Uncovering Network Tarpits with Degreaser

Lance Alt*, Robert Beverly*, Alberto Dainotti†

*Naval Postgraduate School
Center for Measurement and Analysis of Network Data
Computer Science Dept.
†UCSD/CAIDA

December 11, 2014

Annual Computer Security Applications Conference 2014

Background

Network Deception

- A popular form of network defense is *cyber deception*
- Idea: confuse and influence adversary, collect attack data
- E.g., honeypots, sinkholes, **tarpits**

Our Work

Can we detect tarpits?

Motivation

- An adversary able to recognize deception (tarpit) will avoid it
- Understanding weaknesses of existing tarpits helps improve them (better deception)
- Understand the extent to which network measurement tools and surveys are influenced by tarpits in the wild

Background

Network Deception

- A popular form of network defense is *cyber deception*
- Idea: confuse and influence adversary, collect attack data
- E.g., honeypots, sinkholes, **tarpits**

Our Work

Can we detect tarpits?

Motivation

- An adversary able to recognize deception (tarpit) will avoid it
- Understanding weaknesses of existing tarpits helps improve them (better deception)
- Understand the extent to which network measurement tools and surveys are influenced by tarpits in the wild

Background

Network Deception

- A popular form of network defense is *cyber deception*
- Idea: confuse and influence adversary, collect attack data
- E.g., honeypots, sinkholes, **tarpits**

Our Work

Can we detect tarpits?

Motivation

- An adversary able to recognize deception (tarpit) will avoid it
- Understanding weaknesses of existing tarpits helps improve them (better deception)
- Understand the extent to which network measurement tools and surveys are influenced by tarpits in the wild

The Target: Tarpits

Network Tarpits

- Attempts to slow (or stop) various forms of network scanning
- General Idea:
  - A single machine pretends to be all unused hosts on a subnetwork
  - Answers for all requests to those fake hosts
  - Holds the TCP connection by setting TCP window to zero...
  - And never letting go ...
- Two well-known applications:
  - LaBrea
  - Linux Netfilter (via TARPIT plugin)

LaBrea in Detail

LaBrea Layer-2 Capture

- Two modes of operation:
  - ARP-timeout – actively captures unused addresses (default)
  - 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 in Detail

LaBrea Layer-2 Capture

- Two modes of operation:
  - ARP-timeout – actively captures unused addresses (default)
  - 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

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

LaBrea TCP Response

- 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
- Two modes of operation:
  - Persistent: always respond with 0 window
  - Non-Persistent: ignore all future traffic
- 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

LaBrea TCP Response

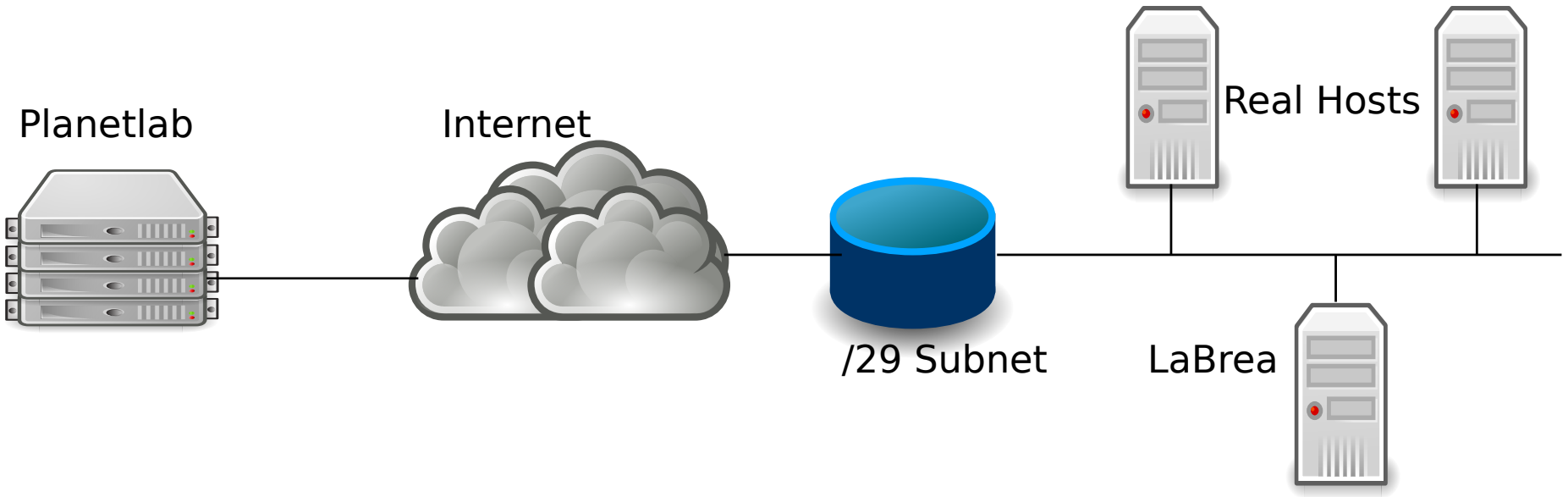
- LaBrea also responds to TCP connection attempts to **any** TCP port
- TCP SYN/ACK has an advertised window of 10 (or 3), and no TCP options
- Two modes of operation:
  - Persistent: always respond with 0 window
  - Non-Persistent: ignore all future traffic
- Example HTTP from `10.1.10.102` to `10.1.10.210`:

```
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
```

