Uncovering Network Tarpits with Degreaser, presentation

Alt, Lance
Monterey, California: Naval Postgraduate School.

http://hdl.handle.net/10945/41916
Uncovering Network Tarpits with Degreaser

Lance Alt and Robert Beverly

Naval Postgraduate School
Center for Measurement and Analysis of Network Data
Computer Science Dept.

May 22, 2014

CAIDA Topology/BGP Meeting
1. Background

2. Degreaser

3. Experiments

4. Next...
Cyber-Deception and Network Measurement

- Internet measurements reliant on (fragile) inferences
- Available tools are Tricks and hacks – Internet was not intended to be measured
- Inherent difficulty means researchers are happy to get any results, and don’t question them

Question:
- Should measurement research assumptions include a more adversarial model?
Background

Cyber-Deception and Network Measurement

- Internet measurements reliant on (fragile) inferences
- Available tools are Tricks and hacks – Internet was not intended to be measured
- Inherent difficulty means researchers are happy to get *any* results, and don’t question them

Question:

- Should measurement research assumptions include a more adversarial model?
Active Cyber Defense

- Typical assumption for active measurements: a host either responds (truthfully) or does not.
- For instance, a non-response:
  - Firewall or other blocking
  - Protocol/service/measurement trick not supported
- However, a third choice is gaining momentum: deception
  - Provide a false response to influence adversary’s behavior
  - Canonical example: honeypots
- In our world: fake networks, fake hosts
How prevalent are deceptive networks/hosts on the Internet?
How do Internet topology scans treat these “fake” networks?
(Or: how much junk/noise is creeping into our global measurements)
Can “fake” networks/hosts be identified?
IS THIS REAL?? ⇒

L. Alt & R. Beverly (NPS)

Degreaser CAIDA Topo 2014
The Target: Tarpits

Network Tarpits

- This talk focuses on one form of deceptive network behavior: tarpits
- Originally conceived as a defensive mechanism
- Idea: attempt to slow (or stop) various forms of network scanning (e.g. for open services)
- Two well-known applications:
  - LaBrea
  - Linux Netfilter (via TARPIT plugin)
- General Idea:
  - A single machine pretends to be all unused hosts on a subnetwork
  - Answers for all requests to those fake hosts
  - By setting TCP window to zero...
  - And never letting go ...
- Let’s look at LaBrea in detail
LaBrea Layer-2 Capture

- Two modes of operation:
  - ARP-timeout – actively captures unused addresses
  - Hard capture – only listens on specific addresses

- LaBrea promiscuously listens for ARP requests

- If no answer to (multiple) requests, LaBrea assumes IP not in use...

- And claims to be that IP (always with same MAC)

- Example: 10.1.10.102 is a real host attempting to connect to (non-existent) host 10.1.10.210:

  06:20:44.848758 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
  06:20:45.953257 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
  06:20:46.962535 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
  06:20:47.970023 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
  06:20:47.970130 ARP, Reply 10.1.10.210 is-at 00:00:0f:ff:ff:ff, length 28
LaBrea

LaBrea Layer-2 Capture

- Two modes of operation:
  - ARP-timeout – actively captures unused addresses
  - Hard capture – only listens on specific addresses
- LaBrea promiscuously listens for ARP requests
- If no answer to (multiple) requests, LaBrea assumes IP not in use...
- And claims to be that IP (always with same MAC)
- Example: 10.1.10.102 is a real host attempting to connect to (non-existent) host 10.1.10.210:

06:20:44.848758 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
06:20:45.953257 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
06:20:46.962535 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
06:20:47.970023 ARP, Request who-has 10.1.10.210 tell 10.1.10.102, length 46
06:20:47.970130 ARP, Reply 10.1.10.210 is-at 00:00:0f:ff:ff:ff, length 28
LaBrea

LaBrea ICMP Response

- After layer-2 capture, LaBrea responds to TCP and ICMP
- Example ping from 10.1.10.102 to 10.1.10.205:

