ZMap | Fast Internet-Wide Scanning, Weak Keys and the HTTPS Certificate Ecosystem

Zakir Durumeric Michael Bailey

University of Michigan

Internet-Wide Network Studies

Previous research has shown promise of Internet-wide surveys

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

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

Census and Survey of the Visible Internet (2008)

Internet-Wide Network Studies

Previous research has shown promise of Internet-wide surveys

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

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

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

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

Census and Survey of the Visible Internet (2008)

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

EFF

SSL
Observatory

NASA
esa

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?

Talk Roadmap

ZMap Scanner

1. **Philosophy and Architecture of ZMap**
2. Characterizing ZMap's Performance

Applications of High Speed Scanning

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

ZMap: The Internet Scanner

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

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

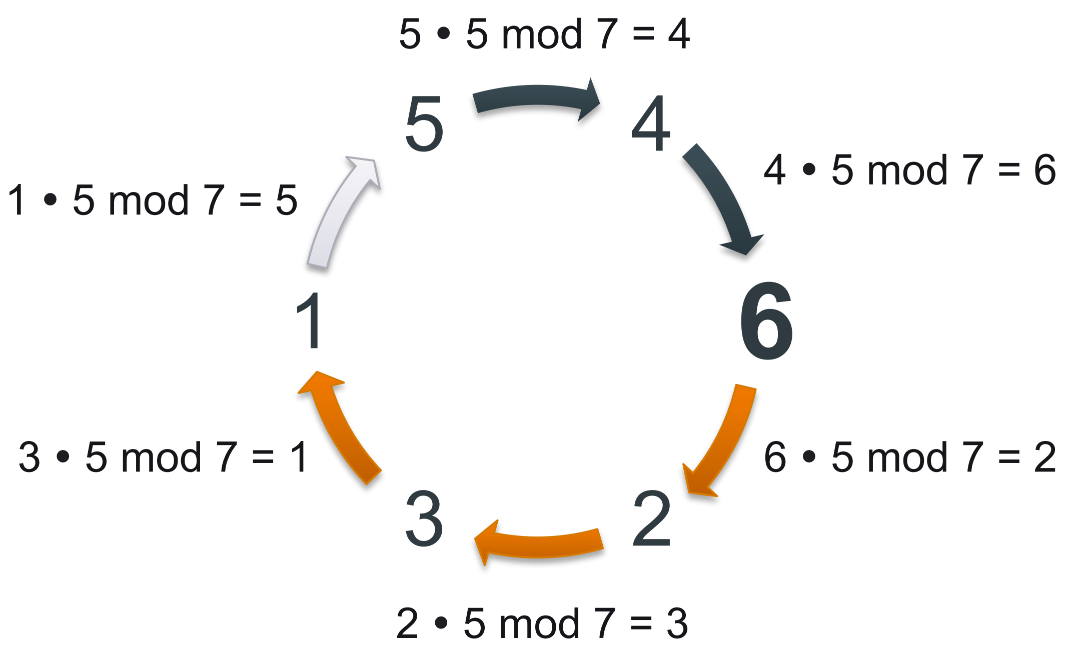
Probe-optimized Network Stack

- Bypass inefficiencies by generating Ethernet frames

Addressing Probes

How do we randomly scan addresses without excessive state?

