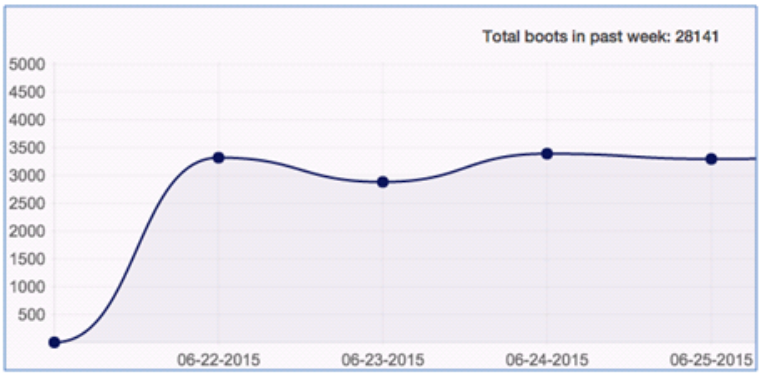


Panel 2

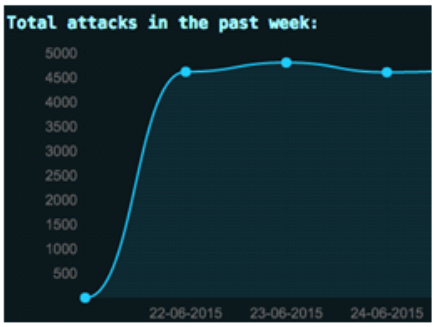


Most of the panels also show global system statistics for nearly 5,000 DDoS attacks per week:

Panel1



Panel2



The charts in Panel 1 and Panel 2 above shows that these systems are truly capable of delivering a 10+ Gbps attack, which could become a significant threat to a critical business system.

About the Tsunami SYN Flood Attack

The Tsunami SYN Flood involves sending a massive amount of TCP SYN packets with a non-zero payload, unlike the common SYN packets which don't contain data.

Unfortunately, it is easy for an attacker to create SYN packets and use them to launch a volumetric DDoS attack. This attack can saturate the Internet connection of your organization even before hitting any stateful network device like the Firewall and Load Balancer.

More detailed information about the Tsunami SYN attack can be found here.

Inside Look at a DDoS Attack

The image below shows network traffic captured during the DDoS attack. Notice the SYN packets with the Len=896 bytes payload highlighted in red. As one can see, the attacker was sending a high rate of SYN packets.

Time	Source	Destination	Protoco	Length	Info
0.743769	52.17.172.123	202.165.[redacted]	TCP	950	63266→4805 [SYN, CWR, Reserved] Seq=0 Win=65535 Len=896
0.743772	61.175.73.197	202.165.[redacted]	TCP	950	52686→4805 [SYN, ECN, CWR, Reserved] Seq=0 Win=65535 Len=896
0.743774	120.209.2.242	202.165.[redacted]	TCP	950	4377→4805 [SYN] Seq=0 Win=65535 Len=896
0.743778	220.170.135.196	202.165.[redacted]	TCP	950	48206→4805 [SYN] Seq=0 Win=65535 Len=896
0.743780	52.17.172.123	202.165.[redacted]	TCP	950	28418→4805 [SYN, CWR, Reserved] Seq=0 Win=65535 Len=896
0.743783	52.11.236.162	202.165.[redacted]	TCP	950	56080→4805 [SYN] Seq=0 Win=65535 Len=896
0.743785	52.74.83.156	202.165.[redacted]	TCP	950	20035→4805 [SYN, ECN, Reserved] Seq=0 Win=65535 Len=896
0.743813	120.209.2.242	202.165.[redacted]	TCP	950	29125→4805 [SYN] Seq=0 Win=65535 Len=896
0.743817	218.22.106.142	202.165.[redacted]	TCP	950	8714→4805 [SYN] Seq=0 Win=65535 Len=896
0.743820	211.142.50.135	202.165.[redacted]	TCP	950	4051→4805 [SYN] Seq=0 Win=65535 Len=896
0.743822	120.209.2.242	202.165.[redacted]	TCP	950	11477→4805 [SYN] Seq=0 Win=65535 Len=896
0.743833	52.11.236.162	202.165.[redacted]	TCP	950	60534→4805 [SYN] Seq=0 Win=65535 Len=896
0.743835	58.49.59.192	202.165.[redacted]	TCP	950	36161→4805 [SYN, Reserved] Seq=0 Win=65535 Len=896
0.743846	52.17.172.123	202.165.[redacted]	TCP	950	41875→4805 [SYN, CWR, Reserved] Seq=0 Win=65535 Len=896
0.743848	120.209.2.242	202.165.[redacted]	TCP	950	17786→4805 [SYN] Seq=0 Win=65535 Len=896
0.743854	52.17.172.123	202.165.[redacted]	TCP	950	63613→4805 [SYN, CWR, Reserved] Seq=0 Win=65535 Len=896
0.743856	52.17.172.123	202.165.[redacted]	TCP	950	39362→4805 [SYN, CWR, Reserved] Seq=0 Win=65535 Len=896
0.743858	120.209.2.242	202.165.[redacted]	TCP	950	44748→4805 [SYN] Seq=0 Win=65535 Len=896

The table on the following page shows a list of the top attackers' IP addresses and the number of SYN packets they've sent over a period of about 70 seconds.