  06:20:31.501417 ARP, Request who-has 10.1.10.205 tell 10.1.10.102, length 46
  06:20:33.501954 ARP, Request who-has 10.1.10.205 tell 10.1.10.102, length 46
  06:20:34.503146 ARP, Request who-has 10.1.10.205 tell 10.1.10.102, length 46
  06:20:34.503257 ARP, Reply 10.1.10.205 is-at 00:00:0f:ff:ff:ff, length 28
  06:20:34.504452 IP 10.1.10.102 > 10.1.10.205: ICMP echo request, id 61467, seq 3, length 64
  06:20:34.504536 IP 10.1.10.205 > 10.1.10.102: ICMP echo reply, id 61467, seq 3, length 64
LaBrea ICMP Response

- After layer-2 capture, LaBrea responds to TCP and ICMP
- Example ping from 10.1.10.102 to 10.1.10.205:

06:20:31:501417 ARP, Request who-has 10.1.10.205 tell 10.1.10.102, length 46
06:20:33:501954 ARP, Request who-has 10.1.10.205 tell 10.1.10.102, length 46
06:20:34:503146 ARP, Request who-has 10.1.10.205 tell 10.1.10.102, length 46
06:20:34:503257 ARP, Reply 10.1.10.205 is-at 00:00:0f:ff:ff:ff, length 28
06:20:34:504452 IP 10.1.10.102 > 10.1.10.205: ICMP echo request, id 61467, seq 3, length 64
06:20:34:504536 IP 10.1.10.205 > 10.1.10.102: ICMP echo reply, id 61467, seq 3, length 64
LaBrea also responds to TCP connection attempts to any TCP port.
TCP SYN/ACK has an advertised window of 10 (or 3), and no TCP options.
Never ACKs or ACKs with zero window (persistent mode).

Example HTTP from 10.1.10.102 to 10.1.10.210:

```
06:20:47.971276 IP 10.1.10.102.51161 > 10.1.10.210.http: Flags [S], seq 3536100821, win 65535,
  options [mss 1460,nop,wscale 4,nop,nop,TS val 1194569089 ecr 0,sackOK,eol], length 0
06:20:47.971475 IP 10.1.10.210.http > 10.1.10.102.51161: Flags [S.], seq 1457023515, ack 3536100822,
  win 10, length 0
```
LaBrea also responds to TCP connection attempts to any TCP port.
TCP SYN/ACK has an advertised window of 10 (or 3), and no TCP options.
Never ACKs or ACKs with zero window (persistent mode).
Example HTTP from 10.1.10.102 to 10.1.10.210:

```
06:20:47.971276 IP 10.1.10.102.51161 > 10.1.10.210.http: Flags [S], seq 3536100821, win 65535, options [mss 1460,nop,wscale 4,nop,nop,TS val 1194569089 ecr 0,sackOK,eol], length 0
06:20:47.971475 IP 10.1.10.210.http > 10.1.10.102.51161: Flags [S.], seq 1457023515, ack 3536100822, win 10, length 0
```
1. Background
2. Degreaser
3. Experiments
4. Next...
Experiments

- In the lab (where things worked great)
- Set up LaBrea tarpit on /29 within Comcast
What Doesn’t Work: Response Time

- Does LaBrea respond faster or slower than a real host?
  - LaBrea is much slower to respond in ARP-timeout mode
  - Unreliable due to ARP caching

- PlanetLab scan to /24 containing LaBrea
  - 60 Planet Lab nodes
  - Red dots are LaBrea responses
  - Blue dots are real host responses

- No distinguishable difference when not running in ARP-timeout mode
What Doesn’t Work: Port Scanning

- What about looking for hosts listening on all TCP ports?
  - Search space too big!
  - \(2^{32} \times 2^{16}\) scans

- We could search for hosts with more than XX listening ports...
  - This still requires multiple scans per host

However it’s easier than that!
What Doesn’t Work: Port Scanning

- What about looking for hosts listening on all TCP ports?
  - Search space too big!
  - $2^{32} \times 2^{16}$ scans

- We could search for hosts with more than XX listening ports...
  - This still requires multiple scans per host