1. Scan hosts according to random permutation
2. Iterate over multiplicative group of integers modulo *p*



1 • 5 mod 7 = 5

5 • 5 mod 7 = 4

4 • 5 mod 7 = 6

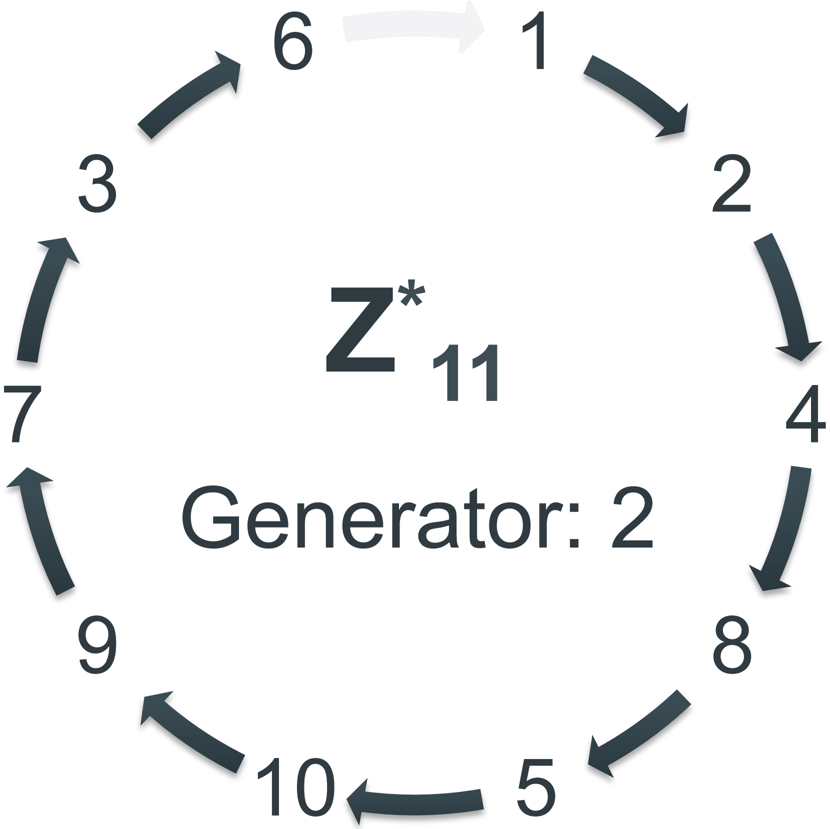
6 • 5 mod 7 = 2

2 • 5 mod 7 = 3

3 • 5 mod 7 = 1

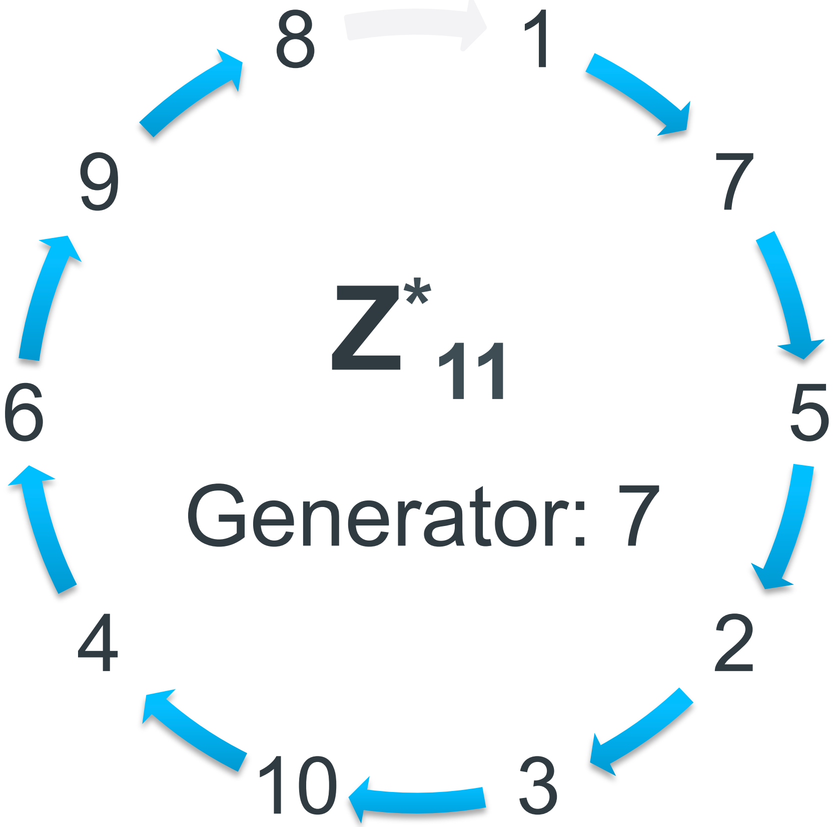
Negligible State

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