# Attacks	Attacker IP	%	Sum	Total # Attacks
645,829	120.209.2.242	30.07%	30.07%	2,148,062
337,753	52.17.172.123	15.72%	45.79%	
170,457	52.11.236.162	7.94%	53.72%	
119,306	52.74.147.146	5.55%	59.28%	
113,771	58.49.55.241 ... 58.49.61.239	5.30%	64.58%	
113,211	52.74.83.156	5.27%	69.85%	
93,972	52.28.27.34	4.37%	74.22%	
91,363	61.175.38.27 ... 61.175.198.25	4.25%	78.47%	
76,679	211.142.27.8 ... 211.142.86.162	3.57%	82.04%	
73,304	52.16.7.159	3.41%	85.46%	
72,358	218.22.106.142	3.37%	88.82%	
56,885	120.193.178.98 ... 120.193.178.116	2.65%	91.47%	
51,927	52.74.177.255	2.42%	93.89%	
49,710	61.175.225.134	2.31%	96.20%	
44,191	220.191.251.11	2.06%	98.26%	
13,205	220.170.135.196	0.61%	98.88%	
7,177	203.191.146.248 ... 203.191.151.246	0.33%	99.21%	
4,053	218.23.149.51	0.19%	99.40%	
3,378	120.209.116.50	0.16%	99.56%	
465	211.142.22.164 ... 211.142.22.255	0.02%	99.58%	

The IPs highlighted in yellow appear to be real servers with real IPs, and responsible for generating over 80% of the traffic. Most of the real IPs are hosted on the Amazon EC2 network (52.*.*.*). The rest of the traffic was spoofed, and used more than 50,000 random IPs belonging to about six different large network segments.

This investigation suggests that the spoofed traffic was generated by 10-15 compromised or rented servers that were all running on hosting providers which do not prevent their clients from sending spoofed IP traffic.

The next section will provide analysis of the tool that has been used for generating these attacks.

Tracking Down the Attackers

Radware's ERT deployed the malware in an isolated and controlled environment to study its behavior and attack vectors.

Upon close examination, it was discovered that the bot was communicating with an active C&C (Command & Control) server, which had been instructing it to launch repeated attacks against different targets. The duration of each attack was between several minutes to several hours.

During a period of 10 days (June 14-23, 2015), we monitored greater than 2000 attacks against more than 60 different targets. The duration of each attack was between several minutes to several hours.

The following list of C&C servers is hardcoded in the binary:

1. **66.102.253.30**
2. **zhegege.3322.org**
3. **122.224.48.63**

The targets of the attack are mostly servers located in China:

# Targets	Country
51	China
4	Philippines
5	Hong Kong
2	United States
1	Thailand
1	Malaysia
1	Japan

Attacked targets include:

IP	CC	Country	Network	Suffix	ASNum	Name
208.89[redacted]	US	United States	208.89[redacted]	::22	AS11[redacted]	Li[redacted]
120.52[redacted]	CN	China	120.52[redacted]	::16	AS13[redacted]	C[redacted]
43.225[redacted]	MY	Malaysia	43.225[redacted]	::22	AS13[redacted]	C[redacted]
119.28[redacted]	CN	China	119.28[redacted]	::19	AS13[redacted]	C[redacted]
117.27[redacted]	CN	China	117.27[redacted]	::22	AS13[redacted]	Fu[redacted]
112.10[redacted]	CN	China	112.10[redacted]	::15	AS17[redacted]	as[redacted]
202.16[redacted]	PH	Philippines	202.16[redacted]	::23	AS18[redacted]	G[redacted]
45.64.6[redacted]	HK	Hong Kong	45.64.6[redacted]	::24	AS19[redacted]	In[redacted]
61.147[redacted]	CN	China	61.147[redacted]	::23	AS23[redacted]	A[redacted]
161.20[redacted]	JP	Japan	161.20[redacted]	::18	AS36[redacted]	S[redacted]
120.55[redacted]	CN	China	120.55[redacted]	::16	AS37[redacted]	H[redacted]
210.56[redacted]	HK	Hong Kong	210.56[redacted]	::20	AS38[redacted]	S[redacted]
182.14[redacted]	CN	China	182.14[redacted]	::18	AS38[redacted]	C[redacted]
45.34.5[redacted]	US	United States	45.34.5[redacted]	::16	AS40[redacted]	Ps[redacted]
122.22[redacted]	CN	China	122.22[redacted]	::14	AS41[redacted]	C[redacted]
113.21[redacted]	PH	Philippines	113.21[redacted]	::23	AS45[redacted]	M[redacted]
122.19[redacted]	CN	China	122.19[redacted]	::14	AS48[redacted]	C[redacted]
103.26[redacted]	TH	Thailand	103.26[redacted]	::22	AS55[redacted]	60[redacted]
14.29.3[redacted]	CN	China	14.29.3[redacted]	::19	AS58[redacted]	G[redacted]
118.19[redacted]	CN	China	118.19[redacted]	::18	AS58[redacted]	Sh[redacted]
43.227[redacted]	CN	China	43.227[redacted]	::19	AS59[redacted]	N[redacted]
120.20[redacted]	CN	China	120.20[redacted]	::16	AS98[redacted]	G[redacted]

Every day different servers are attacked on different ports:

Count	Date	IP	Port
1	6/25/2015	116.31[redacted]	80
3	6/25/2015	117.27.2[redacted]	80
3	6/25/2015	117.27.[redacted]	80
2	6/25/2015	117.27.[redacted]	80
54	6/25/2015	118.193.1[redacted]	80
1	6/25/2015	118.193.[redacted]	80
1	6/25/2015	122.193.[redacted]	80
12	6/25/2015	122.228.2[redacted]	80
9	6/25/2015	122.228.2[redacted]	9800
31	6/25/2015	14.29[redacted]	80
81	6/25/2015	14.29[redacted]	80
2	6/25/2015	203.202.[redacted]	80
10	6/25/2015	218.90.[redacted]	80
4	6/25/2015	42.19[redacted]	80
3	6/25/2015	42.19[redacted]	80
26	6/25/2015	42.19[redacted]	80
15	6/25/2015	43.227[redacted]	1520
8	6/25/2015	43.227[redacted]	22
34	6/25/2015	61.154.1[redacted]	9800

Analyzing the Command Packets Between the Bot and C&C

The bot periodically sends information about itself to the C&C server. The data is "encrypted" with a fixed 16 bytes XOR key (BB2FA36AAA9541F0).