Discriminating Characteristics

Experiments

- In the lab (where things worked great)
- Set up LaBrea tarpit on /29 within Comcast (where we learned a lot)



Discriminating Characteristics

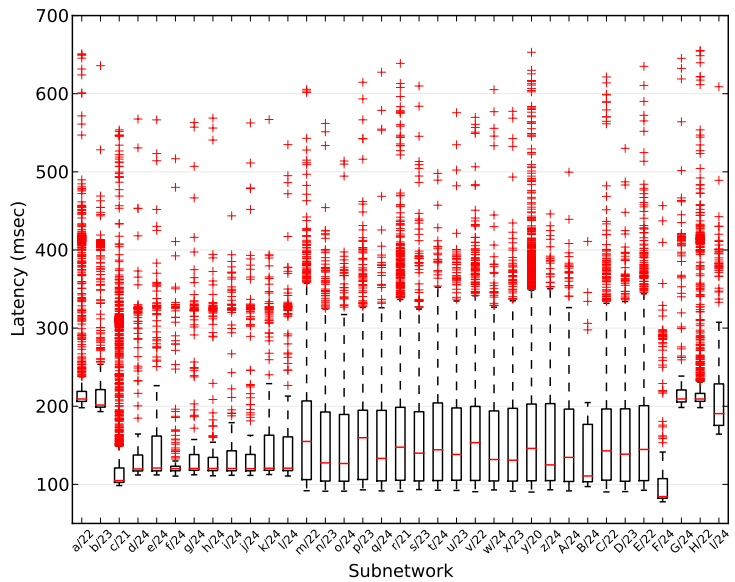
What Doesn't Work: Subnet Occupancy

- Can we find tarpit by locating fully occupied subnetworks?
- No. High-occupancy subnets are often content providers (CDNs, hosting services)
- However, we examine the relationship between Project Sonar (`scans.io`) counts of half-responding hosts and our inferred fake subnets.

Discriminating Characteristics

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
- No distinguishable difference when not running in ARP-timeout mode



Discriminating Characteristics

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 X listening ports...
  - This still requires multiple scans per host
  - And won't detect single-port tarpits (e.g. iptables)

However it's easier than that!

Discriminating Characteristics

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 X listening ports...
  - This still requires multiple scans per host
  - And won't detect single-port tarpits (e.g. iptables)

However it's easier than that!

Discriminating Characteristics

What Does Work

- We can efficiently detect tarpit IPs using:
  - TCP Window Size
  - TCP Options
- Key Advantages
  - Only one TCP connection per target
  - Requires sending only 2-6 packets per target
  - Not susceptible to network noise (e.g. response latency)

Discriminating Characteristics

How do tarpit traffic characteristics differ from "normal" traffic?