Z^*_{11}

Generator: 2



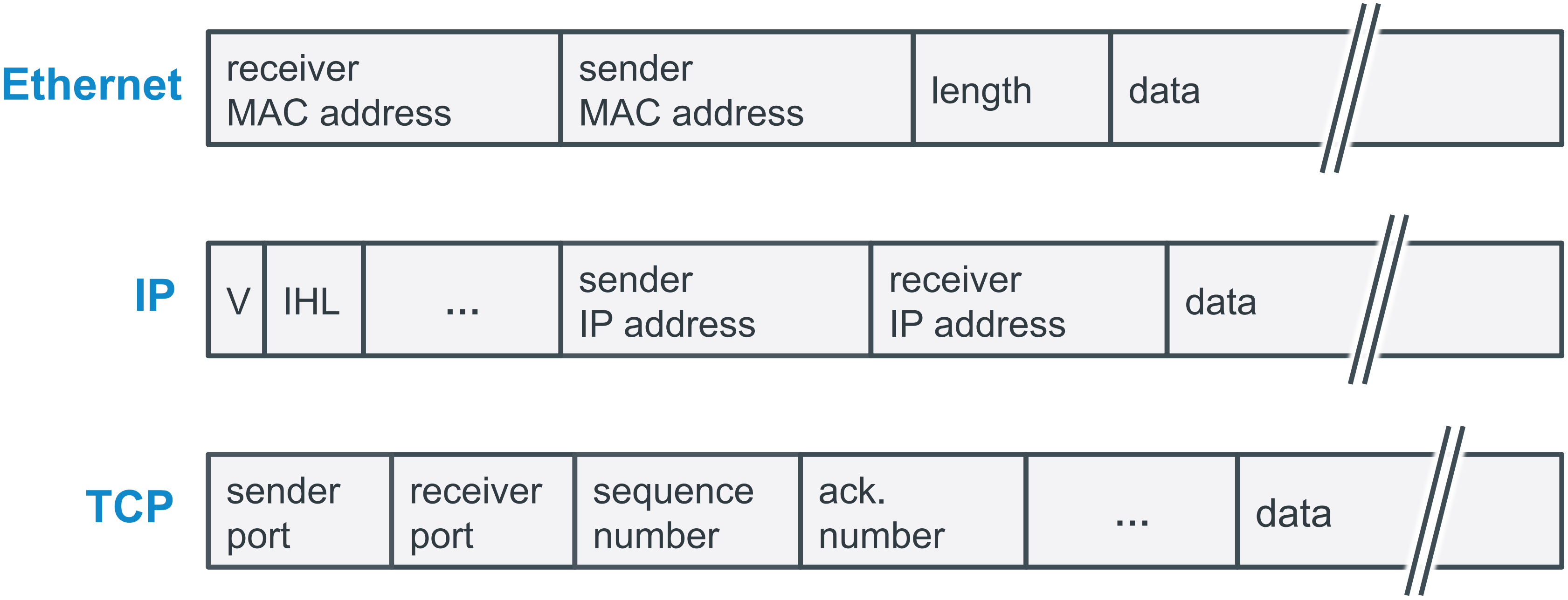
Z^*_{11}

Generator: 7

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

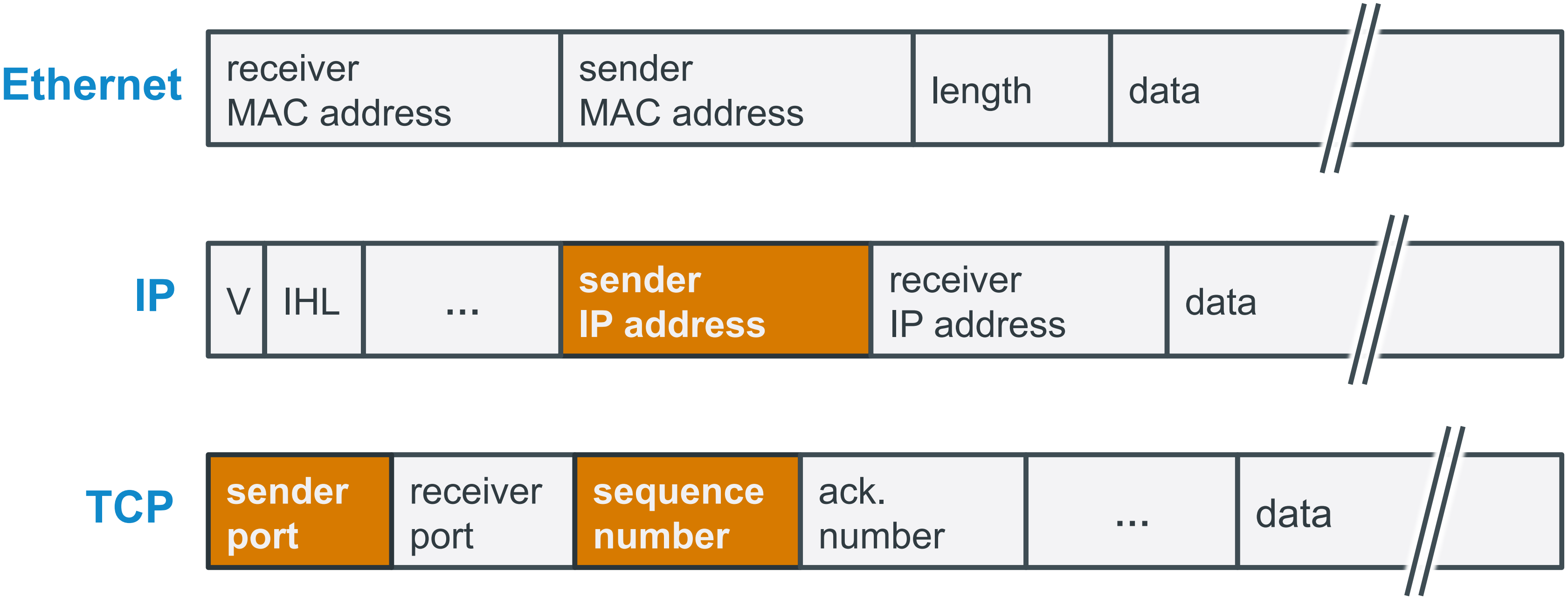


Ethernet	receiver MAC address	sender MAC address	length	data		
IP	V	IHL	...	sender IP address	receiver IP address	data
TCP	sender port	receiver port	sequence number	ack. number	...	data