However its easier than that!
Degreaser

Discriminating Characteristics

What Does Work

- We can easily detect tarpit hosts using only:
  - TCP Window Size
  - TCP Options

Key Advantages

- Only one TCP connection per host
- Requires sending only 3 packets per host
- Not susceptible to network noise (like response time measurements)
To understand how tarpit traffic characteristics differ from “normal” traffic, we analyze two traffic traces:

- **Equinix SanJose (CAIDA)**: 60s, 31M packets, 24G bytes, 5.4M flows
- **Campus (NPS)**: 3600s, 48M packets, 34G bytes, 1.2M flows

---

### Discriminating Characteristics

- **Ground Truth**
  
  - To understand how tarpit traffic characteristics differ from “normal” traffic.
  - We analyze two traffic traces.

<table>
<thead>
<tr>
<th>Trace</th>
<th>Duration</th>
<th>Packets</th>
<th>Bytes</th>
<th>Flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equinix SanJose (CAIDA)</td>
<td>60s</td>
<td>31M</td>
<td>24G</td>
<td>5.4M</td>
</tr>
<tr>
<td>Campus (NPS)</td>
<td>3600s</td>
<td>48M</td>
<td>34G</td>
<td>1.2M</td>
</tr>
</tbody>
</table>
TCP Window Size

- **Observed Window Sizes**
  - 155,490 TCP connections
  - 407 (0.2%) zero windows
  - Everything else greater than 200 bytes

- **LaBrea Window Size**
  - Configurable
  - Default: 10 or 3

- **Netfilter Window Size**
  - Not Configurable
  - Default: 5
Discriminating Characteristics

TCP Options

- Equinix and NPS traces showed a very high percentage of connections that used TCP options.

- LaBrea and Netfilter never reply with TCP options.

<table>
<thead>
<tr>
<th></th>
<th>Equinix Trace</th>
<th>NPS Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7.8% No options</td>
<td>0% No options</td>
</tr>
<tr>
<td></td>
<td>92.2% At least one option</td>
<td>100% At least one option</td>
</tr>
</tbody>
</table>
New tool: **Degreaser**

- Network scanner that can detect tarpitting hosts
- GPL Licensed (will be available soon)
- Multi-threaded, C++
- libcrafter for packet manipulation

<table>
<thead>
<tr>
<th>Host</th>
<th>Response</th>
<th>WinSize</th>
<th>TCPFlags</th>
<th>TCPOptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.240.192.189</td>
<td>No response.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62.97.115.180</td>
<td>Labrea Host.</td>
<td>3</td>
<td>SA</td>
<td></td>
</tr>
<tr>
<td>110.29.8.230</td>
<td>Rejecting.</td>
<td>0</td>
<td>AR</td>
<td></td>
</tr>
<tr>
<td>59.28.4.215</td>
<td>Real Host.</td>
<td>14480</td>
<td>SA</td>
<td>MWST</td>
</tr>
<tr>
<td>186.98.169.75</td>
<td>No response.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>144.93.146.200</td>
<td>No response.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>168.62.42.151</td>
<td>Real Host.</td>
<td>8192</td>
<td>SA</td>
<td>MWST</td>
</tr>
</tbody>
</table>
Detection in the Wild

Degreaser Internals

- Sends TCP SYN to host and waits for responding SYN/ACK
  - Includes MSS, TSVAL, SACK and WSCALE options
- Window size. Is it abnormally small?
  - Small size is good indication of a tarpit
- Did any TCP options get returned?
  - Existence rules out tarpit (except MSS, possibly)

But Wait!

- A real host might legitimately have a small window size and not use options.
Degreaser Internals

- Sends TCP SYN to host and waits for responding SYN/ACK
  - Includes MSS, TSVAL, SACK and WSCALE options
- Window size. Is it abnormally small?
  - Small size is good indication of a tarpit
- Did any TCP options get returned?
  - Existence rules out tarpit (except MSS, possibly)

But Wait!

