Fast Internet-wide scanning and its security applications

J. Alex Halderman
University of Michigan

Based on joint work

ZMap: Fast Internet-Wide Scanning and its Security Applications
Zakir Durumeric, Eric Wustrow, and J. Alex Halderman
22nd Usenix Security Symposium (Sec '13), August 2013

Analysis of the HTTPS Certificate Ecosystem
Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman
13th Internet Measurement Conference (IMC '13), October 2013

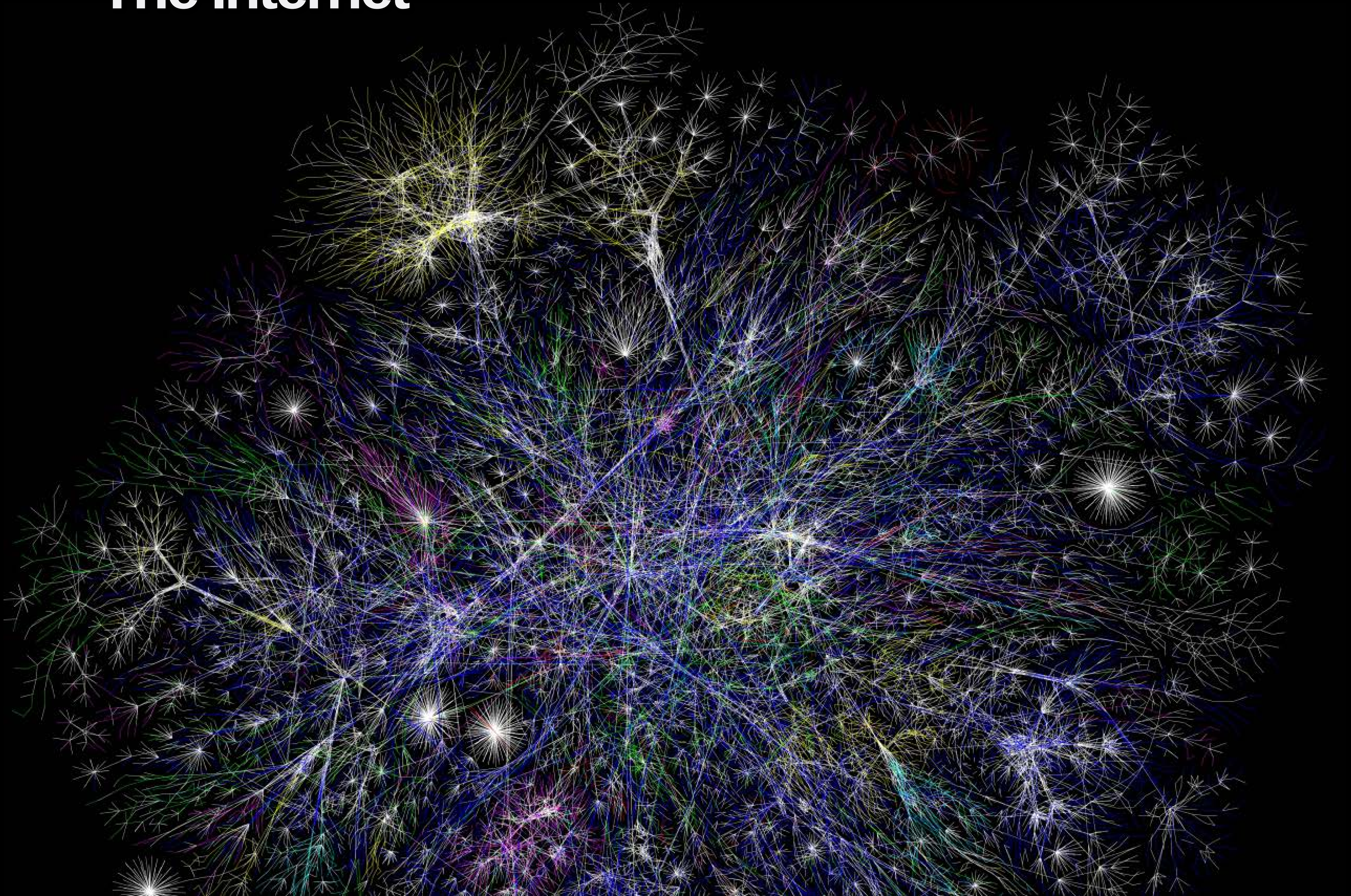
Elliptic Curve Cryptography in Practice
Joppe W. Bos, J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow
To appear. *18th Intl. Conf. on Financial Cryptography and Data Security* (FC '14), March 2014

Illuminating the Security Issues Surrounding Lights-Out Server Management
Anthony Bonkoski, Russ Bielawski, and J. Alex Halderman
7th Usenix Workshop on Offensive Technologies (WOOT '13), August 2013

CAge: Taming Certificate Authorities by Inferring Restricted Scopes
James Kasten, Eric Wustrow, and J. Alex Halderman
17th Intl. Conf. on Financial Cryptography and Data Security (FC '13), April 2013

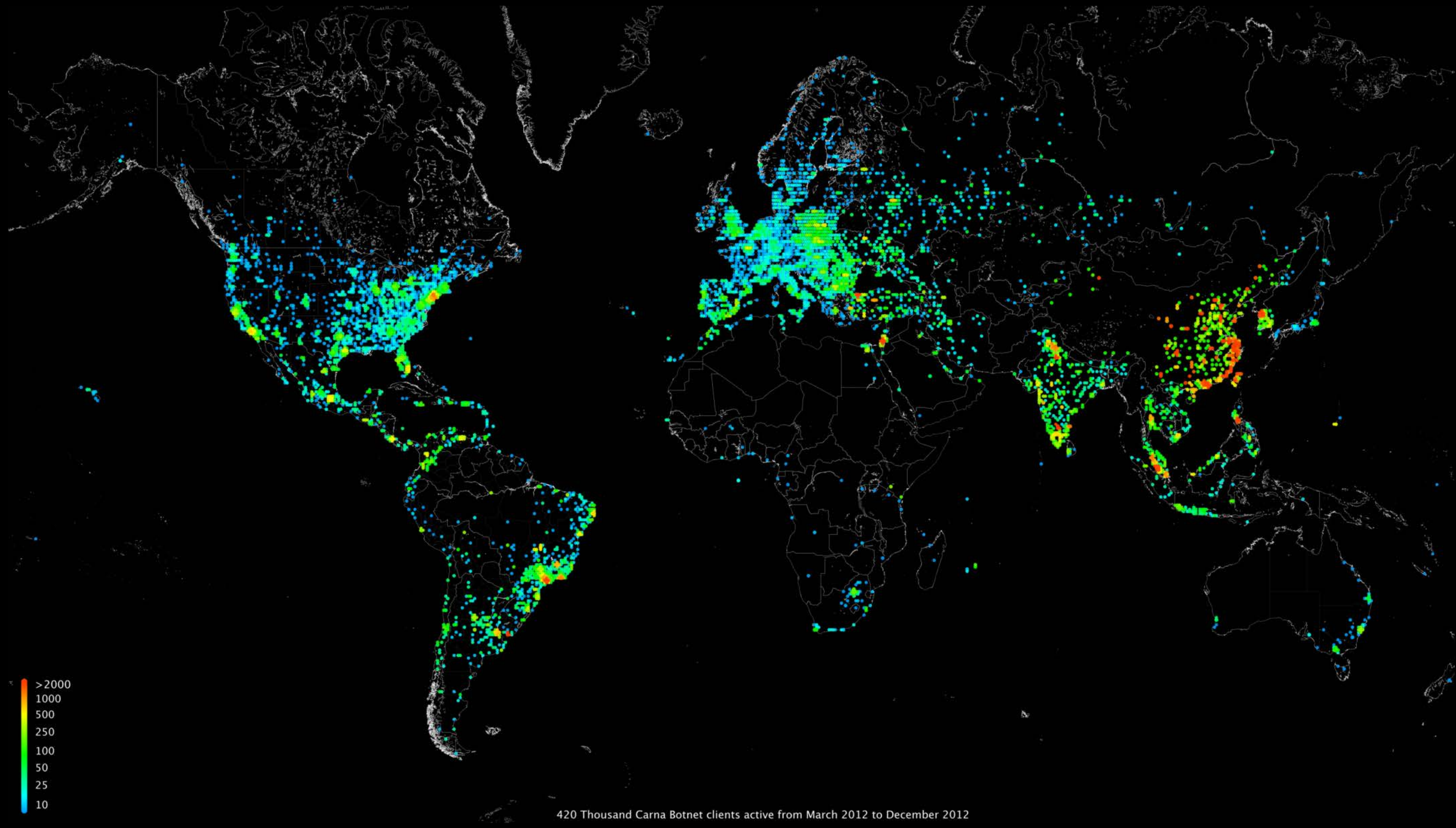
Mining Your Ps and Qs: Widespread Weak Keys in Network Devices
Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman
21st Usenix Security Symposium (Sec '12), August 2012

The Internet



Electronic Frontier Foundation

Carna botnet Internet Census 2012



Internet-Wide Network Studies

Previous research has shown promise of Internet-wide surveys

Census and Survey of the Visible Internet (2008)

EFF SSL Observatory: A glimpse at the CA ecosystem (2010)

Mining Ps and Qs: Widespread weak keys in network devices (2012)

Carna botnet Internet Census (2012)

Internet-Wide Network Studies

Previous research has shown promise of Internet-wide surveys

Census and Survey of the Visible Internet (2008)

3 months to complete ICMP census (2200 CPU-hours)

EFF SSL Observatory: A glimpse at the CA ecosystem (2010)

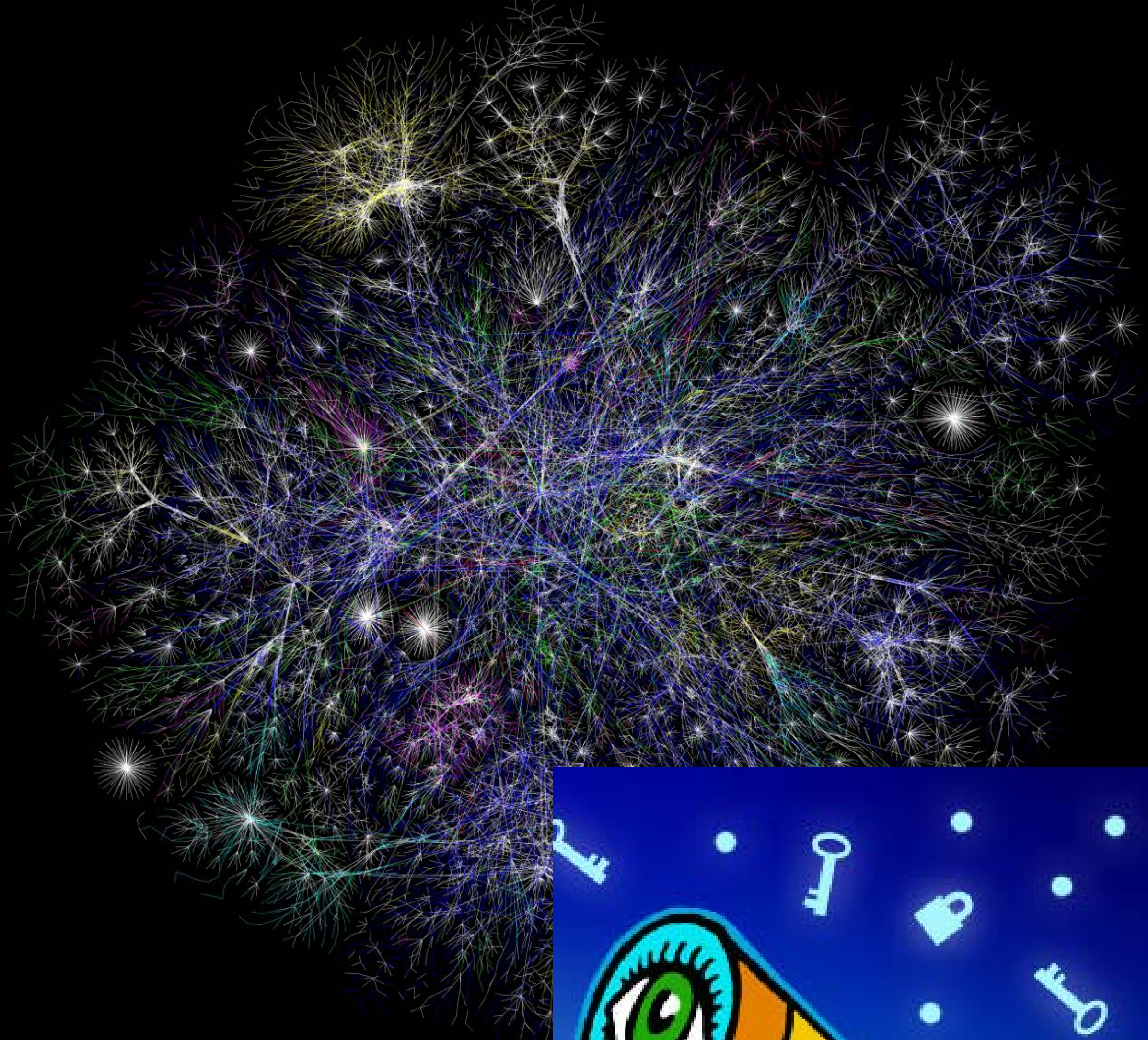
3 months on 3 Linux desktop machines (6500 CPU-hours)