This is what a sample packet looks like – highlighted in red is the current C&C server address:

```
16:29:18.441121 IP 192.168.1.100.59528 > 122.224.51.128.2866: Flags [P.], seq 1295236316:1295
236588, ack 154772684, win 46, options [nop,nop,TS val 1223626769 ecr 13131496], length 272
        0x0000:  4500 0144 b04d 4000 3f06 19fa c0a8 0164  E..D.M@.?......d
        0x0010:  7ae0 3380 e888 0b32 4d33 bcdc 0939 a4cc  z.3....2M3...9..
        0x0020:  8018 002e e147 0000 0101 080a 48ef 1011  .....G......H...
        0x0030:  00c8 5ee8 b041 3246 a930 3641 bebe c6ca  ..^..A2F.06A....
        0x0040:  061f 701e 737a 1f77 7807 1824 2d74 3935  ..p.sz.wx..$-t95
        0x0050:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x0060:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x0070:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x0080:  3458 7008 7442 3246 4133 3641 4141 3935  4Xp.tB2FA36AAA95
        0x0090:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x00a0:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x00b0:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x00c0:  3431 741a 7072 0276 4133 3641 4141 3935  41t.pr.vA36AAA95
        0x00d0:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x00e0:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x00f0:  3431 4630 4242 3246 4133 3641 4141 3935  41F0BB2FA36AAA95
        0x0100:  3431 4649 2730 5a33 384a 5534 3828 435a  41FI'0Z38JU48(CZ
        0x0110:  4147 2357 263b 513f 3255 5b34 3739 4847  AG#W&;Q?2U[479HG
        0x0120:  5d46 3430 1116 7312 0870 3641 4141 3935  ]F40..s..p6AAA95
        0x0130:  3431 4630 706c 0268 7133 3641 4141 3935  41F0pl.hq36AAA95
        0x0140:  3431 4630                                41F0
```

After decryption, this is the data that is being sent:

```
0000000: f2 03 00 00 e8 03 00 00 ff ff ff ff 32 2e 36 2e  ............2.6.
0000010: 31 38 2d 31 39 34 2e 65 6c 35 00 00 00 00 00 00  18-194.el5......
0000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 69 36 38  .............i68
0000050: 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  6...............
0000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 32 2a  ..............2*
0000090: 32 30 30 30 00 00 00 00 00 00 00 00 00 00 00 00  2000............
00000a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 79  ...............y
00000d0: 65 72 68 75 79 79 63 75 79 69 7a 6f 75 76 65 67  erhuyycuyizouveg
00000e0: 64 79 63 79 73 66 6d 75 76 78 71 72 69 77 72 00  dycysfmuvxqriwr.
00000f0: 53 54 41 54 49 43 00 00 00 00 00 00 00 00 00 00  STATIC..........
0000100: 32 2e 30 2e 30 00 00 00 00 00 00 00 00 00 00 00  2.0.0...........
```

It includes information about the current system:

- Linux 2.6.18-194.el5,
- 2x2000 MHz CPUs
- 32 bit i686 processor

After few seconds, the C&C server sends a special packet instructing the bot to **attack.** This would be a 304 bytes TCP packet that looks as follows:

```
  1   0.000000 122.224.51.128 -> 192.168.1.100 TCP 370 2866 > 59528 [PSH, ACK]
Seq=1 Ack=1 Win=65235 Len=304 TSval=13131659 TSecr=1223642011

0000  00 0c 29 c5 bb 4e 00 16 e3 f1 47 5a 08 00 45 00   ..)..N....GZ..E.
0010  01 64 5d 0e 40 00 71 06 3b 19 7a e0 33 80 c0 a8   .d].@.q.;.z.3...
0020  01 64 0b 32 e8 88 09 39 a5 04 4d 33 bd ec 80 18   .d.2...9..M3....
0030  fe d3 28 9c 00 00 01 01 08 0a 00 c8 5f 8b 48 ef   ..(.........._.H.
0040  4b 9b 09 89 aa 61 14 01 00 00 03 00 00 00 01 00   K....a..........
0050  00 00 3c 00 00 00 e8 1e       e8 1e       35 d3   ..<......0...05.
0060        11 33 36 41 41 41 39 35 34 31 46 30 42 42   ...36AAA9541F0BB
0070  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0080  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0090  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
00a0  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
00b0  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
00c0  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
00d0  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
00e0  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
00f0  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0100  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0110  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0120  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0130  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0140  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0150  32 46 41 33 36 41 41 41 39 35 34 31 46 30 42 42   2FA36AAA9541F0BB
0160  32 46 41 33 36 41 44 41 39 35 36 31 46 30 a5 41   2FA36ADA9561F0.A
0170  32 46                                             2F
```

After decryption of the payload data:

Green – the bot external IP address as seen by the C&C server

Red – the target IP address 0xb783XXXX (**183.131.X.X**) on port 0x0050 (port 80)

Blue – attack method – 05 means SYN flood. 0x03e7 means packet size – 999 bytes.

```
0000000: 09 89 aa 61 14 01 00 00 03 00 00 00 01 00 00 00  ...a............
0000010: 3c 00 00 00 e8 1e       e8 1e       b7 83       <......0...0..EJ
0000020: 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  P...............
0000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000120: 00 00 00 00 05 00 00 00 02 00 00 00 e7 03 00 00  ................
```

Although the bot is capable of launching different attack methods, in this report the bot was only using the “05” method – e.g. Tsunami SYN flood, with different packet sizes.

