Improve DDoS Botnet Tracking with Honeypots

Ya Liu
Network Security Research Lab, Qihoo 360
Agenda

• About DDoS botnet tracking
• DDoS botnet families and their PGAs (Packet Generation Algorithm)
• Backscatter collection and analysis
• PGA analysis
• Experiments
DDoS botnet tracking

- It’s aimed to learn botnet assisted DDoS attacks
  - **4w:** **w**ho is being attacked by **w**hat botnet families under **w**hich C2 controllers with **w**hat set of attacking parameters (e.g., attack type)

```
2016/11/23 15:16:07 mirl securityupdates.us  6.188.232.103  23   ddoa tcp_ack_flood, target=109.163.224.34, mask_bits=32, atk_time=60, payload_size=1
2016/11/23 15:16:08 mirl timeserver.host 188.209.49.106  23  ddoa tcp_ack_flood, target=109.163.224.34, mask_bits=32, atk_time=60, payload_size=1
2016/11/23 15:50:27 mirl cnc.routersinhis.com 93.158.212.81  23   ddoa udp_flood, target=217.68.245.94, mask_bits=32, port=80, atk_time=100, port=80
2016/11/23 15:50:27 mirl ftp.xenonbooter.xyz 93.158.212.81  23  ddoa udp_flood, target=217.68.245.94, mask_bits=32, port=80, atk_time=100, port=80
```

- time
- family
- C2 domain
- C2 IP/port
- command type
- attack target & parameters
Stats on our tracking

• Our tracking started in 2014
• 30+ botnet families
• 6,000+ successfully tracked botnets
• 800+ million received attack commands
• 250K+ checked attack targets
• Our data has been shared many times with colleagues outside of our company
How to evaluate it?

• For evaluation purpose, we need to know:
  – what *family-unknown* botnets are active in the wild?
  – what *family-known* C2 controllers are outside of our tracking list?

• Therefore we need information about the real attacks, and a method to connect them to the used botnet families
DDoS backscatter

- It’s generated due to the use of spoofed source IPs in attacking packets
  - e.g., TCP SYN-ACKs acknowledged to spoofed SYNs

- It’s known as a cause of dark space traffic in parallel with scanning and network misconfigurations

- Solutions to detect & monitor DDoS attacks based on backscatters have been proposed in the past years
## Darknet? Or honeypot?

<table>
<thead>
<tr>
<th></th>
<th>Pros.</th>
<th>Cons.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Darknet</strong></td>
<td>• Collect a large number of packets destined to a block of unused addresses</td>
<td>• Non-trivial deployment</td>
</tr>
<tr>
<td><strong>Honeypot</strong></td>
<td>• Cost effective</td>
<td>• Less packets collected</td>
</tr>
<tr>
<td></td>
<td>• Easy to deploy</td>
<td></td>
</tr>
</tbody>
</table>
Our scheme

• Full traffic captures are taken on our dozens of low-interaction honeypots

• A special mechanism is designed to separate the **wanted** traffic from the **unwanted**
  – **wanted**: traffic generated by honeypot applications
  – **unwanted**: scans, backscatters, etc.
In modern botnets, attacking packets are usually generated by the bots according to some specific algorithm which we call PGA.

PGA attributes:
- It’s attack type specific
- It’s usually family specific
- Fixed patterns usually exist in the generated packets

Botnet families can be identified by PGA signatures.
MIRAI’s PGA for *stomp* attack

[*] copied from attack_tcp_stomp() of attack_tcp.c

```c
00472:    iph->check = 0;
00473:    iph->check = checksum_generic((uint16_t *)iph, sizeof (struct iphdr));
00475:    tcph->seq = htons(stomp_data[i].seq++);
00476:    tcph->ack_seq = htons(stomp_data[i].ack_seq);
00477:    tcph->check = 0;
```

fixed "0x0000" can be found in `tcph->seq` and `tcph->ack_seq`
MIRAI’s PGA for `gre_eth` attack

[*] copied from `attack_gre_eth()` in `attack_gre.c`

```c
if (ip_ident == 0xffff)
{
    iph->id = rand_next() & 0xffff;
    greiph->id = ~(iph->id - 1000);
}
if (sport == 0xffff)
    udph->source = rand_next() & 0xffff;
if (dport == 0xffff)
    udph->dest = rand_next();
if (!gcip)
    greiph->daddr = rand_next(),
else
    greiph->daddr = iph->daddr;
```

`greiph->id` is bound to `iph->id`

`greiph->daddr` is bound to `iph->addr`
From packets to bot families

- Buggy implementation and design flaws lead to PGA signatures which can be characterized by their packets.

- PGA signatures can also be concluded by reverse engineering the bot sample.