Mining Ps and Qs: Widespread weak keys in network devices (2012)

25 hours acoss 25 Amazon EC2 Instances (625 CPU-hours)

Carna botnet Internet Census (2012)

420,000 usurped hosts



What if…?

What if Internet surveys didn’t require heroic effort?

What if we could scan the HTTPS ecosystem every day?

What if we wrote a whole-Internet scanner from scratch?

zmap

an open-source tool that can port scan the entire IPv4 address space from just one machine in under 45 minutes with 98% coverage

With Zmap, an Internet-wide TCP SYN scan on port 443 is as easy as:

```
$ zmap -p 443 -o results.txt
34,132,693 listening hosts
(took 44m12s)
```

97% of gigabit Ethernet linespeed

Demo time!

I'll do:

```
$ zmap -T4 -p `printf "%d" 0x30c3`
```

You can do:

```
$ tcpdump src port 12483
```

If you're on a public IP address, you should see a SYN from me by the end of the talk. (Look for 141.212/16.)

zmap	masscan
https://zmap.io	bit.ly/14GZzcT

Talk Roadmap

ZMap Scanner

1. **Architecture of ZMap**
2. Characterizing Performance

Applications of High Speed Scanning

1. Globally Observable Weak Keys
2. Uncovering the CA Ecosystem

ZMap Architecture

Existing Network Scanners

Reduce state by scanning in batches
- Time lost due to blocking
- Results lost due to timeouts

Track individual hosts and retransmit
- Most hosts will not respond

Avoid flooding through timing
- Time lost waiting

Utilize existing OS network stack
- Not optimized for immense number of connections

ZMap

Eliminate local per-connection state
- Fully asynchronous components
- No blocking except for network

Shotgun Scanning Approach
- Always send *n* probes per host

Scan widely dispersed targets
- Send as fast as network allows

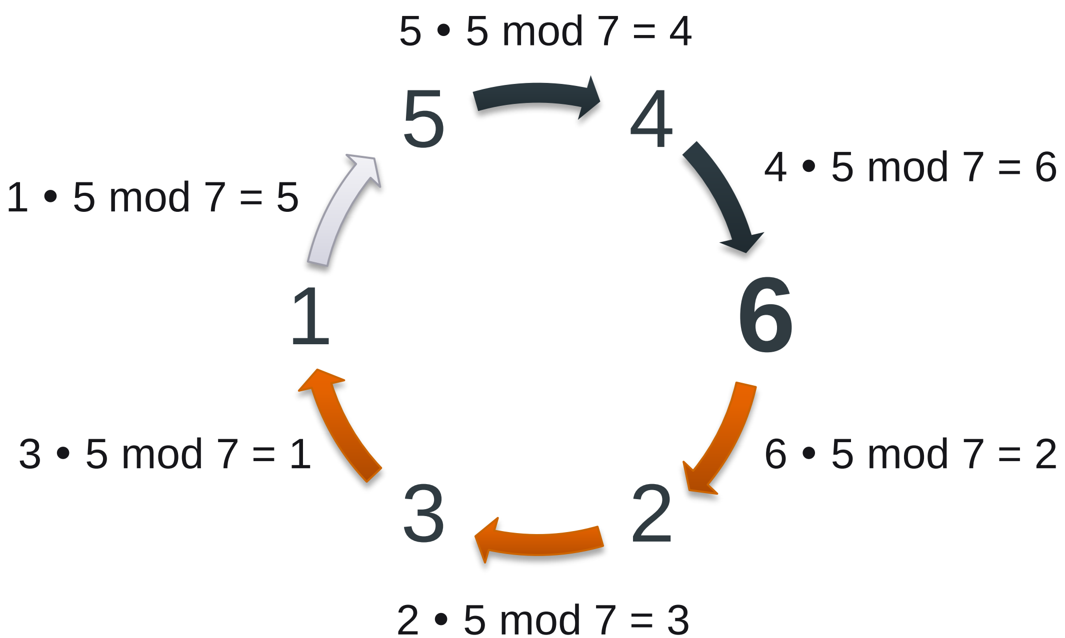
Probe-optimized Network Stack
- Bypass inefficiencies by generating Ethernet frames

Addressing Probes

How do we randomly scan addresses without excessive state?

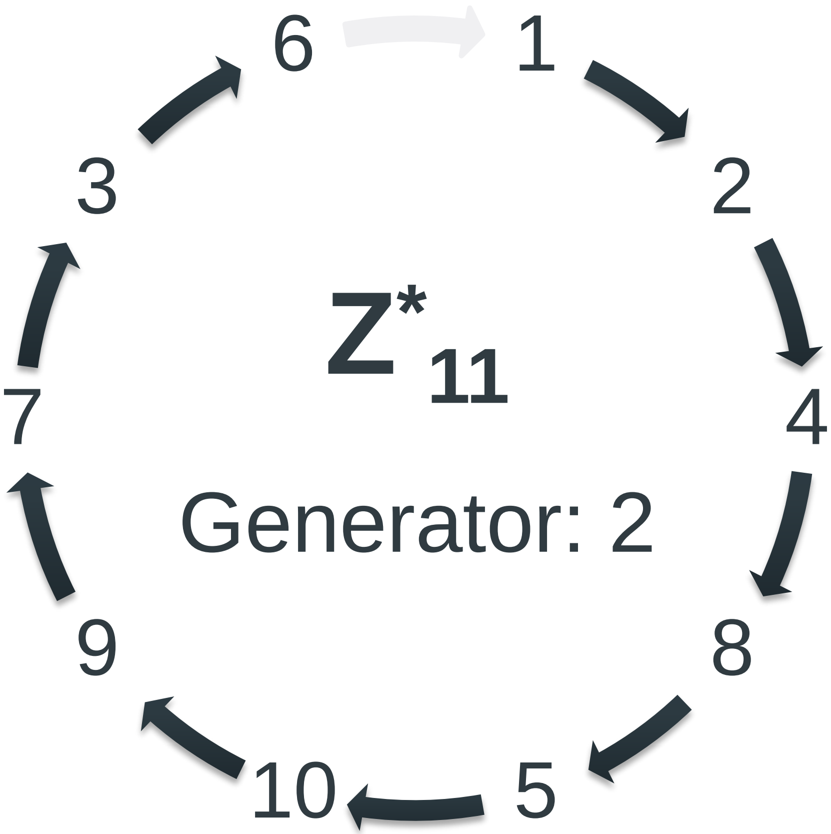
Scan hosts according to random permutation.

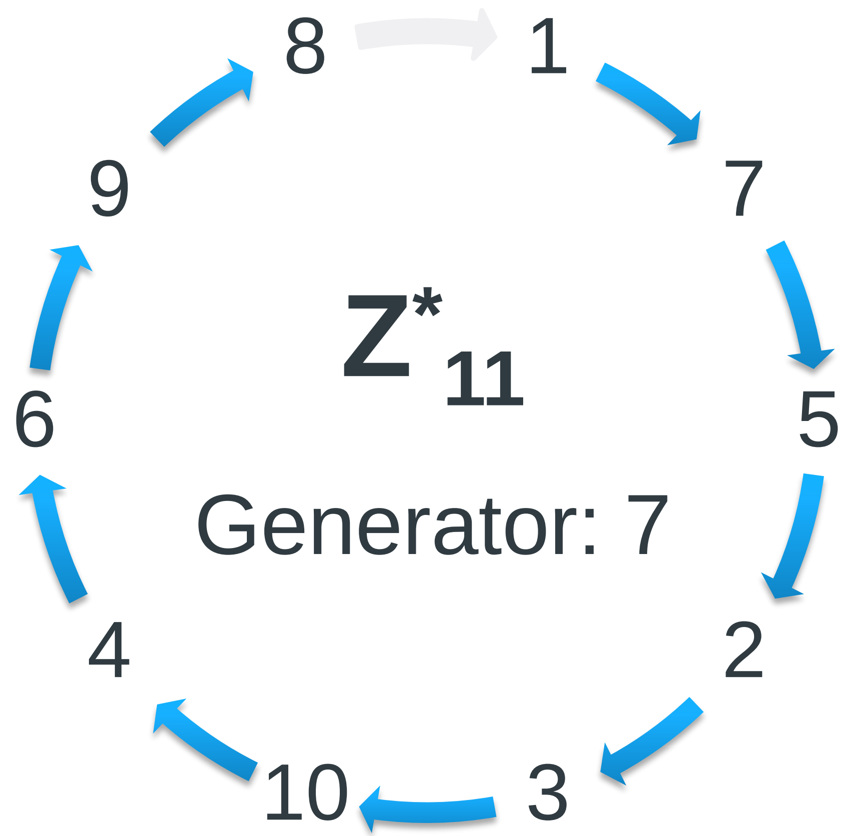
Iterate over multiplicative group of integers modulo *p*.