Shortly after receiving this message, the bot will start attacking his target by sending large SYN packets:

```
2015-06-25 16:55:24.440250 IP 192.168.1.100.23803 > 183.131.     .80: Flags [S], seq 1559991420:1559992427, win 65535, length 1007
2015-06-25 16:55:24.640225 IP 192.168.1.100.39869 > 183.131.     .80: Flags [S], seq 2612919604:2612920611, win 65535, length 1007
2015-06-25 16:55:24.840229 IP 192.168.1.100.15461 > 183.131.     .80: Flags [S], seq 1013254509:1013255516, win 65535, length 1007
2015-06-25 16:55:25.040239 IP 192.168.1.100.43653 > 183.131.     .80: Flags [S], seq 2860887050:2860888057, win 65535, length 1007
2015-06-25 16:55:25.240231 IP 192.168.1.100.21145 > 183.131.     .80: Flags [S], seq 1385812009:1385813016, win 65535, length 1007
2015-06-25 16:55:25.440242 IP 192.168.1.100.26337 > 183.131.     .80: Flags [S], seq 1726030938:1726031945, win 65535, length 1007
2015-06-25 16:55:25.640236 IP 192.168.1.100.30124 > 183.131.     .80: Flags [S], seq 1974250085:1974251092, win 65535, length 1007
2015-06-25 16:55:25.840232 IP 192.168.1.100.11094 > 183.131.     .80: Flags [S], seq 727118084:727119091, win 65535, length 1007
2015-06-25 16:55:26.040325 IP 192.168.1.100.41203 > 183.131.     .80: Flags [S], seq 2700343609:2700344616, win 65535, length 1007
2015-06-25 16:55:26.241560 IP 192.168.1.100.39462 > 183.131.     .80: Flags [S], seq 2586214752:2586215759, win 65535, length 1007
2015-06-25 16:55:26.440228 IP 192.168.1.100.12101 > 183.131.     .80: Flags [S], seq 793112700:793113707, win 65535, length 1007
```

Network Isolation

The previous screenshot shows a very slow attack, running at 5 packets per second. The image above was produced in a controlled sandbox environment, developed by Radware's ERT for researching and monitoring malware activity without the risk of participation in any attack activity.

Without the limited sandbox environment, the malware would have sent about 100,000 packets per second, utilizing the entire 1Gbps virtual network port. The screenshot below shows the OS statistics on VM the malware was running on – the 105 MB/sec outgoing traffic is highlighted in red:

-----time-----	-net/total-	
date/time	recv	send
23-06 18:13:31	264B	1208B
23-06 18:13:32	502B	27M
23-06 18:13:33	66B	91M
23-06 18:13:34	132B	-
23-06 18:13:35	66B	105M
23-06 18:13:36	66B	103M
23-06 18:13:37	66B	102M
23-06 18:13:38	66B	100M
23-06 18:13:39	66B	101M
23-06 18:13:40	66B	105M

DDoS Bot Performs Self-Diagnostic Tests

The malware, together with the C&C, is engaging in sophisticated 'self-tests' to both check its ability to send spoofed IP traffic and perform a 'self-benchmark' test to checking its flooding ability.

In the image below, one can see the malware running on 10.0.5.105, communicating with Google's DNS server (8.8.8.8) to resolve the C&C domain (**zhegege.3322.org**), which results in the IP **122.224.51.128**. Then it begins to send UDP packets from different 'spoofed' IPs, targeting the C&C server itself.

```
15:06:40.258475 IP 10.0.5.105.37057 > 8.8.8.8.53: 62163+ A? zhegege.3322.org. (34)
15:06:40.726971 IP 8.8.8.8.53 > 10.0.5.105.37057: 62163 1/0/0 A 122.224.51.128 (50)
15:06:40.730609 IP 10.0.5.105.53631 > 122.224.51.128.2866: Flags [S], seq 2922816321,
15:06:41.129553 IP 122.224.51.128.2866 > 10.0.5.105.53631: Flags [S.], seq 215022169,
15:06:41.129615 IP 10.0.5.105.53631 > 122.224.51.128.2866: Flags [.], ack 1, win 46, c
15:06:41.140430 IP 10.0.5.105.53631 > 122.224.51.128.2866: Flags [P.], seq 1:29, ack 1
15:06:41.302777 IP 122.224.51.128.2866 > 10.0.5.105.53625: Flags [.], seq 0:1, ack 29,
15:06:41.302840 IP 10.0.5.105.53625 > 122.224.51.128.2866: Flags [.], ack 1, win 46, c
15:06:42.984124 IP 10.0.5.105.53631 > 122.224.51.128.2866: Flags [P.], seq 1:29, ack 1
15:06:43.593549 IP 122.224.51.128.2866 > 10.0.5.105.53631: Flags [.], ack 29, win 6550
15:06:43.593600 IP 10.0.5.105.53631 > 122.224.51.128.2866: Flags [P.], seq 29:301, ack
15:06:43.991663 IP 122.224.51.128.2866 > 10.0.5.105.53631: Flags [P.], seq 1:29, ack 3
15:06:43.991733 IP 10.0.5.105.53631 > 122.224.51.128.2866: Flags [.], ack 29, win 46,
15:06:44.993659 IP 10.0.5.104.45570 > 122.224.51.128.2866: UDP, length 44
15:06:44.993837 IP 10.0.5.106.15876 > 122.224.51.128.2866: UDP, length 44
15:06:44.995441 IP 10.0.5.102.2820 > 122.224.51.128.2866: UDP, length 44
15:06:44.995575 IP 10.0.5.108.23814 > 122.224.51.128.2866: UDP, length 44
15:06:44.997439 IP 10.0.5.98.48390 > 122.224.51.128.2866: UDP, length 44
```