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



Ethernet

receiver MAC address | sender MAC address | length | data

IP

V | IHL | ... | **sender IP address** | receiver IP address | data

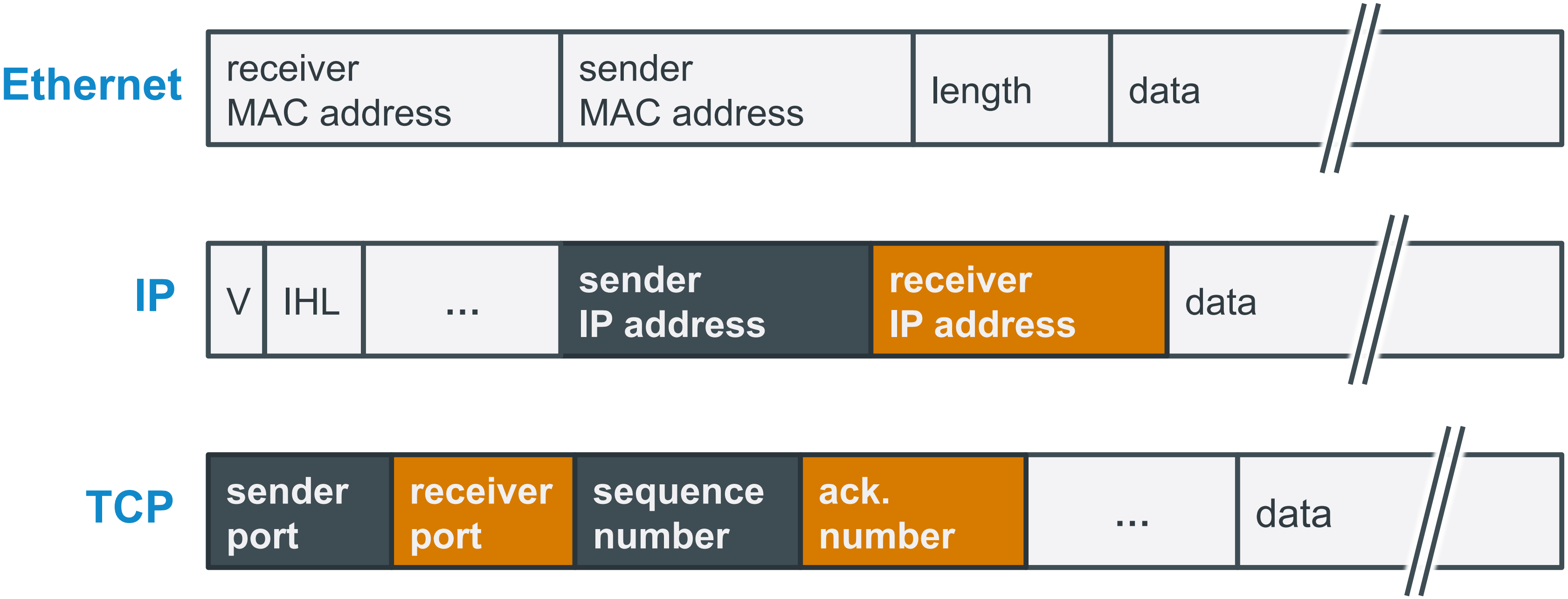
TCP

sender port | receiver port | **sequence number** | ack. number | ... | data

Validating Responses

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

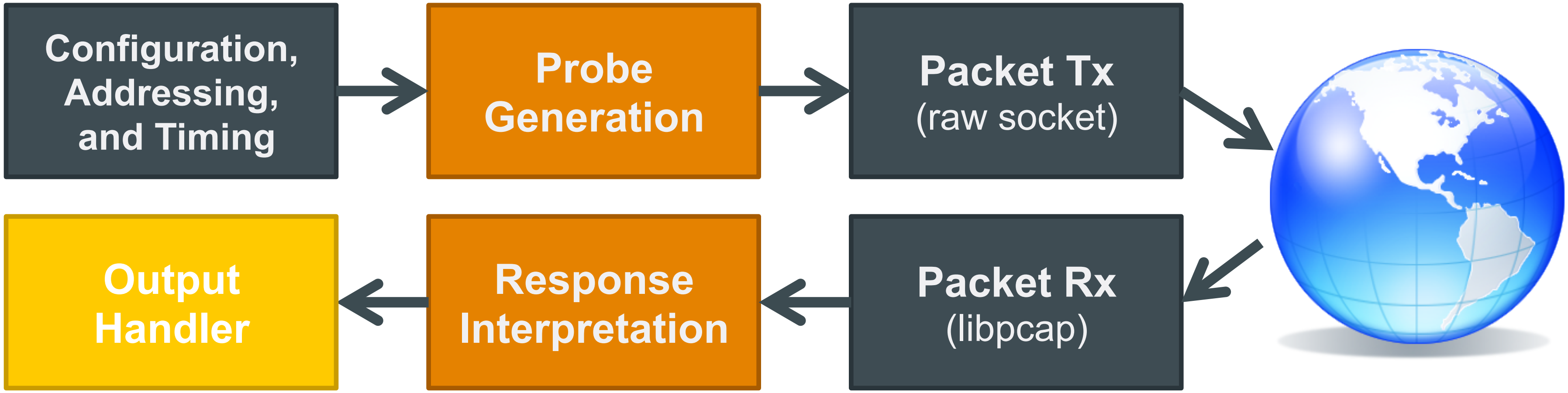
Encode secrets into mutable fields of probe packets that will have recognizable effect on responses