Negligible State

1. Primitive Root
2. Current Location
3. First Address

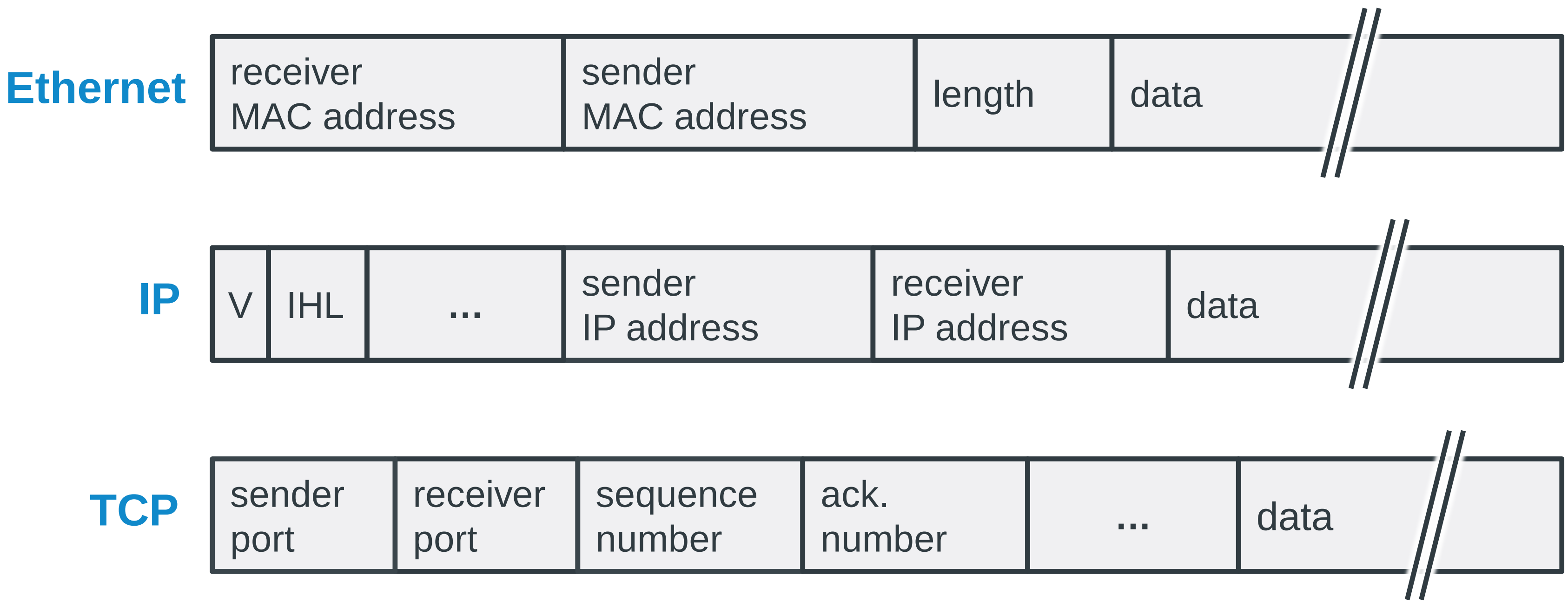

Z*11
Generator: 2
1
2
4
8
5
10
9
7
3
6


Z*11
Generator: 7
1
7
5
2
3
10
4
6
9
8

Validating Responses

How do we validate responses without local per-target state?

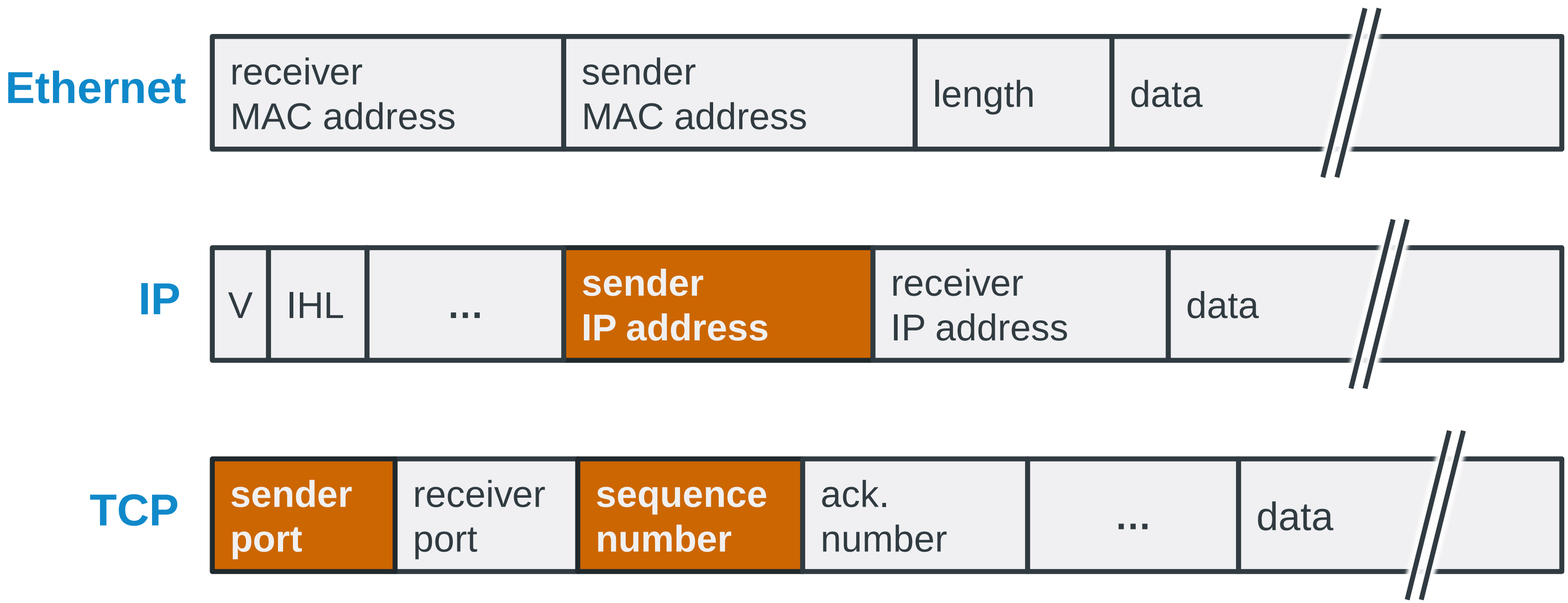
Encode secrets into mutable fields of probe packets
that will have recognizable effect on responses



Validating Responses

How do we validate responses without local per-target state?

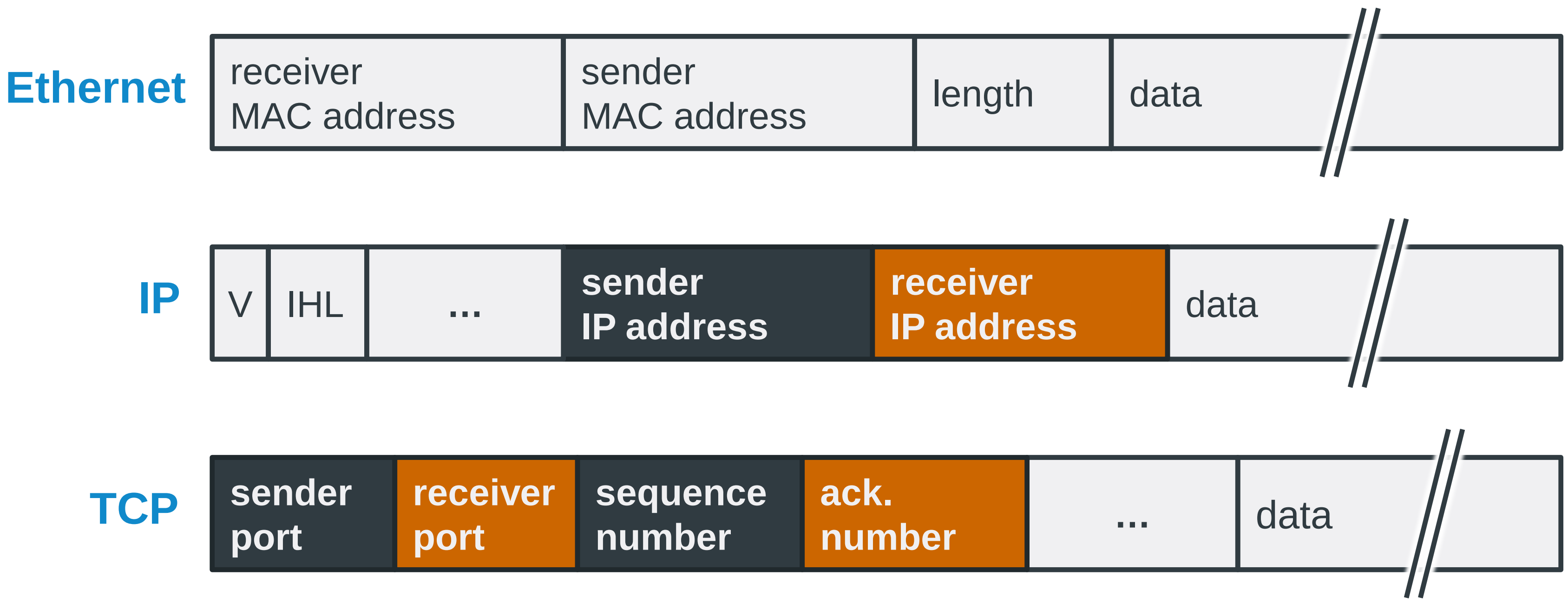
Encode secrets into mutable fields of probe packets
that will have recognizable effect on responses



Validating Responses

How do we validate responses without local per-target state?

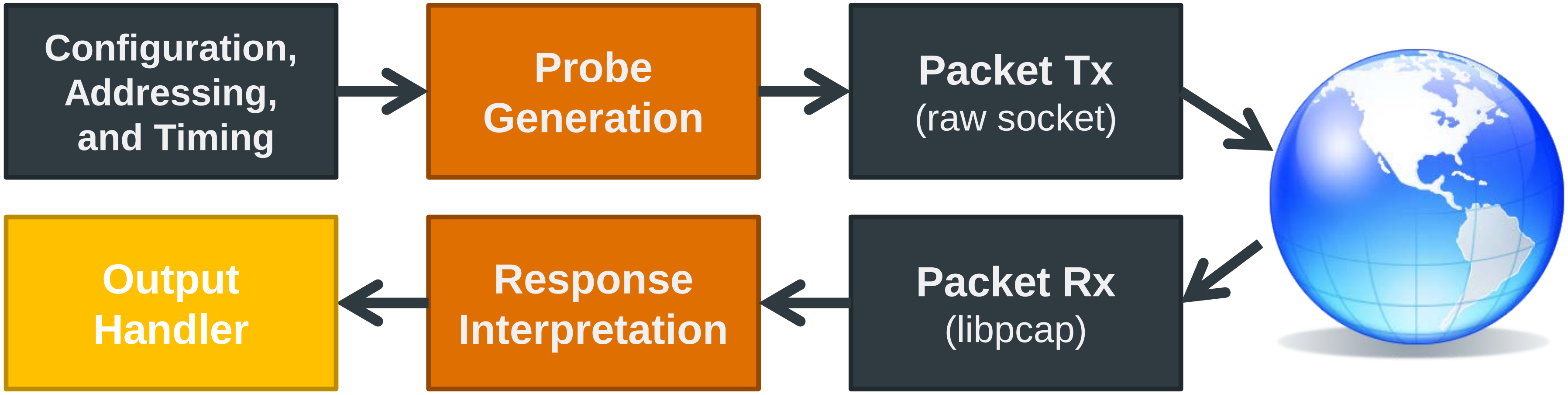
Encode secrets into mutable fields of probe packets
that will have recognizable effect on responses



Packet Transmission and Receipt

How do we make processing probes easy and fast?