TCP Options

- Analyze two packet captures to get a feel for "normal" traffic

Trace	Length	Pkts	Flows	Min Non-Zero Window Size	No TCP Opts
Equinix	60s	31M	5.4M	246	0.5%
Campus	3660s	48M	1.2M	2,920	0.0%

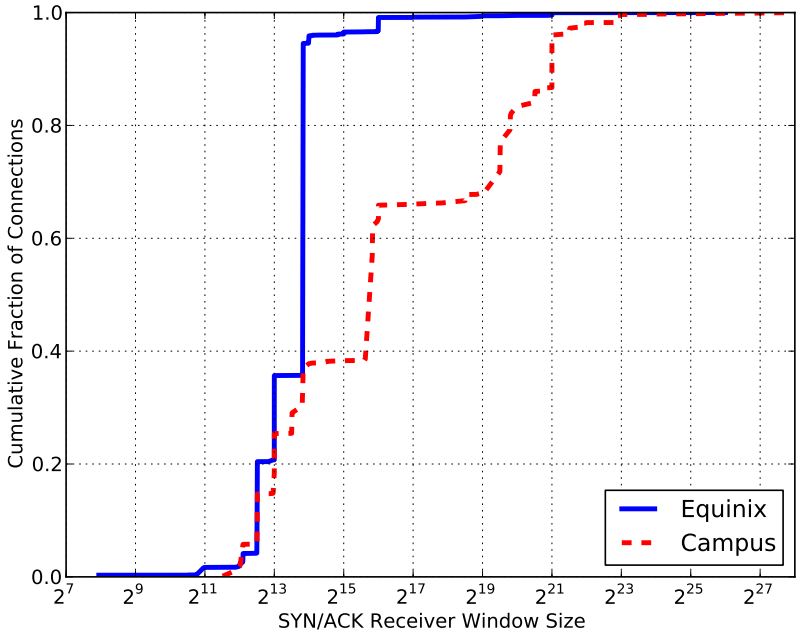
- Normal traffic **almost always** contains TCP options
- LaBrea and Netfilter **never** reply with TCP options

Discriminating Characteristics

How do tarpit traffic characteristics differ from "normal" traffic?

TCP Window Size

- Observed Window Sizes
  - 407 (0.2%) zero windows
  - Remainder $\geq$200 bytes
- LaBrea Window Size
  - Configurable
  - Default: 10 or 3
- Netfilter Window Size
  - Not Configurable
  - Default: 5



Introducing Degreaser

New tool: *Degreaser*

- Network scanner that can detect tarpitting hosts
- Multi-threaded, C++
- Open Source (currently on github)
- Can detect:
  - LaBrea Persistent (LaBrea-P)
  - LaBrea Non-persistent (LaBrea-NP)
  - Netfilter TARPIT (iptables-T)
  - Netfilter DELUDE (iptables-D)
    - Respond with a SYN/ACK, RST otherwise

Introducing Degreaser

Degreaser: Network scanner to detect tarpitting

```
IP:    311552/496690176   Scanned IPs: 311552          Excluded IPs: 0
Real Hosts: 0             Rejecting Hosts: 5062        Errors: 15225
Tarpits: 125335           LaBrea: 123739               iptables(tarpit): 1596
                                                       iptables(delude): 9414
  1% [==>                                                                  ]
IP Address            Response Time    Window Size  TCP Flags   TCP Options  Scan Result
199.133.85.176                95885              0                           Error in TCP packet
136.227.165.15               165304              0       SA          M       LaBrea
148.228.33.42                     0              0                           No response
209.129.242.227                   0              0                           No response
188.118.162.36               222828              0                           Unreachable
208.184.85.68                     0              0                           No response
108.59.196.198               106382              0       SA          M       LaBrea
203.106.97.168                    0              0                           No response
210.240.212.93               181553              0       SA          M       LaBrea
196.74.235.92                     0              0                           No response
197.61.159.19                     0              0                           No response
195.232.132.215                   0              0                           No response
202.38.248.236                    0              0                           No response
```

Degreaser

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

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 Algorithm

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.