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. Philosophy and Architecture of ZMap
2. **Characterizing ZMap's Performance**

Applications of High Speed Scanning

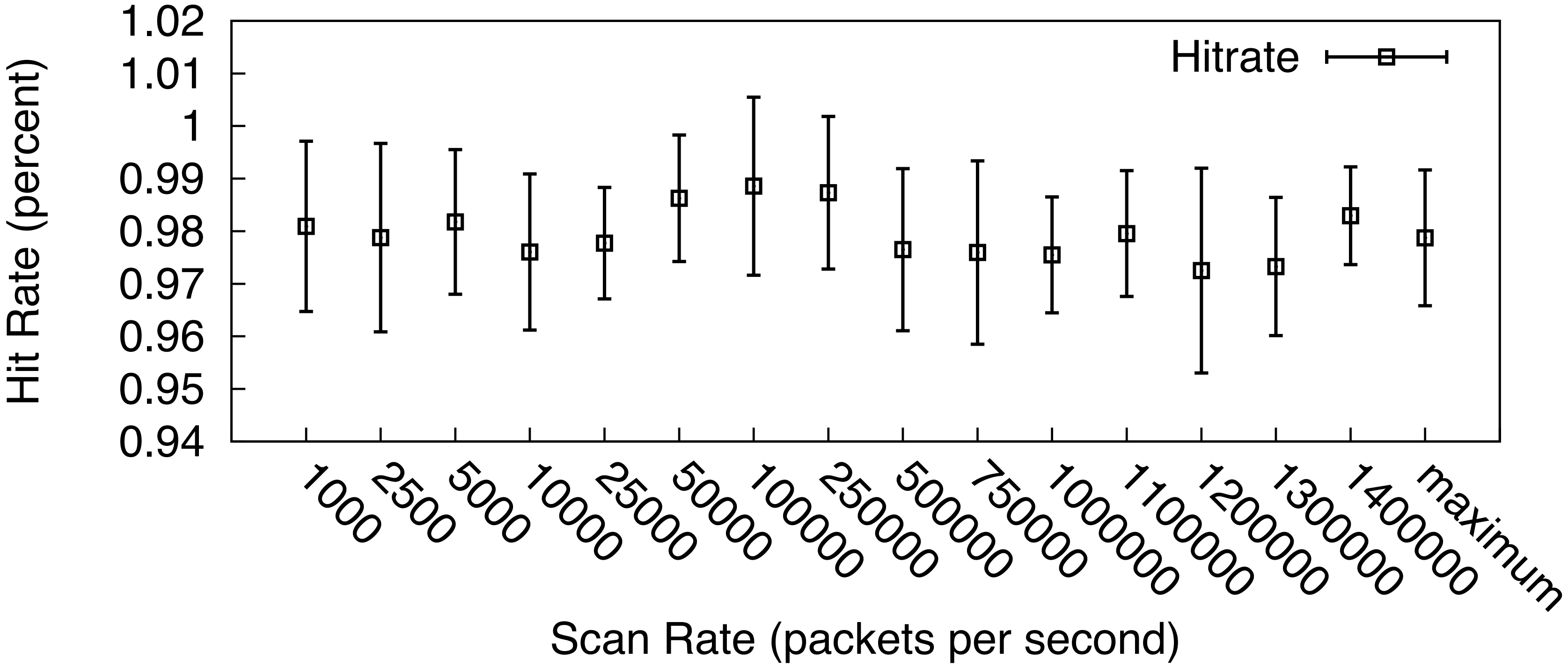
1. Globally Observable Weak Keys
2. Uncovering the CA Ecosystem

Scan Rate

How fast is too fast?