1. **ZMap framework** handles the hard work
2. **Probe modules** fill in packet details, interpret responses
3. **Output modules** allow follow-up or further processing



Talk Roadmap

ZMap Scanner

1. Architecture of ZMap
2. **Characterizing Performance**

Applications of High Speed Scanning

1. Globally Observable Weak Keys
2. Uncovering the CA Ecosystem

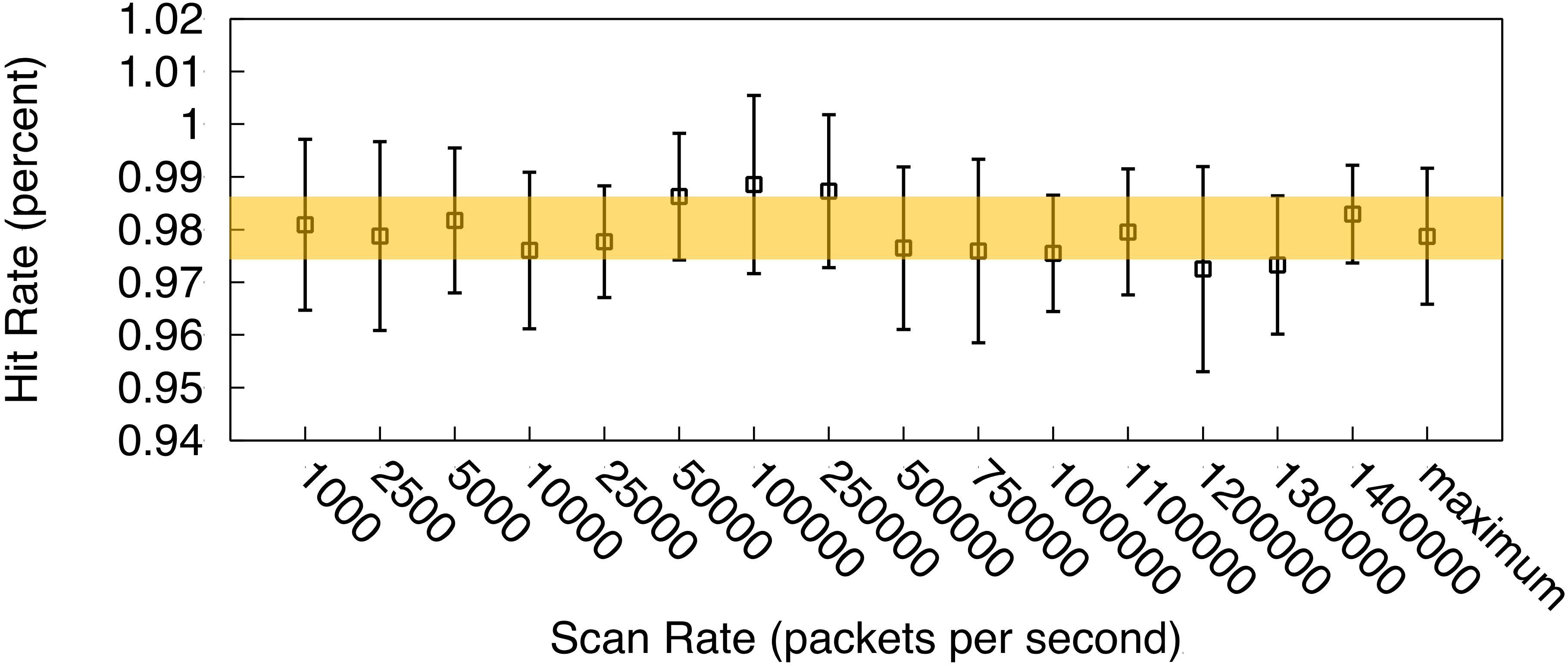
Scan Rate

How fast is too fast?

Your mileage may vary!

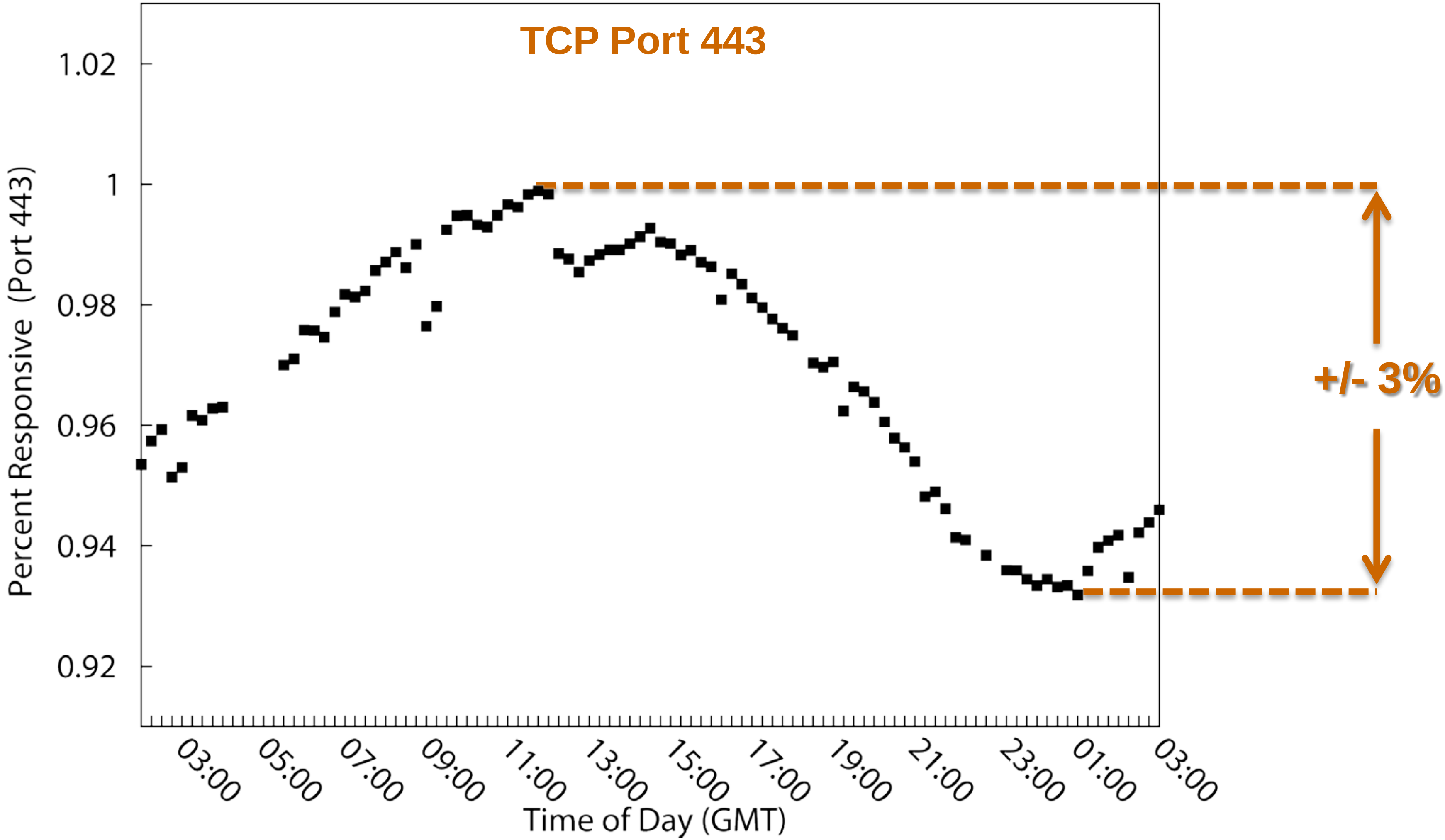
No meaningful correlation between sp

Slower scanning does not reveal additional hosts.



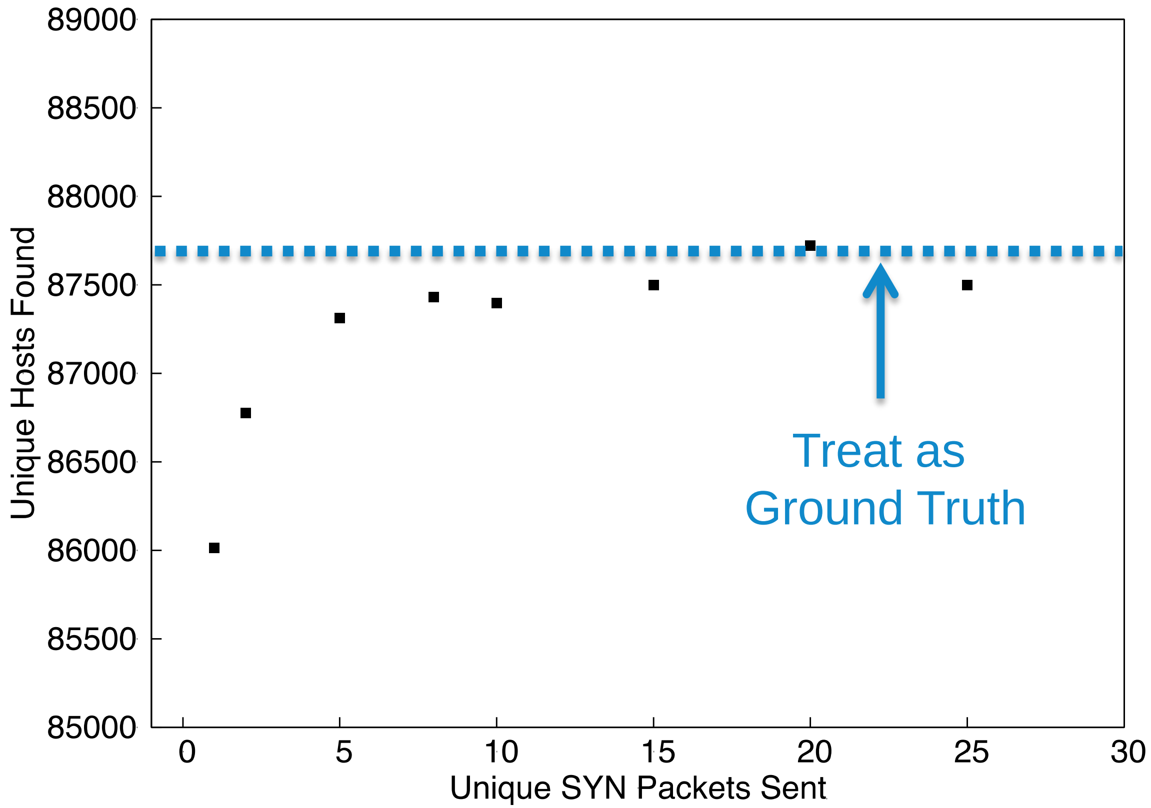
Temporal Variation

Response rates show significant diurnal variation.



Coverage

Is one probe sufficient?



We expect to see a plateau in response rate, regardless of additional probes.

Response Rate

1 Packet: 97.9%

2 Packets: 98.8%

3 Packets: 99.4%

Zmap vs. Nmap

Averages for scanning 1 million random hosts:

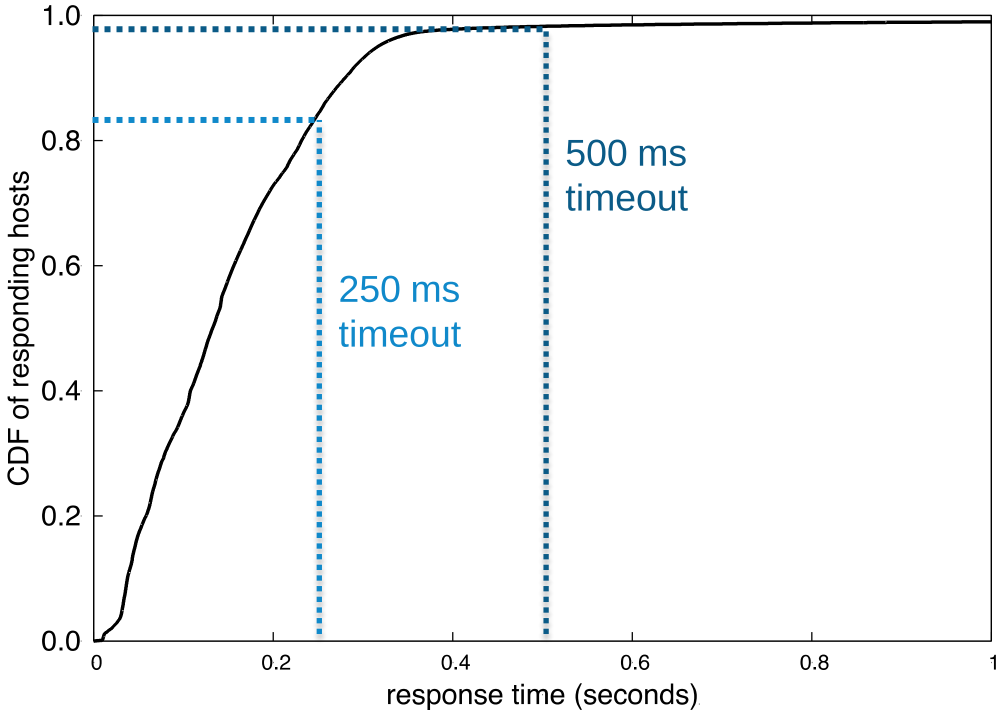
	Normalized Coverage	Duration (mm:ss)	Est. Internet Wide Scan
Nmap (1 probe)	81.4%	24:12	62.5 days
Nmap (2 probes)	97.8%	45:03	116.3 days
ZMap (1 probe)	98.7%	00:10	1:09:35
ZMap (2 probes)	100.0%	00:11	2:12:35

ZMap can scan more than **1300 times faster** than the most aggressive Nmap default configuration (“insane”)

Surprisingly, ZMap also finds more results than Nmap

Probe Response Times

Why does ZMap find more hosts than Nmap?