Distinguishing between LaBrea-P and LaBrea-NP:

- Send a zero-window probe
  - LaBrea-P: Responds with zero-win ACK
  - LaBrea-NP: No response

Detection Algorithm

Special Case: Zero Window

- Can't send a data packet, so we send a FIN
- Response?
  - Yes → Real Host
  - No → Other
- Lots of oddities observed with "other" hosts!
  - Blacklisting
  - Double SYN/ACKs
- Could be LaBrea with non-default configuration, or something completely different

Detection in the Wild

Googling

- Does anyone actually admit to using this stuff?
  - We found only one company (3 tarpitting IP addresses)
- What about on the larger Internet?

Scanning

Instead...

- Scanned over 20 million IP addresses
- Used cryptographic permutation to randomize the scan: avoid triggering IDS/anomaly detectors
- Scanned at least one host from 100% of the /24 subnets in Internet
- Found **1,451** tarpitting IPs directly via *degreaser*

Detection in the Wild

Googling

- Does anyone actually admit to using this stuff?
  - We found only one company (3 tarpitting IP addresses)
- What about on the larger Internet?

Scanning

Instead...

- Scanned over 20 million IP addresses
- Used cryptographic permutation to randomize the scan: avoid triggering IDS/anomaly detectors
- Scanned at least one host from 100% of the /24 subnets in Internet
- Found **1,451** tarpitting IPs directly via *degreaser*

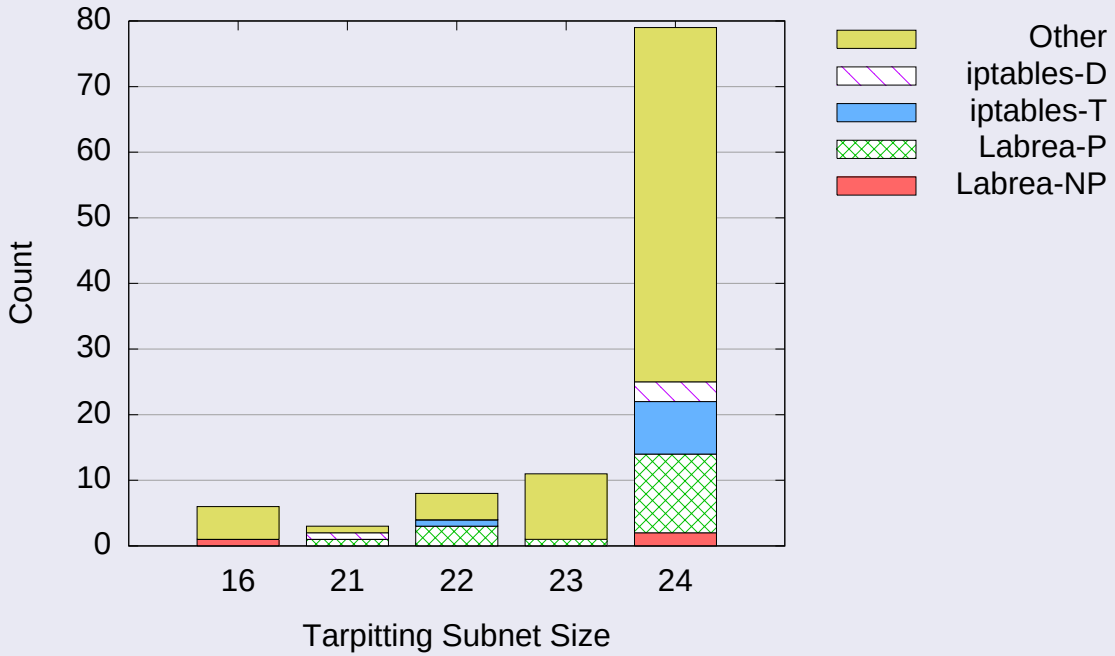
Results

Scanning Results

For each of the 1,451 tarpitting IPs:

- Completed an exhaustive search on subnets containing these hosts
- Next, expand search to adjacent subnets

- Largest Subnet: /16
- Over **215,000** IP addresses showing tarpit-like behavior.
- 77 autonomous systems
- 29 countries