- A real host might legitimately have a small window size and not use options.
Detection in the Wild

Send a Data Packet

Send a data packet of size one less than the window size

- A real host would send an ACK, but neither LaBrea nor Netfilter do!
- The data packet can also distinguish between LaBrea and Netfilter:
  - LaBrea: Won’t respond with ACK unless payload > window size
  - Netfilter: Immediately sets window to zero.
Outline

1 Background

2 Degreaser

3 Experiments

4 Next...
Experiments

Probing

Scanning

- Does anyone actually admit to using this stuff?
  - BizSystems (3 IP addresses)
- What about on the larger Internet?

scans.io

- Began our experiments by looking at scans.io
- Idea: *degrease* networks in order of their occupancy
- Didn’t work:
  - High-occupancy networks were CDNs, hosting centers
  - scans.io looking for application-layer connects, not just TCP establishment
Experiments

Probing

Does anyone actually admit to using this stuff?
- BizSystems (3 IP addresses)
- What about on the larger Internet?

scans.io

Began our experiments by looking at scans.io
- Idea: *degrease* networks in order of their occupancy
- Didn’t work:
  - High-occupancy networks were CDNs, hosting centers
  - scans.io looking for application-layer connects, not just TCP establishment
Probing

Scanning

- Instead...
- Scanned over 4 million IP addresses from NPS over a 4 week period, starting in April, 2014
  - Scanned slowly not to raise suspicion from IT dept.
  - Used cryptographic permutation to “randomize” the scan
  - We have scanned at least one host from 25% of the /24 subnets
- Found 18 tarpitting hosts directly via degreaser
Results

Scanning Results

- Of the 18 hosts:
  - 10 were LaBrea (non-persist mode)
  - 6 were LaBrea (persist mode)
  - 16 were address blocks assigned to universities
  - 2 were commercial address blocks

- Completed an exhaustive search on subnets containing these hosts

- Largest: /20
- Over 20,700 IP addresses showing tarpit-like behavior.
- Across 7 autonomous systems and 3 countries.
Some example from census data. The indicated blocks of green cells — high occupancy subnets? Nope. All fake.
A view from Ark

- Impacts Ark traceroute data too...
- How many randomly chosen destinations respond to traceroute?
- Survey of Ark traces in April, 2014

A typical subnetwork (1/6 respond):

130.207.24.0/23:
  - 130.207.24.20 Status: False
  - 130.207.25.62 Status: True
  - 130.207.25.98 Status: False
  - 130.207.24.149 Status: False
  - 130.207.24.156 Status: False
  - 130.207.25.161 Status: False

A LaBrea subnet (16/16 respond):

XXX.YYY.252.0/22:
  - XXX.YYY.252.89 Status: True
  - XXX.YYY.253.62 Status: True
  - XXX.YYY.254.164 Status: True
  - XXX.YYY.255.86 Status: True
  - XXX.YYY.252.133 Status: True
  - XXX.YYY.253.6 Status: True
  - XXX.YYY.254.148 Status: True
  - XXX.YYY.255.6 Status: True
  - XXX.YYY.252.98 Status: True
  - XXX.YYY.253.136 Status: True
  - XXX.YYY.254.76 Status: True
  - XXX.YYY.255.232 Status: True
  - XXX.YYY.252.203 Status: True
  - XXX.YYY.253.127 Status: True
  - XXX.YYY.254.26 Status: True
  - XXX.YYY.255.80 Status: True
Outline

1 Background
2 Degreaser
3 Experiments
4 Next...
Conclusions

Take Aways

- Cyber deception is real
- Open question as to whether its use is increasing
- But, general caution to measurement researchers to be more cognizant of deception
- What we’ve discovered is in the noise relative to the entire Internet, but still represents large networks
- And significant that we were able to discover these needles in a haystack
Future Work

- Integrate into *nmap*?
- Understand subnets that return zero window (particularly 166/8)
- Build a better tarpit?
- Combine with topology deception?
- Measure tarpits (and general deception behavior) over time.
Summary

- Developed methodology and tool, *degreaser*, to detect tarpits
- Found strong evidence of active tarpits in the Internet
- Observations on deception within Internet measurement work

Thanks!

Questions?