Response Times

250 ms:	**< 85%**
500 ms:	**98.2%**
1000 ms:	**99.0%**
8000 ms:	**99.9%**

Statelessness leads to both higher performance ***and*** increased coverage.

Talk Roadmap

ZMap Scanner

1. Architecture of ZMap
2. Characterizing Performance

Applications of High Speed Scanning

1. Globally Observable Weak Keys
2. Uncovering the CA Ecosystem

Enumerating Vulnerable Hosts

Discovering UPnP Vulnerabilities En Masse

HD Moore disclosed vulnerabilities in several common UPnP frameworks in January 2013.

Under 6 hours to code and run UPnP discovery scan. Custom probe module, 150 SLOC.

We found that 3.34 M of 15.7 M devices were vulnerable.

Compromise possible with a single UDP packet!

Uncovering Hidden Services

Enumerating Unadvertised Tor Bridges

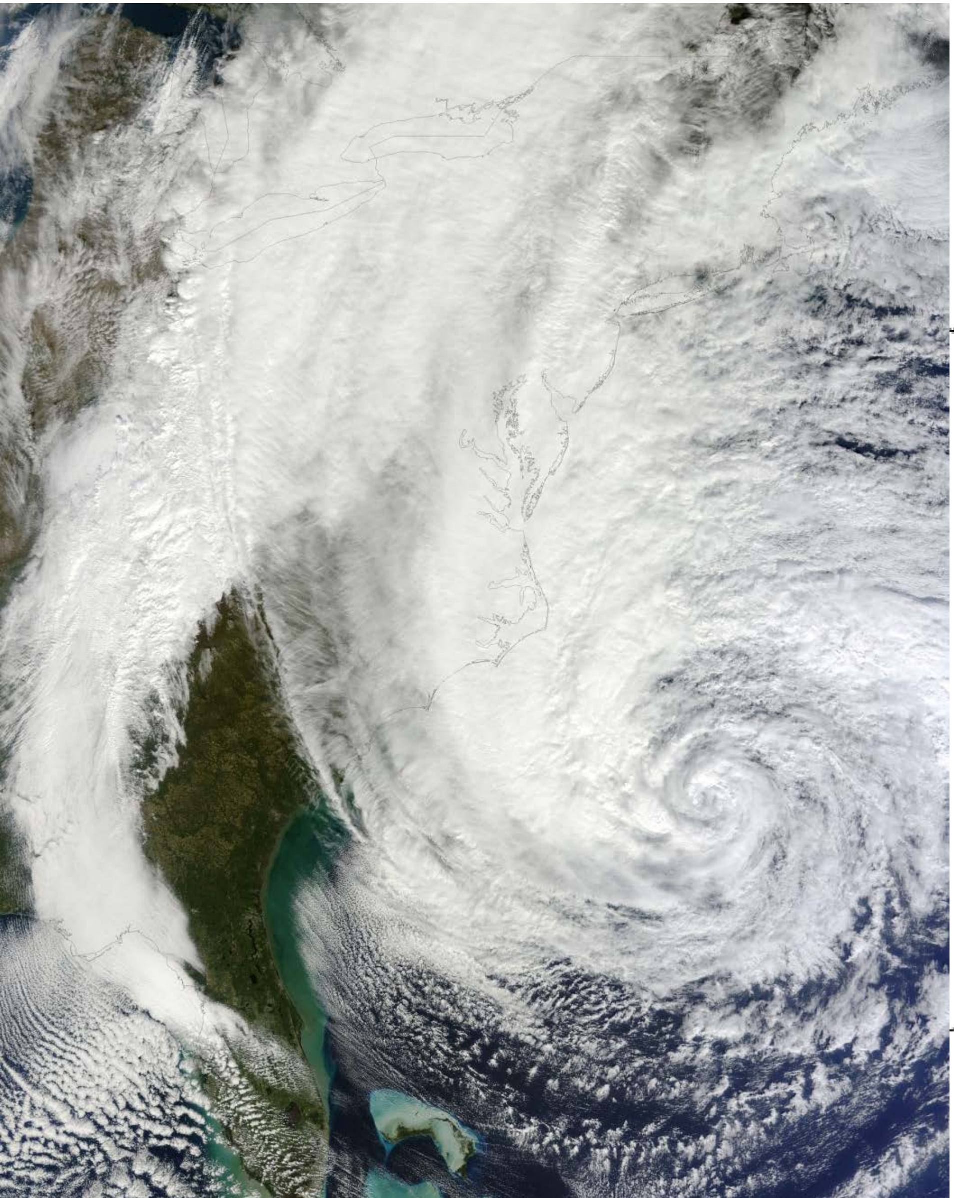
Scanning has potential to uncover unadvertised services

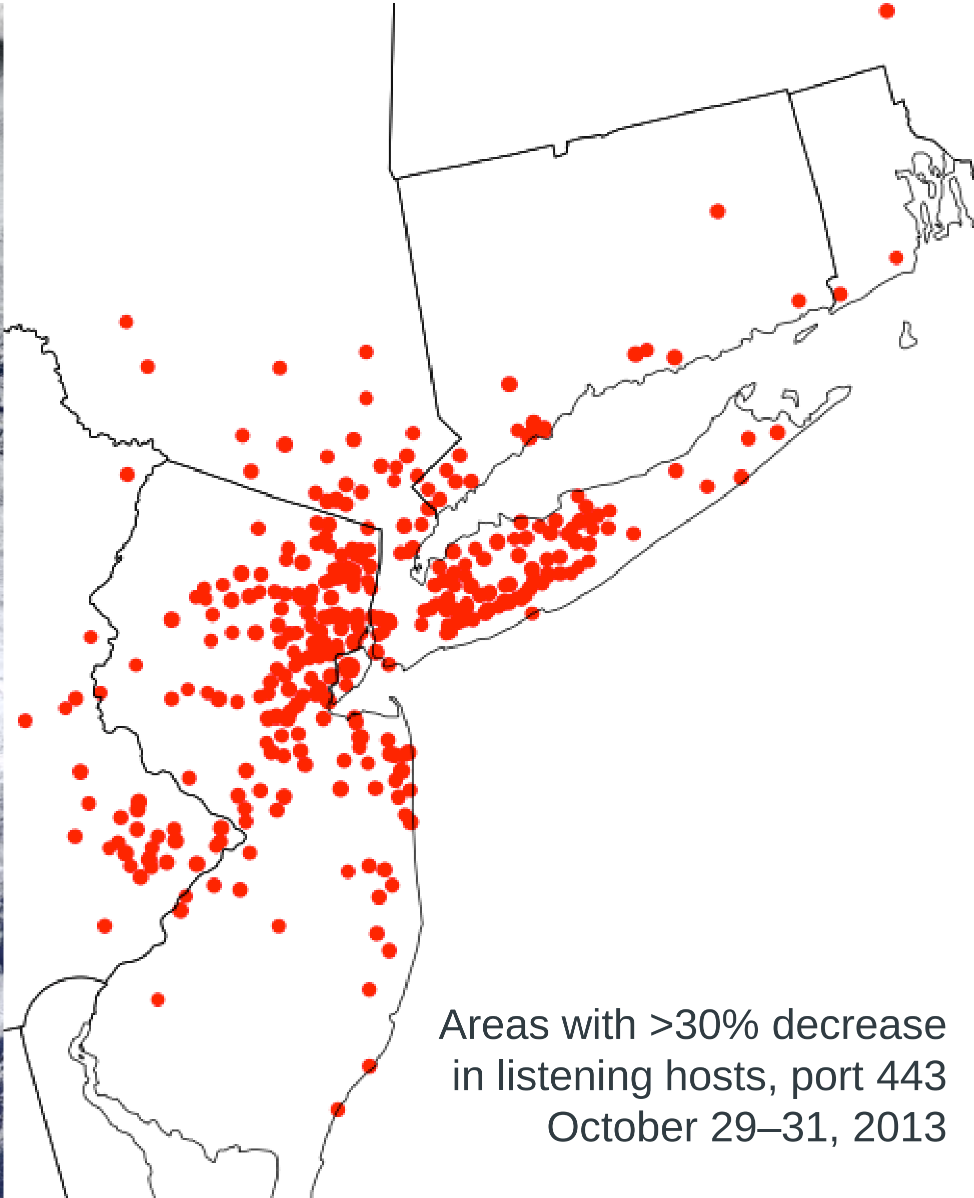
We perform a Tor handshake with public IPv4 addresses on port 9001 and 443

Identified >86% of live allocated Tor bridges with a single scan

(Tor has developed *obfsproxy* that listens on random ports to counter this type of attack.)

Detecting Service Disruptions





Areas with >30% decrease
in listening hosts, port 443
October 29–31, 2013

Globally Observable Phenomenon

Uncovering weak cryptographic keys and poor entropy collection

We considered the cryptographic keys used by HTTPS and SSH

	HTTPS	SSH
Live Hosts	12.8 million	10.2 million
Distinct RSA Public Keys	5.6 million	3.8 million
Distinct DSA Public Keys	6241	2.8 million

There are many legitimate reason that hosts might share keys…

Shared Cryptographic Keys

Why are a large number of hosts sharing cryptographic keys?

We find that 5.6% of TLS hosts and 9.6% of SSH hosts share keys in a vulnerable manner:

- Default certificates and keys
- Apparent entropy problems

What other, more serious, problems could be present if devices aren’t properly collecting entropy?

Factoring RSA Public Keys

What else could go wrong if devices aren't collecting entropy?