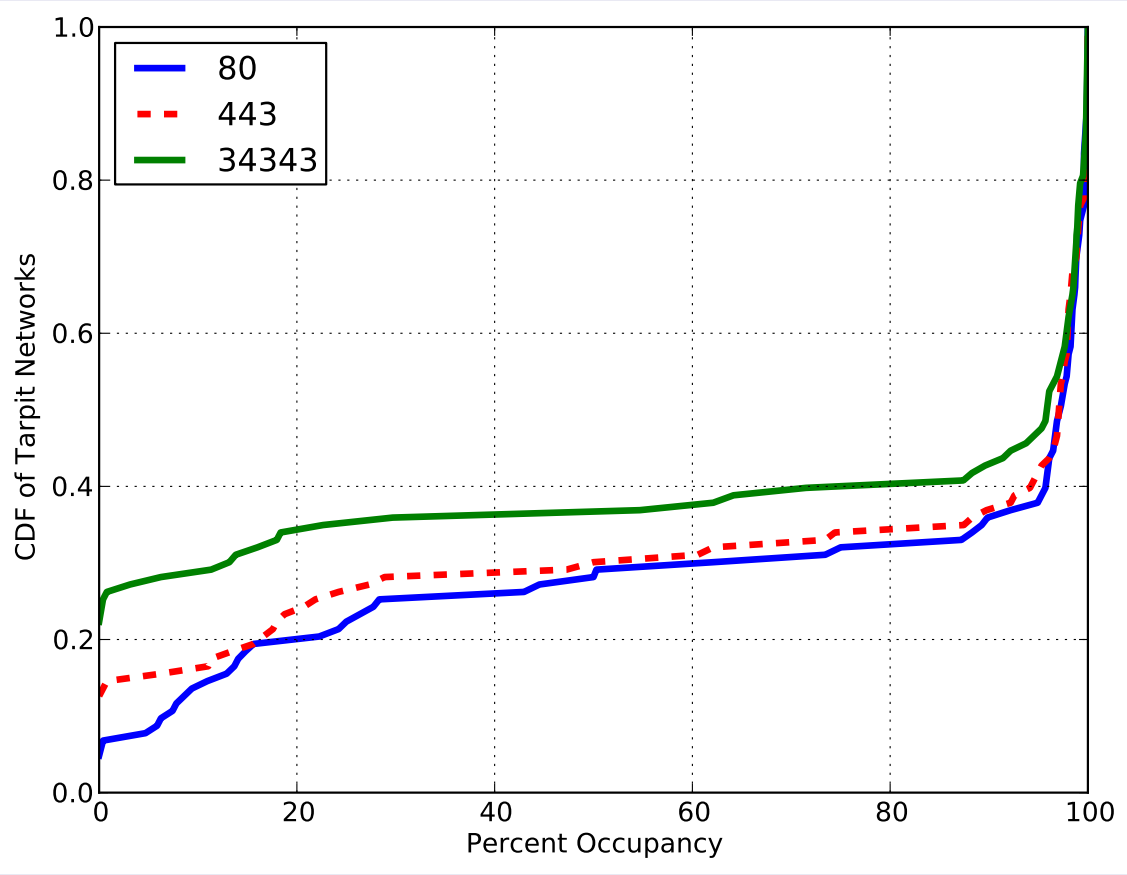


Results

Port Density

- Scanned two well-know and one random port (34343) on each host
- We would expect very few hosts to be listening on the random port