In another screenshot, one can see the malware generating traffic from additional random IPs, again targeting the C&C server:

```
08:05:32.759311 IP 10.0.37.104.8178 > 122.224.51.128.2866: UDP, length 44
08:05:33.591168 IP 9.255.197.106.22817 > 122.224.51.128.2866: UDP, length 44
08:05:33.591204 IP 10.0.69.104.8162 > 122.224.51.128.2866: UDP, length 44
08:05:34.366684 IP 9.255.133.106.22849 > 122.224.51.128.2866: UDP, length 44
08:05:34.366721 IP 10.0.133.104.8130 > 122.224.51.128.2866: UDP, length 44
08:05:35.404877 IP 9.255.5.106.22913 > 122.224.51.128.2866: UDP, length 44
08:05:35.404920 IP 10.1.5.104.8066 > 122.224.51.128.2866: UDP, length 44
08:05:35.983990 IP 9.254.5.106.22785 > 122.224.51.128.2866: UDP, length 44
08:05:35.984024 IP 10.2.5.104.7938 > 122.224.51.128.2866: UDP, length 44
08:05:36.632236 IP 9.252.5.106.22785 > 122.224.51.128.2866: UDP, length 44
08:05:36.632289 IP 10.4.5.104.7938 > 122.224.51.128.2866: UDP, length 44
08:05:42.383378 IP 9.248.5.106.22785 > 122.224.51.128.2866: UDP, length 44
08:05:42.383470 IP 10.8.5.104.7938 > 122.224.51.128.2866: UDP, length 44
08:05:42.385849 IP 9.240.5.106.22785 > 122.224.51.128.2866: UDP, length 44
08:05:42.385901 IP 10.16.5.104.7938 > 122.224.51.128.2866: UDP, length 44
08:05:42.386870 IP 9.224.5.106.22785 > 122.224.51.128.2866: UDP, length 44
08:05:42.386921 IP 10.32.5.104.7938 > 122.224.51.128.2866: UDP, length 44
08:05:42.387878 IP 9.192.5.106.22785 > 122.224.51.128.2866: UDP, length 44
08:05:42.387926 IP 10.64.5.104.7938 > 122.224.51.128.2866: UDP, length 44
```

This behavior shows how the malware manages to "self-discover" whether it is capable of sending spoofed IP traffic, and if so, from which subnets. Later, the malware uses this information to perform a large DDoS attack from thousands of spoofed IP addresses, which are very similar to the 50,000 Unique IPs seen in the attack investigated previously.

Reverse Engineering of the Malware Sample

This malware belongs to the **Linux XOR/DDoS** family, which has been active throughout the past year and has been previously analyzed by Avast[2], Fireeye[3], and others[4]. During the writing of this paper, additional infections have been reported [5] [6].

It is a sophisticated bot, written in C or C++, and remotely controlled and operated by a central server. It has the following characteristics:

- Uses an XOR key for encryption/obfuscation
- Uses a persistence mechanism, which makes it difficult for an average user to remove it from an infected Linux machine
- Periodically connects to a list of C&C (Command and Control) servers and receives commands such as:
    - Start attack
    - Stop attack
    - Download a file and execute, which is a remote upgrade mechanism
- Supports multiple attack methods – SYN, ACK, DNS
- Supports HTTP GET/POST commands
- Supports sending DNS Queries
- Supports IP spoofing

[2] https://blog.avast.com/2015/01/06/linux-ddos-trojan-hiding-itself-with-an-embedded-rootkit/
[3] https://www.fireeye.com/blog/threat-research/2015/02/anatomy_of_a_brutef.html
[4] http://www.kernelmode.info/forum/viewtopic.php?t=3509
[5] https://isc.sans.edu/diary/XOR+DDOS+Mitigation+and+Analysis/19827
[6] http://blog.malwaremustdie.org/2015/06/mmd-0033-2015-linuxxorddos-infection_23.html

A sample of the binary can be found on VirusTotal with the following SHA256 hash:
498f3348df1b6804db2692e4f937d7cbefd71916e83a9421347077fb1cdafa95

The malware was first submitted to VirusTotal on May 21, 2015, approximately 1 month prior to the writing of this paper. As of May 26, the antivirus detection ratio is only 18/57.

virustotal

SHA256: 498f3348df1b6804db2692e4f937d7cbefd71916e83a9421347077fb1cdafa95
File name: mtjrnexwtq
Detection ratio: 18 / 57
Analysis date: 2015-05-26 08:40:34 UTC (4 weeks ago)

0 0

Analysis | File detail | Additional information | Comments 0 | Votes

Antivirus	Result	Update
AVG	Linux/DDoS.XOR	20150526
AhnLab-V3	Linux/Xorddos.610320	20150525
Avast	ELF:Xorddos-K [Trj]	20150526
CAT-QuickHeal	ELF629	20150523
ClamAV	Linux.Trojan.Xorddos	20150525

Malware Static Analysis

The binary is a Linux ELF 32 bit executable, built for the 32 bit x86 Intel platform.

```
root@box:~# ls -la mtjrnexwtq
-rw------- 1 root root 610320 Jun 21 06:59 mtjrnexwtq
root@box:~# file mtjrnexwtq
mtjrnexwtq: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked,
 for GNU/Linux 2.6.9, not stripped
```

It was compiled and statically linked for an old Linux kernel version – 2.6.9.