RSA Public Key: n = *p* • *q*, *p* and *q* are two large random primes

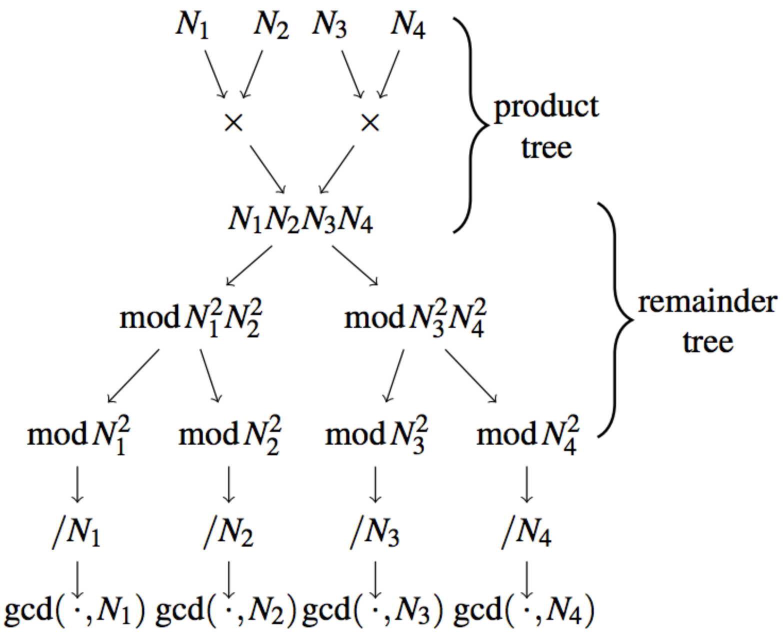
Most efficient known method of compromising an RSA key is to factor *n* back to *p* and *q*

While *n* is normally difficult to factor, for

$N_1 = p \cdot q_1$ and $N_2 = p \cdot q_2$

we can trivially compute

$p = GCD(N_1, N_2)$



Broken Cryptographic Keys

Why are a large number of hosts sharing cryptographic keys?

We find 2,134 distinct primes and compute the RSA private keys for **64,081 (0.50%) of TLS hosts**

Using another approach for DSA, we are able to compute the private keys for **105,728 (1.03%) of SSH hosts**

What was causing these vulnerable keys?

```
CN=self-signed, CN=system generated, CN=0168122008000024
CN=self-signed, CN=system generated, CN=0162092009003221
CN=self-signed, CN=system generated, CN=0162122008001051
C=CN, ST=Guangdong, O=TP-LINK Technologies CO., LTD., OU=TP-LINK SOFT, CN=TL-R478+1145D5C30089/emailAddre
C=CN, ST=Guangdong, O=TP-LINK Technologies CO., LTD., OU=TP-LINK SOFT, CN=TL-R478+139819C30089/emailAddre
CN=self-signed, CN=system generated, CN=0162072011000074
CN=self-signed, CN=system generated, CN=0162122009008149
CN=self-signed, CN=system generated, CN=0162122009000432
CN=self-signed, CN=system generated, CN=0162052010005821
CN=self-signed, CN=system generated, CN=0162072008005267
C=US, O=2Wire, OU=Gateway Device/serialNumber=360617088769, CN=Gateway Authentication
CN=self-signed, CN=system generated, CN=0162082009008123
CN=self-signed, CN=system generated, CN=0162072008005385
CN=self-signed, CN=system generated, CN=0162082008000317
C=CN, ST=Guangdong, O=TP-LINK Technologies CO., LTD., OU=TP-LINK SOFT, CN=TL-R478+3F5878C30089/emailAddre
CN=self-signed, CN=system generated, CN=0162072008005597
CN=self-signed, CN=system generated, CN=0162072010002630
CN=self-signed, CN=system generated, CN=0162032010008958
CN=109.235.129.114
CN=self-signed, CN=system generated, CN=0162072011004982
CN=217.92.30.85
CN=self-signed, CN=system generated, CN=0162112011000190
CN=self-signed, CN=system generated, CN=0162062008001934
CN=self-signed, CN=system generated, CN=0162112011004312
CN=self-signed, CN=system generated, CN=0162072011000946
C=US, ST=Oregon, L=Wilsonville, CN=141.213.19.107, O=Xerox Corporation,
CN=XRX0000AAD53FB7.eecs.umich.edu, CN=(141.213.19.107|XRX0000AAD53FB7.ee
CN=self-signed, CN=system generated, CN=0162102011001174
```

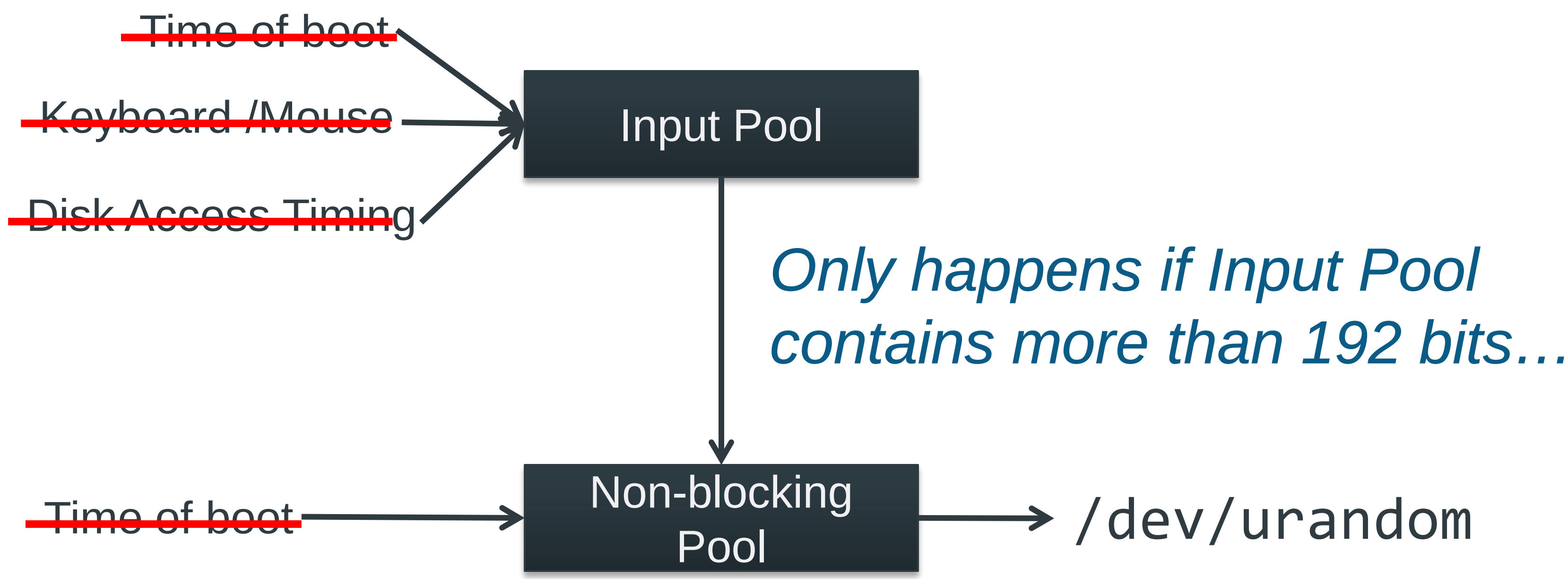
Most compromised keys are generated by headless or embedded network devices

Identified devices from > 40 manufacturers

Linux /dev/urandom

Why are embedded systems generating broken keys?

Nearly everything uses /dev/urandom

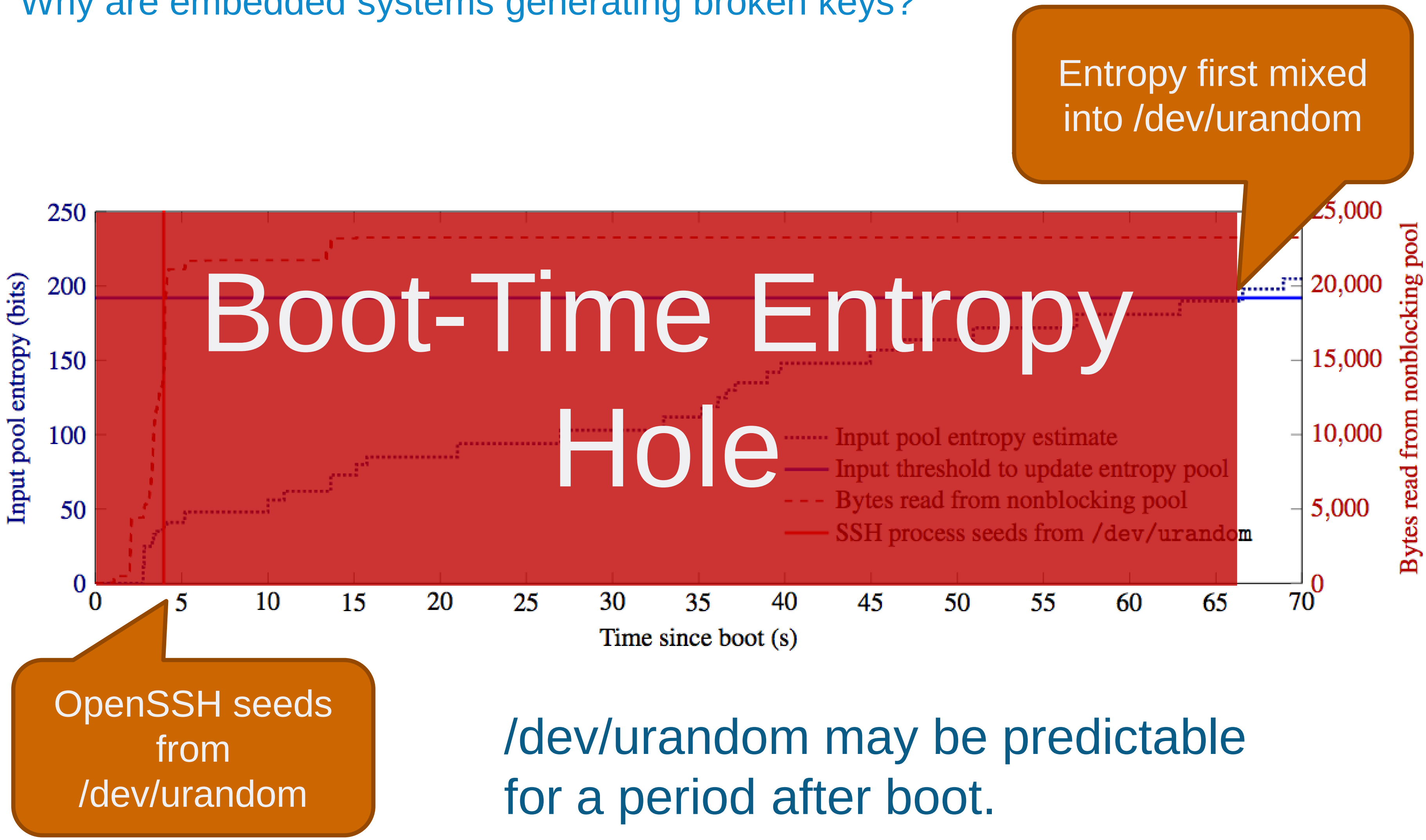


Problem 1: Embedded devices may lack all these sources

Problem 2: /dev/urandom can take a long time to “warm up”

Typical Ubuntu Server Boot

Why are embedded systems generating broken keys?



/dev/urandom may be predictable for a period after boot.

Moving Forward

What do we do about fixing the Linux kernel and affected devices?

Patches have been committed to the Linux 3.x Kernel

- Use interrupts until other entropy is available
- Mix in unique information such as MAC address

Manufacturers have been notified. DHS, ICS-CERT, NSA, JPCERT, and other agencies are working with affected companies and helping manufacturers correct vulnerabilities.

Online Key Check Service available at https://factorable.net

Talk Roadmap

ZMap Scanner

1. Architecture of ZMap
2. Characterizing Performance

Applications of High Speed Scanning

1. Globally Observable Weak Keys
2. **Exposing the CA Ecosystem**

Certificate Authority Ecosystem

HTTPS is dependent on a supporting PKI composed of "certificate authorities" that vouch for websites' identities.