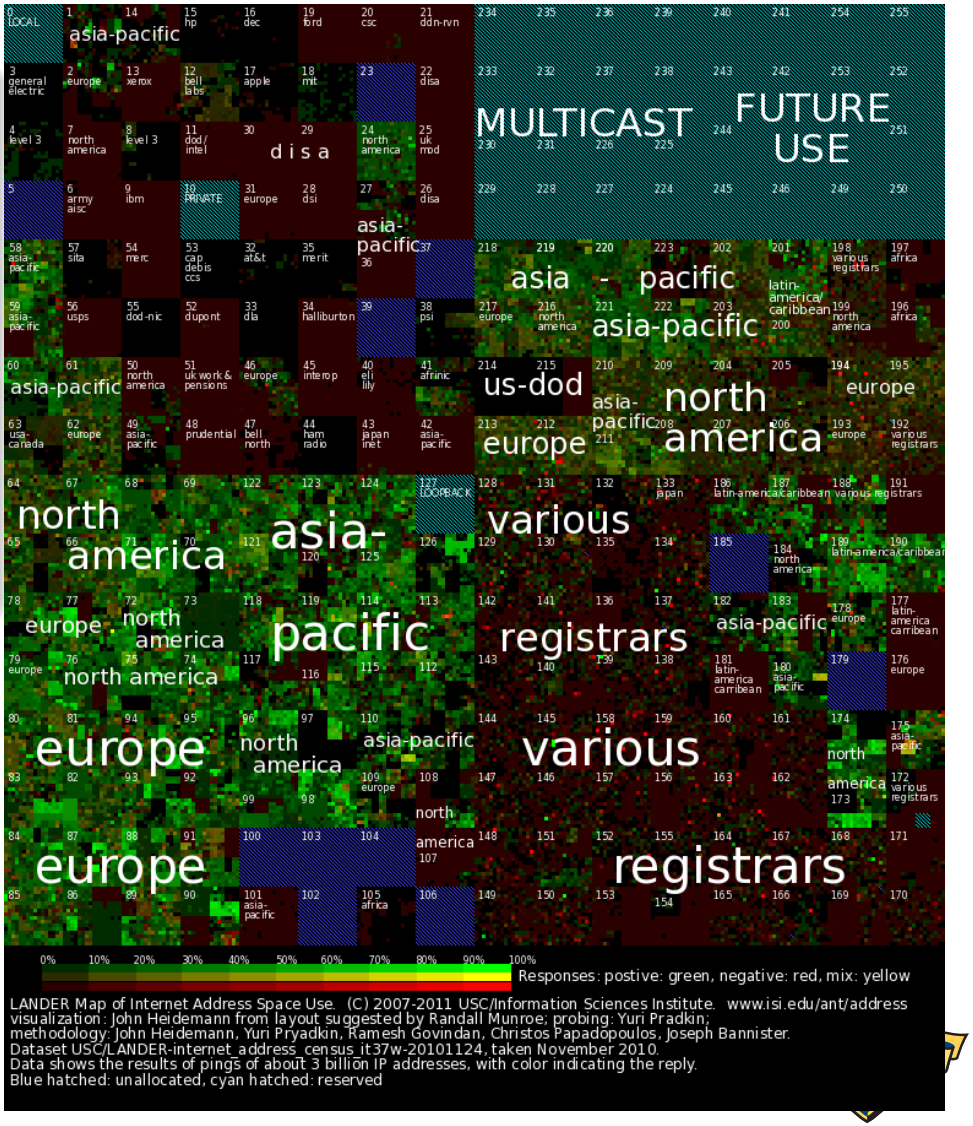
- Notice the random port has a density close to the well-know ports
- Indicates a high percentage of hosts listening on all ports
- This is expected behavior for deceptive hosts