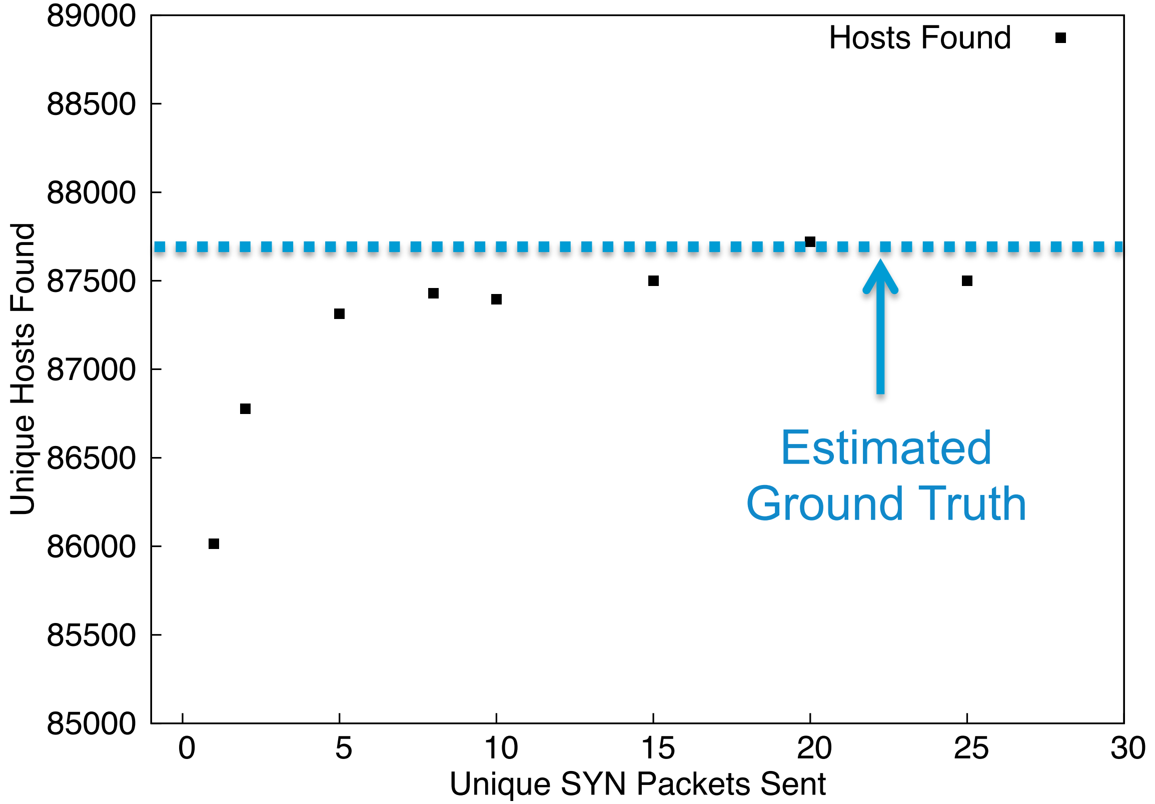
No correlation between hit-rate and scan-rate.

Slower scanning does not reveal additional hosts.



Coverage

Is one probe packet sufficient?



We expect an eventual plateau in responsive hosts, regardless of additional probes.

Scan Coverage

1 Packet:	97.9%
2 Packets:	98.8%
3 Packets:	99.4%

Comparison with Nmap

Averages for scanning 1 million random hosts

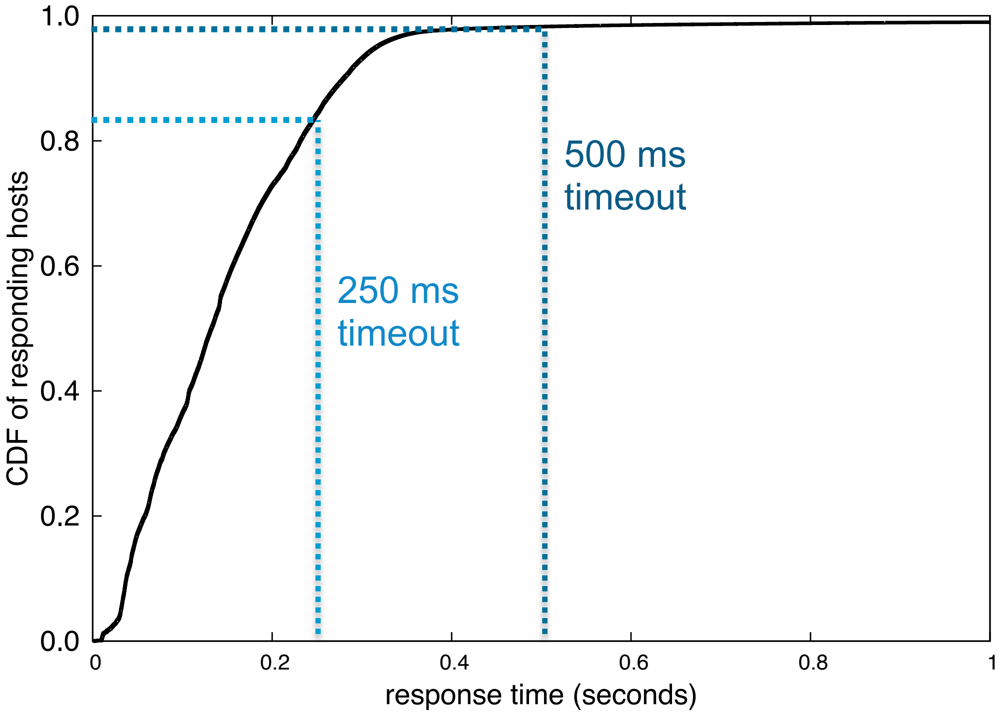
	Normalized Coverage	Duration (mm:ss)	Est. Internet Wide Scan
Nmap (1 probe)	81.4%	24:12	62.5 days
Nmap (2 probes)	97.8%	45:03	116.3 days
ZMap (1 probe)	98.7%	00:10	1:09:35
ZMap (2 probes)	100.0%	00:11	2:12:35