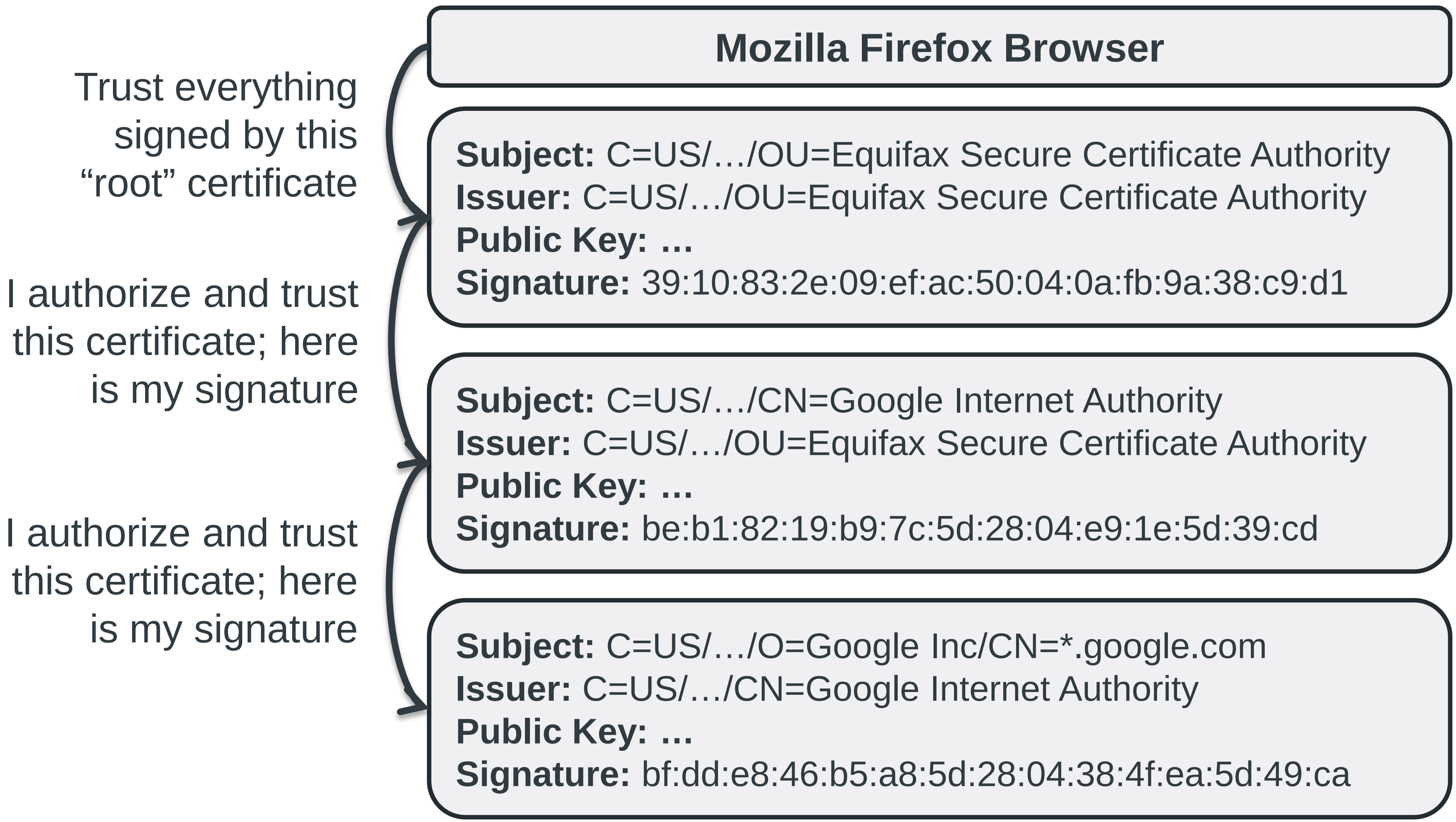
Every certificate authority can sign for *any* website.

There is no central repository of certificate authorities.

- We don't know who we trust until we see CAs in the wild…

Certificate Chains

A Brief Review of Certificates



Certificate Chains

A Brief Review of Certificates

Mozilla Firefox Browser

Trust everything signed by this “root” certificate

Subject: C=US/…/OU=Equifax Secure Certificate Authority
Issuer: C=US/…/OU=Equifax Secure Certificate Authority
Public Key: …
Signature: 39:10:83:2e:09:ef:ac:50:04:0a:fb:9a:38:c9:d1

I authorize and trust this certificate; here is my signature

Subject: C=US/…/CN=Google Internet Authority
Issuer: C=US/…/OU=Equifax Secure Certificate Authority
Public Key: …
Signature: be:b1:82:19:b9:7c:5d:28:04:e9:1e:5d:39:cd

I authorize and trust this certificate; here is my signature

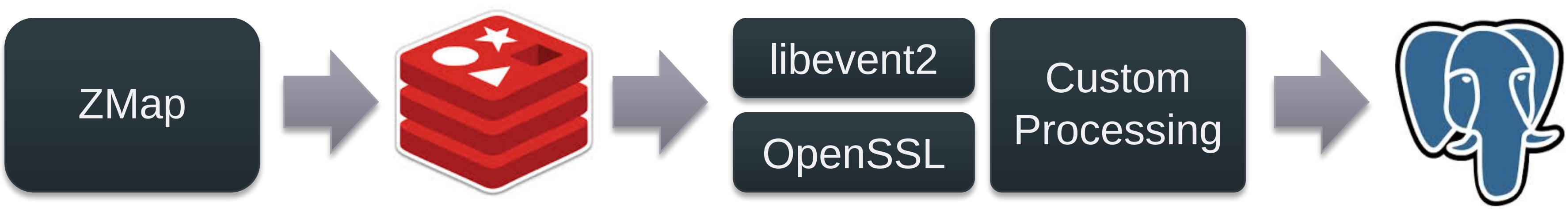
Subject: C=US/…/O=Google Inc/CN=*.google.com
Issuer: C=US/…/CN=Google Internet Authority
Public Key: …
Signature: bf:dd:e8:46:b5:a8:5d:28:04:38:4f:ea:5d:49:ca

Uncovering the HTTPS Ecosystem

How do we regularly collect certificates from Internet?

We completed 110 scans of the HTTPS ecosystem over the last year

1. Identify certificate authorities
2. Uncover worrisome practices



We collected **42 million unique certificates** of which **6.9 million were browser trusted** from **109 million unique hosts**

Identifying Certificate Authorities

Who do we trust to correctly sign certificates?

Identified 1,800 CA certificates belonging to 683 organizations

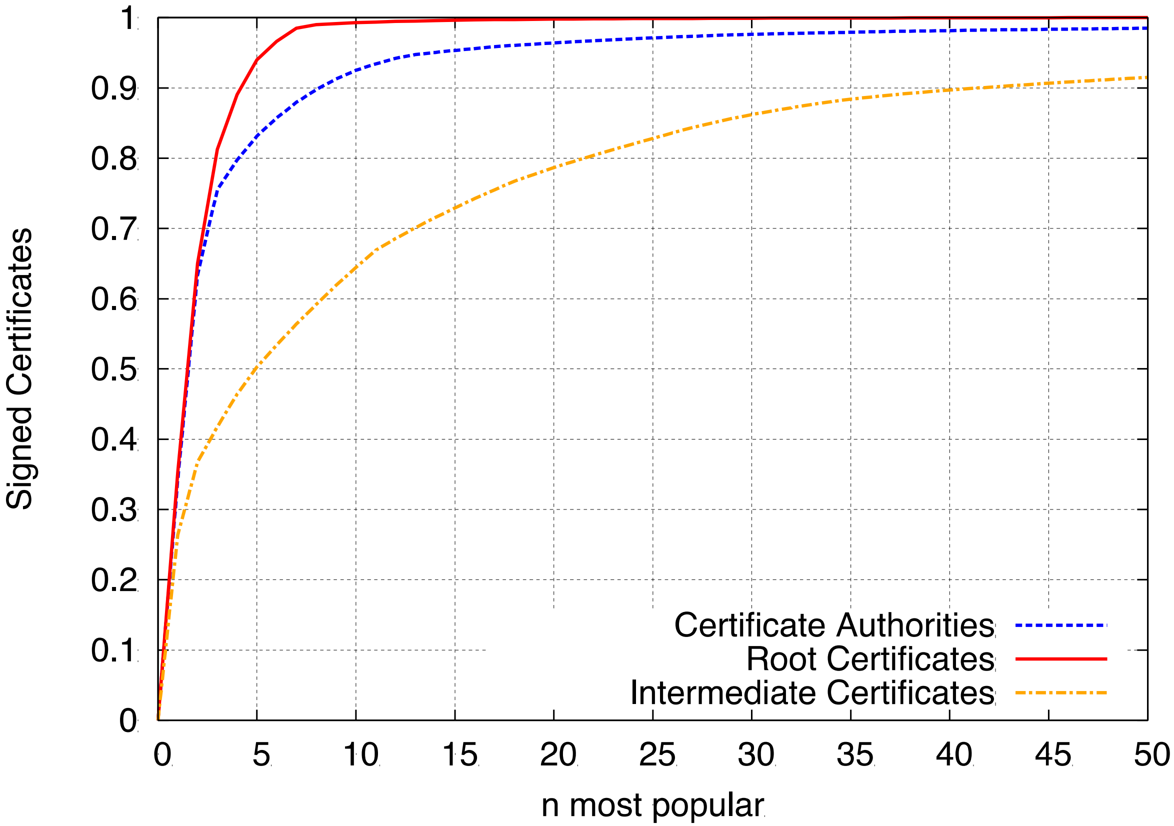
- Including religious institutions, libraries, non-profits, financial institutions, governments, and hospitals
- More than 80% of organizations controlling a CA certificate aren’t commercial certificate authorities

More than half of the certificates were provided by the German National Research and Education Network (DFN)

All major browser roots are selling intermediates to third-party organizations without any constraints

Distribution of Trust

Who actually signs the certificates we use on a daily basis?



90% of Trusted Certificates

- **-** signed by 5 organizations
- descendants of 4 roots
- signed by 40 intermediates

Symantec, GoDaddy, and Comodo control 75% of the market through acquisitions

26% of trusted sites are signed by a single intermediate certificate!

Ignoring Foundational Principles

What are authorities doing that puts the ecosystem at risk?