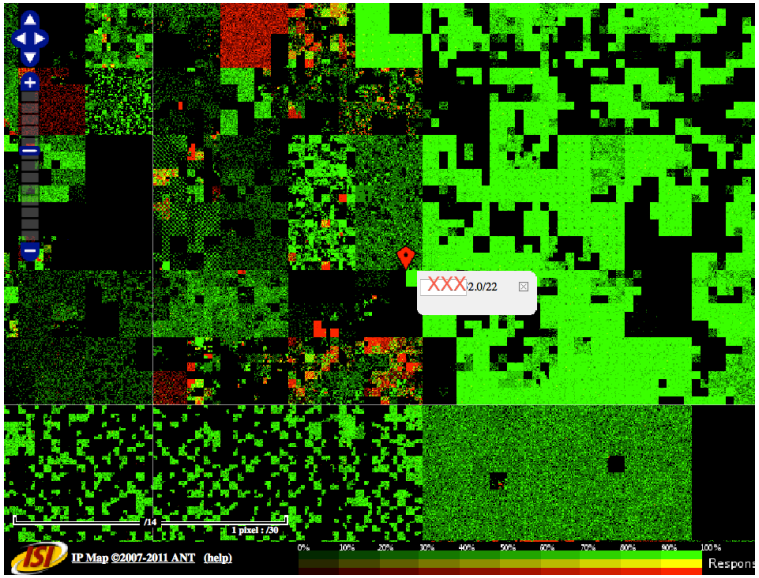
Internet Census

Internet-wide Tarpit Influence

- How prevalent is tarpit deception on the Internet?
- How much junk/noise is creeping into global measurements and surveys?
- IS THIS REAL?? $\Rightarrow$



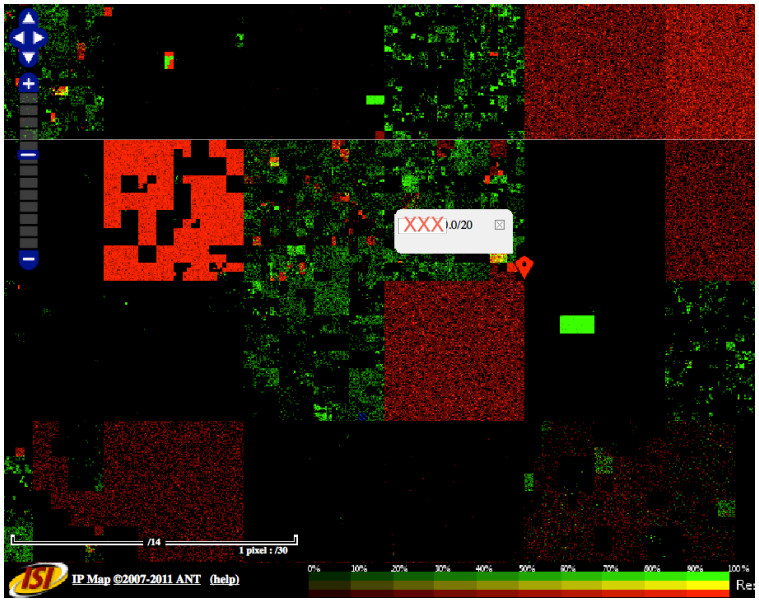
Results

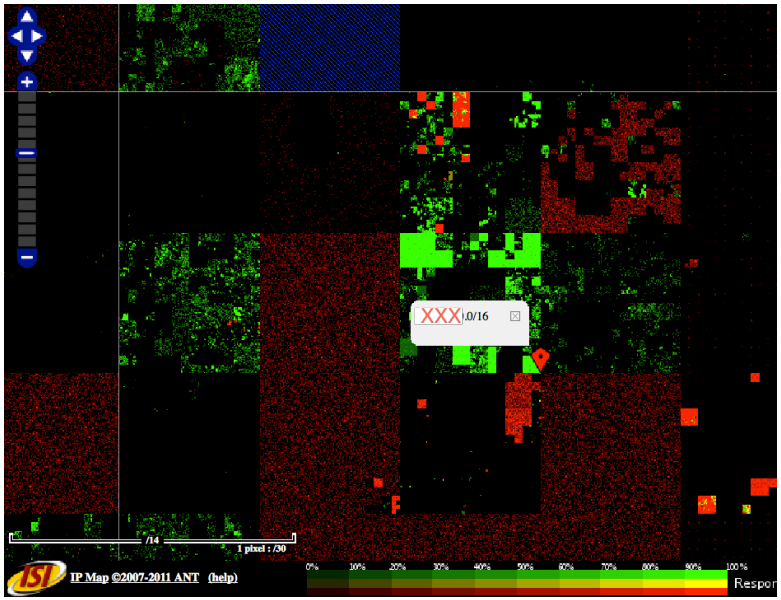


Examples from the ISI Internet Census Data:

Are the indicated blocks of green cells – high occupancy subnets?