- It’s possible to correlate an DDoS attack to the used botnet families by PGA signature matching.
The architecture overview

Honeypot Traffic → DDoS Backscatter → Spike Detection → PGA Analysis → Signature Match → Results

Labeled Signature

Traffic Spike Detection

Backscatter

DDoS

Honeypot

PGA Analysis

Results
A TCP-SYN spike example
The reflected spike
Our spike detection scheme

- Backscatters are sub-grouped based on:
  - \{packet type, source IP, source port\}
  - or \{packet type, queried domain\} in case DNS responses

<table>
<thead>
<tr>
<th>Policy name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKET_TIME_INTERVAL</td>
<td>If 2 packets’ interval is less than this value, they are grouped to the same spike.</td>
</tr>
<tr>
<td>LEAST_NUMBER_OF_PACKETS</td>
<td>The least number of packets a valid spike MUST have.</td>
</tr>
<tr>
<td>LEAST_NUMBER_OF_HONEYPOTS</td>
<td>The least number of honeypots a valid spike MUST hit.</td>
</tr>
</tbody>
</table>
3 supported backscatters

- TCP SYN-ACK packets for detecting SYN flood
- DNS response packets for detecting query flood
- ICMP unreachable messages
  - ICMP type=3, code=3
  - The original attacking packets could be restored
# Restoring attacking pkt fields

<table>
<thead>
<tr>
<th>Backscatter type</th>
<th>Restored attacking packet fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP SYN-ACK</td>
<td>- source/destination IP/port</td>
</tr>
<tr>
<td></td>
<td>- initial sequence number</td>
</tr>
<tr>
<td>DNS response</td>
<td>- source/destination IP</td>
</tr>
<tr>
<td></td>
<td>- source port</td>
</tr>
<tr>
<td></td>
<td>- transaction ID (tid for short)</td>
</tr>
<tr>
<td></td>
<td>- queried domain</td>
</tr>
<tr>
<td>ICMP unreachable message</td>
<td>- sip/dip/sport/dport/ISN for SYN</td>
</tr>
<tr>
<td></td>
<td>- sip/dip/sport/tid for DNS query</td>
</tr>
</tbody>
</table>
Packet feature vector & matrix

• They are defined to find the fixed patterns in attacking packets
  – And to calculate the spike feature vector

• The vector is constructed by restored field bytes
  – A 16-dimension vector for SYN packet
    • \{sip, dip, ISN, sport, dport\}
  – A 12-dimension vector for DNS query packet
    • \{sip, dip, sport, transaction-id\}
A real SYN packet feature matrix

fixed patterns

21 packets

SIP  DIP  ISN  sport  dport
Spike feature vector

• It’s used to find the bound relations among attacking packet fields, and to do spike clustering

• It’s obtained by calculating the Shannon Entropy of each column of packet feature matrix
A real DNS spike feature vector

21 packets

sip.o3 = tid.low

sip.o4 = tid.high + 1

{e1=1.52, e2=1.52, e3=1.52, e4=1.87, e5=4.16, e6=4.65, e7=4.60, e8=4.60, e9=1.52, e10=2.81, e11=1.87, e12=1.52}
Spike clustering & PGA profiling

• Spike clustering is to find the similar spikes that are probably generated by the same family

• PGA is profiled in 2 ways:
  – fixed patterns are detected by checking element of 0.0
  – bounds are detected by checking the same elements

• The profiling result is used for signature matching
About the bound relation

• One field is derived from another one:

\[ field_n = f(field_m) \]

• While the simplest bound relation is simple byte sharing, the real situation is complicated

• Our approach only detects the bound relations, with the exact relations left for manual analysis
About the evaluation

• If a detected spike is not successfully correlated, it means there are family unknown botnets in the wild.

• If a spike is successfully correlated, we just check our tracking list to see whether the attack has been tracked or not.
Experiments

- 2,333 SYN-ACK spikes and 1,835 DNS spikes are checked from August, 2015 to October, 2016
- 4 large PGA clusters are found

<table>
<thead>
<tr>
<th>Cluster</th>
<th>PGA signatures</th>
<th>Spikes</th>
<th>Botnet family</th>
</tr>
</thead>
<tbody>
<tr>
<td>sa_cls1</td>
<td>{(p_9=p_{13}), (p_{10}=p_{14})}</td>
<td>1318</td>
<td>XOR.DDoS</td>
</tr>
<tr>
<td>sa_cls2</td>
<td>{(p_{13}=0x00), (p_{14}=0x01)}</td>
<td>131</td>
<td>unknown</td>
</tr>
<tr>
<td>dns_cls1</td>
<td>{(p_4=p_{11}+1), (p_3=p_{12})}</td>
<td>626</td>
<td>unknown</td>
</tr>
<tr>
<td>dns_cls2</td>
<td>{(p_3=p_9), (p_4=p_{10})}</td>
<td>21</td>
<td>unknown</td>
</tr>
</tbody>
</table>
About dns_cls1

• It can be connected to a family-unknown attack tool which supports DNS random subdomain attack

• It shares the same subdomain pattern with Elknot/BillGates, but has different PGA signature

• It’s still active, and mainly used to attack China online-game domains
Conclusions

• A backscatter collection scheme with honeypots

• A spike based attack detection scheme from DDoS backscatters

• A PGA analysis approach based on recovered attacking packet fields
Q&A

liuya@360.cn