We classically teach concepts such as *defense in depth* and the *principle of least privilege*

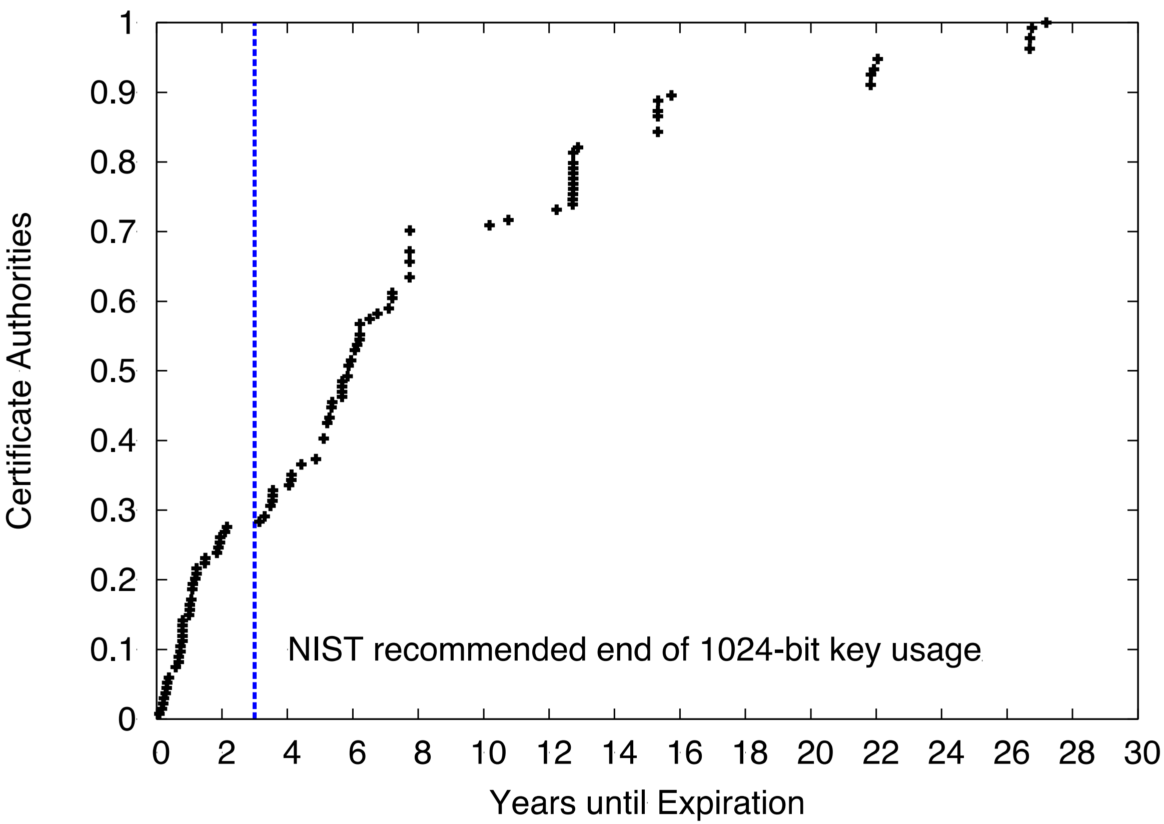
We have methods of constraining what CAs can sign for, yet all but 7 of the 1,800 CA certs we found can sign for anything

Lack of constraints allowed a rogue CA certificate in 2012, but in another case prevented 1,400 invalid certificates

Almost 5% of certificates include local domains, e.g. localhost, mail, exchange

Cryptographic Reality

What are authorities doing that puts the ecosystem at risk?



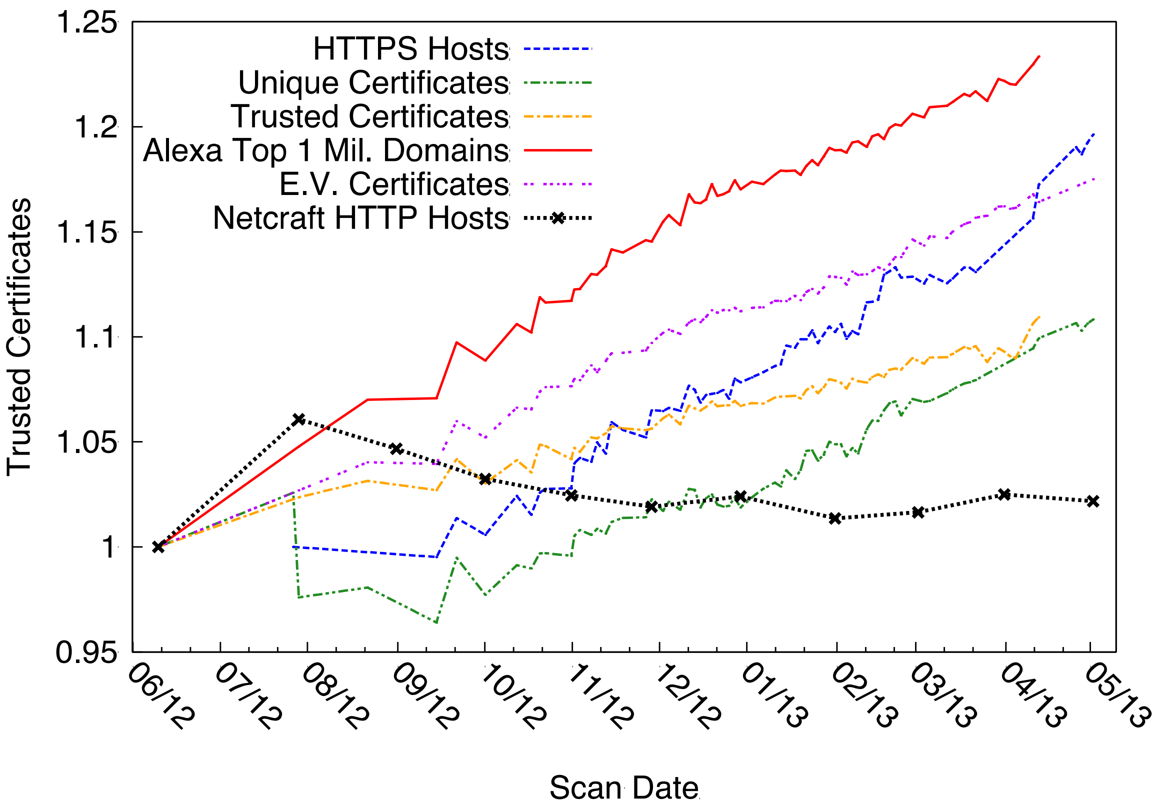
90% of certificates use a 2048 or 4096-bit RSA key

50% of certificates are rooted in a 1024-bit key

More than 70% of these roots will expire after 2016

Growth in HTTPS Adoption

What has changed in the last year of scanning?



June 2012–May 2013

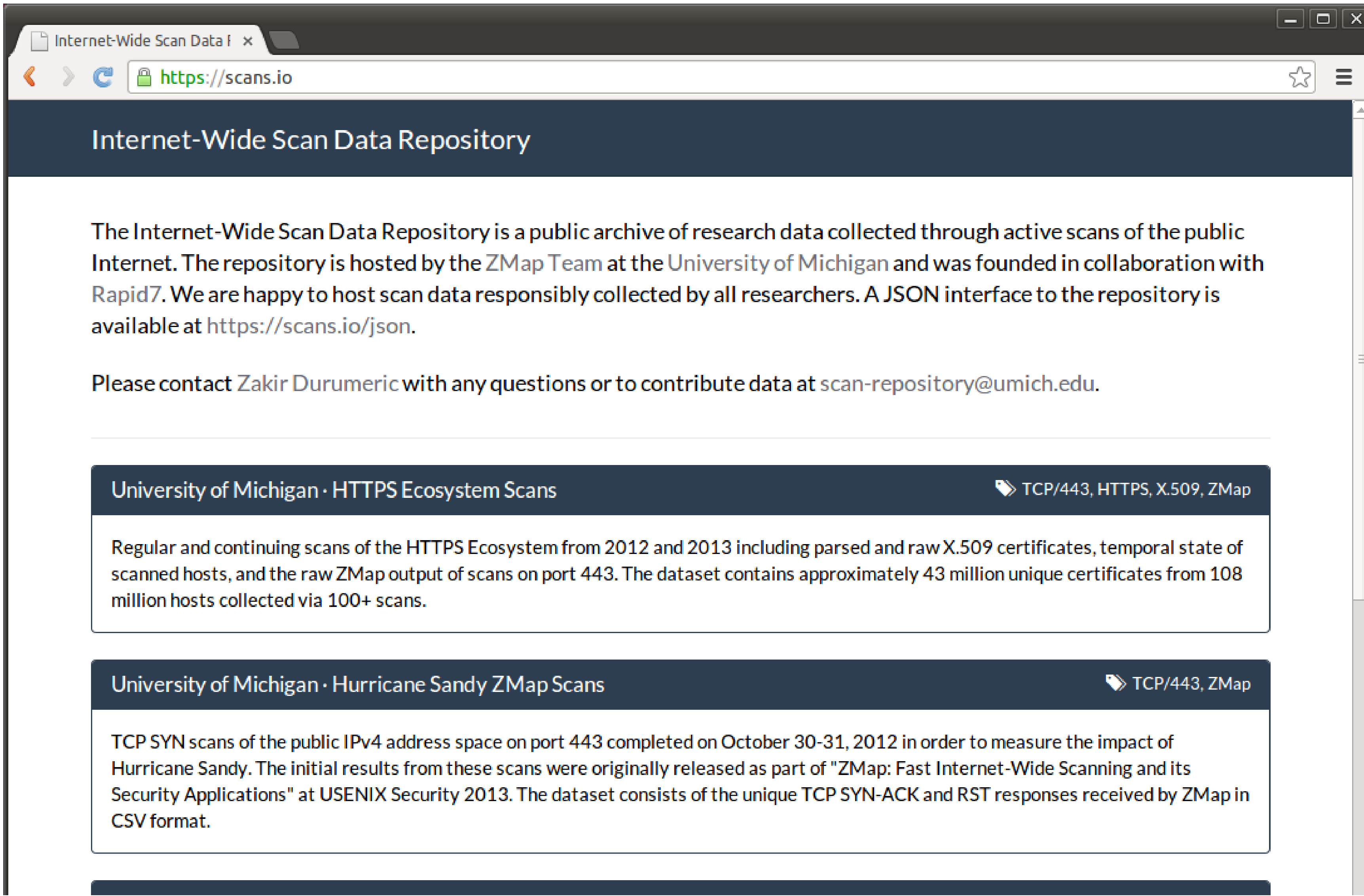
10% ⇧ HTTPS servers.

23% ⇧ Use on Alexa Top-1M sites.

11% ⇧ Browser-trusted certificates.

Scans.IO Data Repository

How do we share all this scan data?



ZMap Public Release

ZMap is an actively developed open source project

Downloaded it now from **https://zmap.io**

Scanning the Internet ***really is*** as simple as:

```
$ zmap -p 443 -o results.txt
```

Let’s check on our demo...

Ethics of Active Scanning

Considerations

- Impossible to request permission from all owners
- No IP-level equivalent to robots exclusion standard
- Administrators may believe that they are under attacka

Reducing Scan Impact

- Scan in random order to avoid overwhelming networks
- Signal benign nature over HTTP and w/ DNS hostnames
- Honor all requests to be excluded from future scans

Bottom Line: Be a Good Neighbor

User Responses

Over 200 Internet-wide scans over 1.5 years (>1 trillion probes)

Responses from 145 users

Blacklisted 91 entities
(3.7 M total addresses)

15 hostile responses

2 cases of retaliatory traffic

Entity Type	Responses
Small Business	41
Home User	38
Corporation	17
Academic Institution	22
Government	15
ISP	2
Unknown	10
Total	**145**

Iranians used University of ×

freebeacon.com/iran-strikes-back/

THE WASHINGTON FREE BEACON

Iran Strikes Back

Iranians used University of Michigan network in recent bank attacks

Recommend 724 | Follow @FreeBeacon


Secretary of Defense Leon Panetta / AP

BY: Bill Gertz Follow @BillGertz

October 15, 2012 5:00 am

UPDATE, Oct. 19, 2012: The University of Michigan on Oct. 18 denied its computers were involved in the Iranian cyber attacks and said it believes the security firm's reporting erroneously monitored activity done by a professor engaged in security research.

"These assertions are simply not true," said Paul Howell, the university's chief information technology security officer.

The statement said the university believes the reported hacking attempts were "actually benign connection attempts generated from one computer, not a network, in the College of Engineering."

Future Work

10gigE Network Surveys

TLS Server Name Indication

Scanning Exclusion Standards

IPv6 Scanning Methdology?

Use scanning to do great research!

Conclusion

Living in a unique period

IPv4 can be quickly, exhaustively scanned

IPv6 has not yet been widely deployed

Low barriers to entry for Internet-wide surveys

Now possible to scan the entire IPv4 address space from **one host** in under **45 minutes** with **98% coverage**

Explored applications of high-speed scanning

My goal is to enable all of you to do more research

zmap

masscan

https://zmap.io

bit.ly/14GZzcT

Scan Data Repository

https://scans.io