ZMap is capable of scanning 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%**
1.0 s:	**99.0%**
8.2 s:	**99.9%**

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

Talk Roadmap

ZMap Scanner

1. Philosophy and Architecture of ZMap
2. Characterizing ZMap's Performance

Applications of High Speed Scanning

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

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

We identified 86% of live allocated bridges with a single scan

Tor has developed *obfsproxy* that listens on random ports to count this type of attack

ZMap Applications

Potential Applications

Detect Service Disruptions

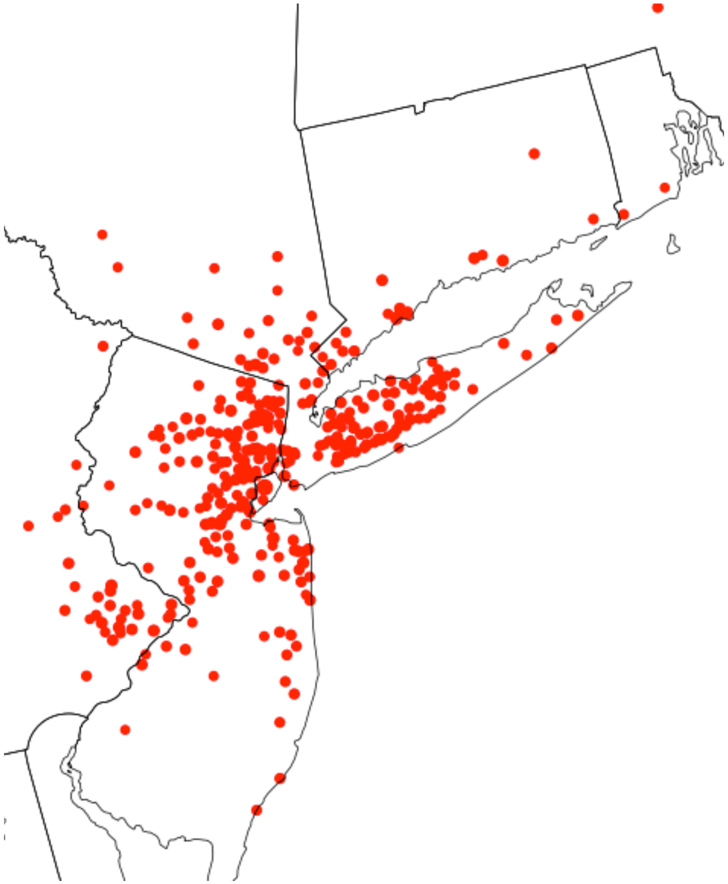
Track Adoption of Defenses

Study Criminal Behavior

Security Implications

Anonymous Communication

Track users between IP leases



Snapshot of HTTPS outages
caused by Hurricane Sandy

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	6.241	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