It is also an unstripped binary, meaning it still contains Debug Symbols with visible source code file names:

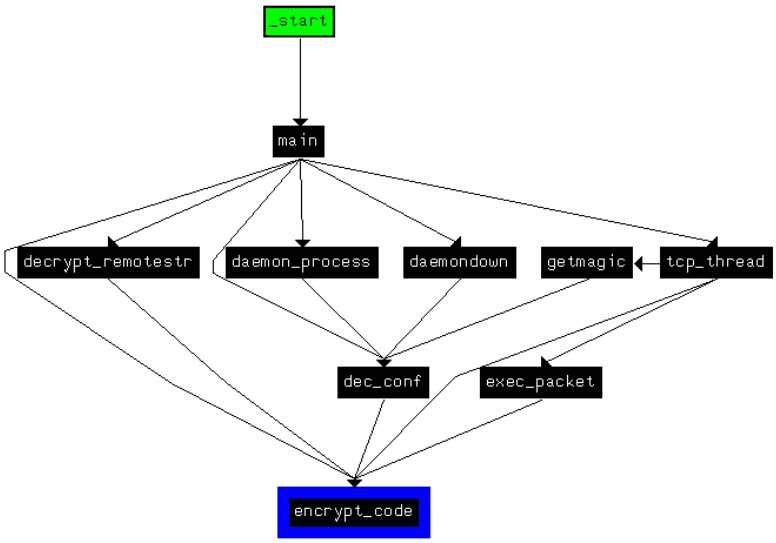
```
root@box:~# readelf -s mtjrnexwtq | grep '\.c$'
    26: 00000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
    36: 00000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
    41: 00000000     0 FILE    LOCAL  DEFAULT  ABS autorun.c
    42: 00000000     0 FILE    LOCAL  DEFAULT  ABS crc32.c
    43: 00000000     0 FILE    LOCAL  DEFAULT  ABS encrypt.c
    44: 00000000     0 FILE    LOCAL  DEFAULT  ABS execpacket.c
    45: 00000000     0 FILE    LOCAL  DEFAULT  ABS buildnet.c
    46: 00000000     0 FILE    LOCAL  DEFAULT  ABS http.c
    47: 00000000     0 FILE    LOCAL  DEFAULT  ABS kill.c
    48: 00000000     0 FILE    LOCAL  DEFAULT  ABS main.c
    49: 00000000     0 FILE    LOCAL  DEFAULT  ABS proc.c
    50: 00000000     0 FILE    LOCAL  DEFAULT  ABS socket.c
    51: 00000000     0 FILE    LOCAL  DEFAULT  ABS tcp.c
    52: 00000000     0 FILE    LOCAL  DEFAULT  ABS thread.c
    53: 00000000     0 FILE    LOCAL  DEFAULT  ABS findip.c
    54: 00000000     0 FILE    LOCAL  DEFAULT  ABS dns.c
```

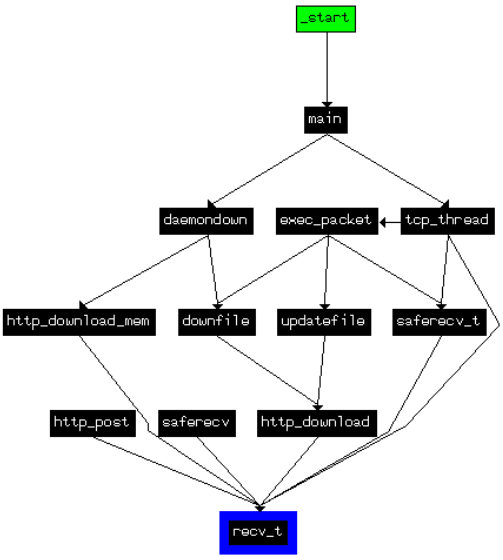
Function names are in the code section, and variable names are in the data section:

1	copyfile	encrypt_code	http_post	recv_
2	writefile	exec_packet	rand_filename	set_sock_keep_alive
3	randmd5	checksum	http_download	tcp_connect
4	doublefork	build_udphdr	randstr	make_network_ip
5	LinuxExec_Argv2	build_tcphdr	daemondown	getmagic
6	LinuxExec_Argv	build_iphdr	updatefile	tcp_thread
7	LinuxExec	tcp_checksum	downfile	add_remotelist
8	DelService	udp_checksum	getpath	decrypt_remotestr
9	AddService	in_checksum	daemon_process	clear_task
10	readnfile	fix_findip	main	threadwork
11	readfile	fix_dns	getdevname	get_socket_ip
12	CreateDir	fix_ack	GetLanSpeed	add_task
13	get_file_form_pid	fix_syn	ReadWord	getrandom
14	DelService_form_pid	build_ack	GetCpuInfo	init_array
15	get_self	build_syn	GetMemStat	send_udp
16	get_self_path	build_findip	Get_AllIP	recvfromTimeOut
17	dec_conf	fix_domain	send_t	recvAnalyse
18	init_crc_table	build_dns	safesend	randomid
19	CalcCrc32	abstract_save_name	send_	pkgDNS
20	CalcHeaderCrc	abstract_file_name	recv_t	resolve
21	CalcFindIpCrc	abstract_url	saferecv_t	
22	CalcFileCrc	http_download_mem	saferecv	

Name	Address
DNS_PORT	080CBE18
DNS_ADDR2	080CBE08
DNS_ADDR	080CBDF8
http_data_e	080CBDF4
http_data_m	080CBDF0
additional	080CBDE4
options	080CBDDC
SIZE_PSEUDO_HDR	080CBDD8
SIZE_DNS_T	080CBDD4
SIZE_DNS_H	080CBDD0
SIZE_TCP_H	080CBDCC
SIZE_UDP_H	080CBDC8
SIZE_IP_H	080CBDC4
xorkeys	080CBDB4
corn	080CBD20
aUptime	080CBCF8
aPwd	080CBCE4
aWhoami	080CBCD0
aWho	080CBCBC
aId	080CBCA8
aGnomeTerminal	080CBC94
aRouteN	080CBC80
aIfconfig	080CBC6C