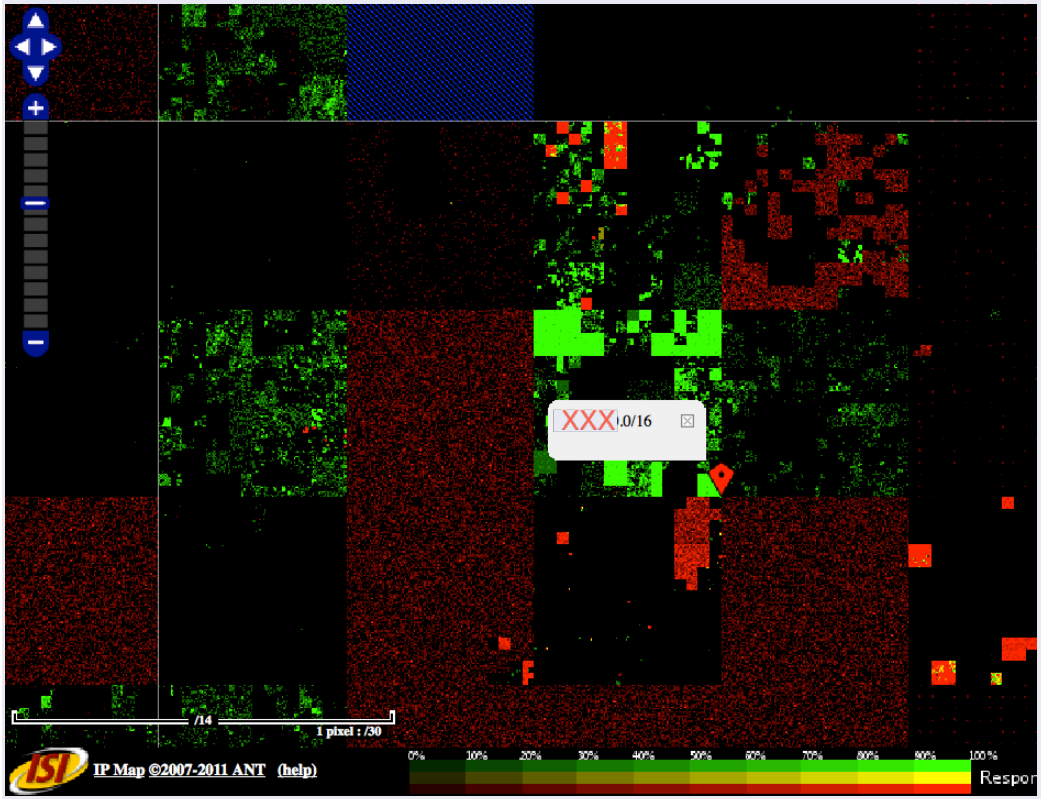
Nope. All fake.





Results

ISI Internet Census Data



For example, this /16:

- 58 (of 256 possible) /24 subnetworks are fake (23%)

Overall:

- 2 of 6 /16s with tarpits we found are fully occupied!
- These chunks represent 2^{17} fake addresses alone!

Conclusions

Take Aways

1. Cyber deception is real
   - What we discovered in the noise relative to the entire Internet – but still represents large networks.
   - Significant that we were able to discover these needles in a haystack
   - We obtain (limited) ground truth to verify our detection methodology
   - And, small blocks of tarpits have significant effect on scanning speed
2. Cyber deception is detectable
   - Existing tarpits are easy to detect
   - Detection techniques could be used by adversaries to evade tarpits
   - Open question as to whether use of deception is increasing
3. Cyber deception has real effect on the accuracy of Internet measurement scans

Building a Better Tarpit

Improvement 1: TCP Options

- Easy! Just copy or slightly modify the options sent by the remote host.
- Requires no state

Improvement 2: TCP Retransmissions

- Use TCP retransmissions to draw out the connection
- Requires tarpit to maintain per-connection state

Improvement 3: Window Obfuscation

- Advertise a large initial window
- Accept some data, but not all the client wants to send
- Eventually reduce window to zero

Future Directions

Future Directions

- Understand "other" IPs that return zero window
- Measure tarpits (and general deception behavior) over time.
- Build a better tarpit
- Build a tarpit-immune TCP stack

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?

`http://www.cmand.org/degreaser/`

Work supported in part by Department of Homeland Security (DHS) Science and Technology Directorate