Below is a diagram of the execution flow of important functions in this malware:





The binary contains the following strings:

```
103.25.9.228
103.25.9.229
8.8.8.8
/etc/cron.hourly/udev.sh
/etc/init.d/%s
/etc/ld.so.cache
/etc/localtime
/etc/rc%d.d/S90%s
/etc/rc.d/rc%d.d/S90%s
/etc/resolv.conf
/etc/suid-debug
/etc/cron.hourly/udev.sh
/lib/libgcc4.4.so
/proc/%d/exe
/proc/self/exe
glibc 2.5
GCC: (GNU) 4.1.2 20080704 (Red Hat 4.1.2-46)
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin
GET %s HTTP/1.1
POST %s HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; TencentTraveler ;
.NET CLR 1.1.4322)
cp /lib/libgcc4.so /lib/libgcc4.4.so
sed -i '/\/etc\/cron.hourly\/udev.sh/d' /etc/crontab
echo '*/3 * * * * root /etc/cron.hourly/udev.sh' >> /etc/crontab
/proc/cpuinfo
/proc/meminfo
/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin
```

Using a simple XOR obfuscation, it contains additional “hidden” strings. These include:

1. 66.102.253.30:2866|zhegege.3322.org:2866|122.224.48.63:2866
2. http://info1.3000uc.com/b/u.php?id=public
3. /var/run/udev.pid
4. /var/run
5. /lib/libgcc4.so

The first string is a list of C&C servers to which the malware will try to connect every few seconds.

The second string is a URL which will be fetched after the bot is running for 72 hours. The complete URL will look as follows:

http://info1.3000uc.com/b/u.php?id=public&ip=66.102.253.30:2866|zhegege.3322.org:2866|122.224.48.63:2866

The malware will attempt to download a file from that URL, and if successful – it will execute it.

Another group of obfuscated strings is a list of "fake" process names which the malware is using during runtime for hiding its presence. When a user will use the "ps aux" command, for example, he will not see the process name of the malware, but rather one of the following fake strings:

```
cat resolv.conf
sh
bash
su
ps -ef
ls
ls -la
top
netstat -an
netstat -antop
grep "A"
sleep 1
cd /etc
echo "find"
ifconfig eth0
ifconfig
```

Binary Replication

During the malware runtime, the process will create multiple threads. One of these threads is the daemon thread, which is responsible for ensuring both the process will be executed automatically after reboot and that it is difficult to terminate the process. This thread will keep copying the original file to a new, randomly generated 10-character file name. If the original process is killed, then the new one will automatically take over execution.

Additionally, the newly created binary contains an extra 11 bytes which are appended to the end of the file. View the below image showing the differences between the original file (mtjrnexwtq) and the one that was executed automatically (uubtuadsrx) after we manually killed the first one:

```
mtjrnexwtq
0009 4FE0: 5F 6E 6C 5F 6C 6F 61 64  5F 6C 6F 63 61 6C 65 5F  _nl_load _locale_
0009 4FF0: 66 72 6F 6D 5F 61 72 63  68 69 76 65 00 77 63 74  from_arc hive.wct
0009 5000: 72 61 6E 73 00 67 73 76  78 6B 61 6D 6C 6A 6A 00  rans.gsv xkamljj.
uubtuadsrx
0009 4FE0: 5F 6E 6C 5F 6C 6F 61 64  5F 6C 6F 63 61 6C 65 5F  _nl_load _locale_
0009 4FF0: 66 72 6F 6D 5F 61 72 63  68 69 76 65 00 77 63 74  from_arc hive.wct
0009 5000: 72 61 6E 73 00 67 73 76  78 6B 61 6D 6C 6A 6A 00  rans.gsv xkamljj.
0009 5010: 72 6E 79 70 76 71 68 66  72 65 00                 rnypvqhf re.
0009 5020:
```

(First one is 610320 - bytes, the second one is 610330 bytes)

Process Name Hiding

As mentioned in the previous section, the malware will attempt to hide its process name by changing to a fake process name during runtime. In the screenshot below one can see the original malware process is highlighted in **red** with a fake process name “id”, while the original name is actually “gupccjvpff”.

```
root      3197  0.0  0.0   1664   424 tty4     Ss+  Jun11   0:00 /sbin/mingetty tty4
root      3200  0.0  0.0   1664   424 tty5     Ss+  Jun11   0:00 /sbin/mingetty tty5
root      3201  0.0  0.0   1664   444 tty6     Ss+  Jun11   0:00 /sbin/mingetty tty6
root      3652  0.0  0.0   1664   428 tty1     Ss+  Jun11   0:00 /sbin/mingetty tty1
root      4566  9.9  0.0  41988   576 ?        Ssl  Jun20 532:05 id
root     10659  0.0  0.2   9896  2792 ?        Ss   19:43   0:00 sshd: root@pts/0
root     10661  0.0  0.1   4664  1472 pts/0    Ss   19:43   0:00 -bash
root     17571  0.0  0.0   1368   920 ?        Ss   20:20   0:00 bash
root     17574  0.0  0.0   1368   920 ?        Ss   20:20   0:00 grep "A"
root     17575  0.0  0.0   1368   920 ?        Ss   20:20   0:00 sleep 1
root     17578  0.0  0.0   1368   916 ?        Ss   20:20   0:00 pwd
root     17579  0.0  0.0   1368   920 ?        Ss   20:20   0:00 sh
root     17581  0.0  0.0   4260   944 pts/0    R+   20:20   0:00 ps aux
[root@zoo-centos5 ~]# ls -la /proc/4566/exe
lrwxrwxrwx 1 root root 0 Jun 20 03:12 /proc/4566/exe -> /usr/bin/gupccjvpff
```

Notice the “532:05” text in the CPU column which indicates this process has been keeping the CPU very busy attacking targets. One can see a group of five processes highlighted in **blue** with fake names. These are processes through which the malware continuously creates and replicates itself to new binaries with new names, before killing and deleting the previous one.

If the malware is run using *strace*, it creates a copy of itself by copying **/lib/libgcc4.so** to a new random file **/usr/bin/jyyddrbtp :**

```
open("/lib/libgcc4.so", O_RDONLY)        = 4
lseek(4, 0, SEEK_SET)                    = 0
open("/usr/bin/jyyddrbtbp", O_WRONLY|O_CREAT, 0777) = 5
lseek(5, 0, SEEK_SET)                    = 0
read(4, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\2\0\3\0\1\0\0\0
write(5, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\2\0\3\0\1\0\0
```

After the copy is completed, it will append 11 bytes to the new binary (highlighted in **red**).

Then it will create five child processes, and delete the original binary using the unlink command (highlighted in **blue**).

```
open("/usr/bin/jyyddrbtbp", O_WRONLY)   = 4
lseek(4, 0, SEEK_END)                   = 610320
gettimeofday({1435082021, 112089}, NULL) = 0
write(4, "caepkyraas\0", 11)            = 11
close(4)                                = 0
gettimeofday({1435082021, 112246}, NULL) = 0
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x8c23888) = 23601
waitpid(23601, 0xbfa97610, 0)           = -1 ECHILD (No child processes)
open("/usr/bin/jyyddrbtbp", O_WRONLY)   = -1 ETXTBSY (Text file busy)
gettimeofday({1435082021, 113663}, NULL) = 0
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x8c23888) = 23603
waitpid(23603, 0xbfa97610, 0)           = -1 ECHILD (No child processes)
open("/usr/bin/jyyddrbtbp", O_WRONLY)   = -1 ETXTBSY (Text file busy)
gettimeofday({1435082021, 114992}, NULL) = 0
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x8c23888) = 23605
waitpid(23605, 0xbfa97610, 0)           = -1 ECHILD (No child processes)
open("/usr/bin/jyyddrbtbp", O_WRONLY)   = -1 ETXTBSY (Text file busy)
gettimeofday({1435082021, 116578}, NULL) = 0
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x8c23888) = 23607
waitpid(23607, 0xbfa97610, 0)           = -1 ECHILD (No child processes)
open("/usr/bin/jyyddrbtbp", O_WRONLY)   = -1 ETXTBSY (Text file busy)
gettimeofday({1435082021, 117583}, NULL) = 0
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x8c23888) = 23609
waitpid(23609, 0xbfa97610, 0)           = -1 ECHILD (No child processes)
rt_sigprocmask(SIG_BLOCK, [CHLD], □, 8) = 0
rt_sigaction(SIGCHLD, NULL, {0x1, [CHLD], SA_RESTART}, 8) = 0
nanosleep({1, 0}, {1, 0})               = 0
rt_sigprocmask(SIG_SETMASK, □, NULL, 8) = 0
unlink("/usr/bin/jyyddrbtbp")           = 0
unlink("/bin/jyyddrbtbp")               = -1 ENOENT (No such file or directory)
unlink("/tmp/jyyddrbtbp")               = -1 ENOENT (No such file or directory)
```

Summary

Malware authors appear to be relentlessly writing new malware for fun and profit.

Security vulnerabilities in servers and applications are discovered daily. Unfortunately, malware authors exploit those vulnerabilities for the purpose of gaining access to remote systems, infecting them, and then assembling an army of "zombies" to launch DDoS attacks, send spam, and more.

Even worse, malware is rapidly infecting servers, personal routers, and other network equipment for which the owner either did not change the default password, or simply used a trivial one. Weak passwords make it easy to launch brute force attacks.

Defending against DDoS attacks can be quite difficult. Unfortunately, there are still many ISPs and hosting companies that do not prevent IP Spoofing by filtering their ingress traffic and fail to implement (BCP38 / RFC 2827). Because IP blacklisting is not possible, it's even more difficult to defend against DDoS attacks.

Over the last year there has seen a significant increase in the adoption and use of Linux-based malware to infect SOHO routers[7], PBX systems, and various Linux servers. This malware presents many significant security threats with unique characteristics. Notably, a known vulnerability and target for exploitation are the many powerful Linux-based systems which do not enforce password policies. They are often left overlooked, and instead, become a sitting target for DDoS malware infection.

[7] https://www.incapsula.com/blog/ddos-botnet-soho-router.html

Recommended Steps

DDoS protection mechanisms should prevent both volumetric and sophisticated attacks at the same time**.** It is imperative for business critical applications to be able to accurately mitigate attacking traffic while allowing uninterrupted, legitimate traffic to flow through the network.

The following best practices are recommended:

1. Enforce a strong password security policy for servers, workstations and network appliances.
2. Ensure systems are up-to-date and patched with all critical security fixes.
3. Use Antivirus for both Windows AND Linux systems.
4. Check router ACLs and firewall rules, and implement BCP38.
5. Perform a security audit at least once or twice a year.

Whether currently experiencing a DDoS attack, or concerned about DDoS protection mechanisms and attack mitigation solutions for safeguarding business operations, please contact Radware's Emergency Response Team (ERT) for information and assistance.

For additional research reports and DDoS prevention techniques, please visit Radware's DDoS Security Research Center.