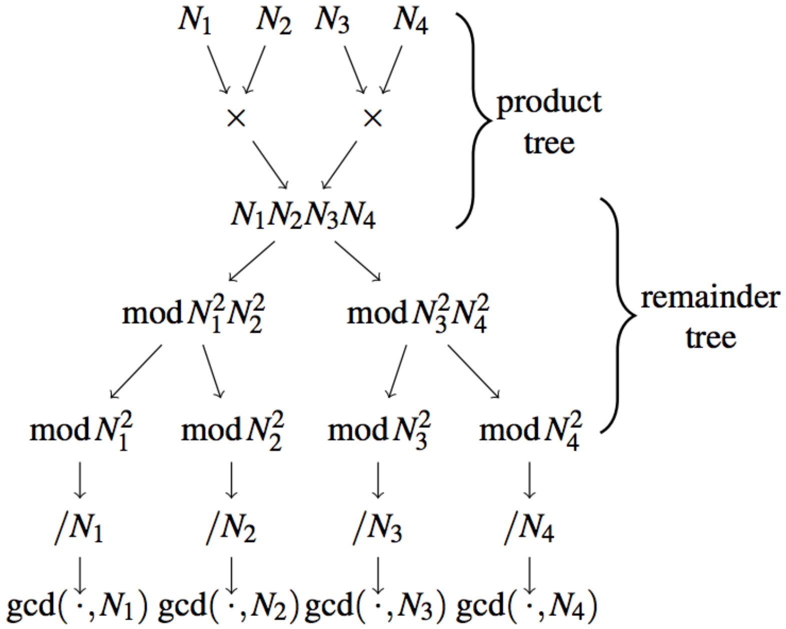
Most efficient known method of compromising an RSA key is to factor *n* back to *p* and *q*

While *n* is difficult to factor, for

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

we can trivially compute

$p = GCD(N_1, N_2)$


N_1 N_2 N_3 N_4
×
×
$N_1N_2N_3N_4$
product tree
$\bmod N_1^2N_2^2$
$\bmod N_3^2N_4^2$
remainder tree
$\bmod N_1^2$ $\bmod N_2^2$ $\bmod N_3^2$ $\bmod N_4^2$
$/N_1$ $/N_2$ $/N_3$ $/N_4$
$\gcd(\cdot, N_1)$ $\gcd(\cdot, N_2)$ $\gcd(\cdot, N_3)$ $\gcd(\cdot, N_4)$

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

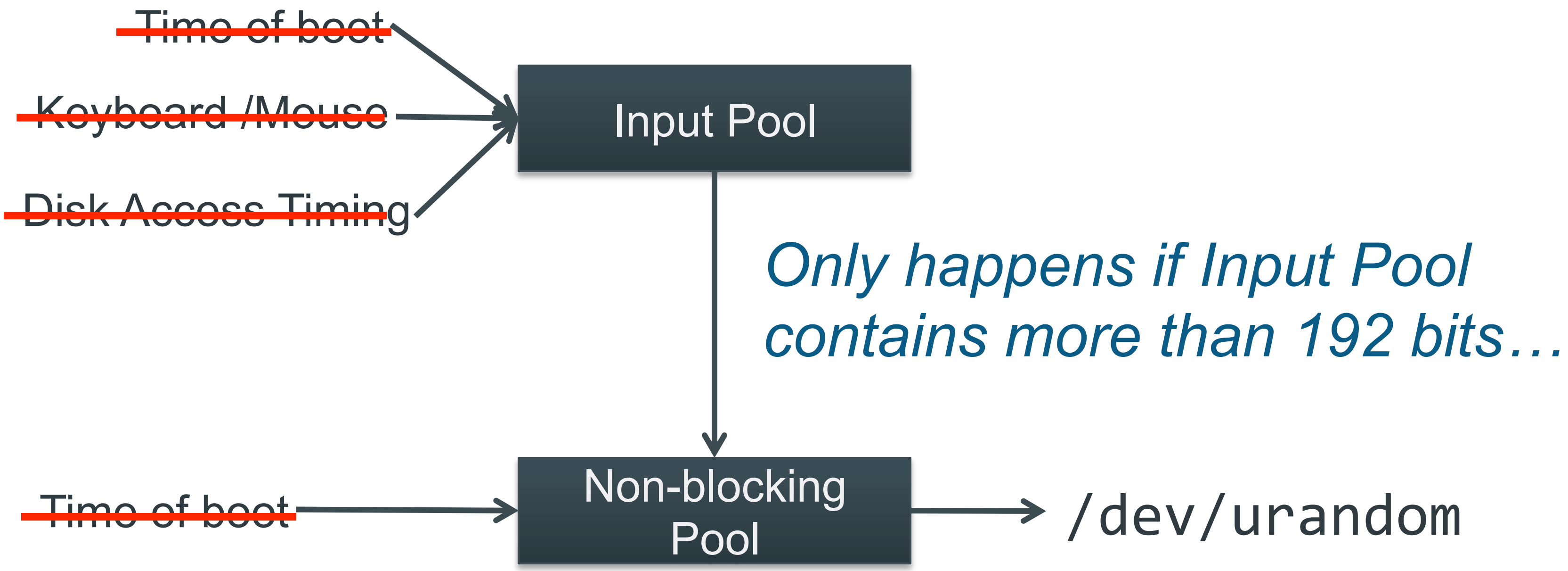
Compromised keys are generated by headless or embedded network devices

Identified devices from > 40 manufacturers

Linux /dev/urandom

Why are embedded systems generating broken keys?

Nearly everything uses /dev/urandom



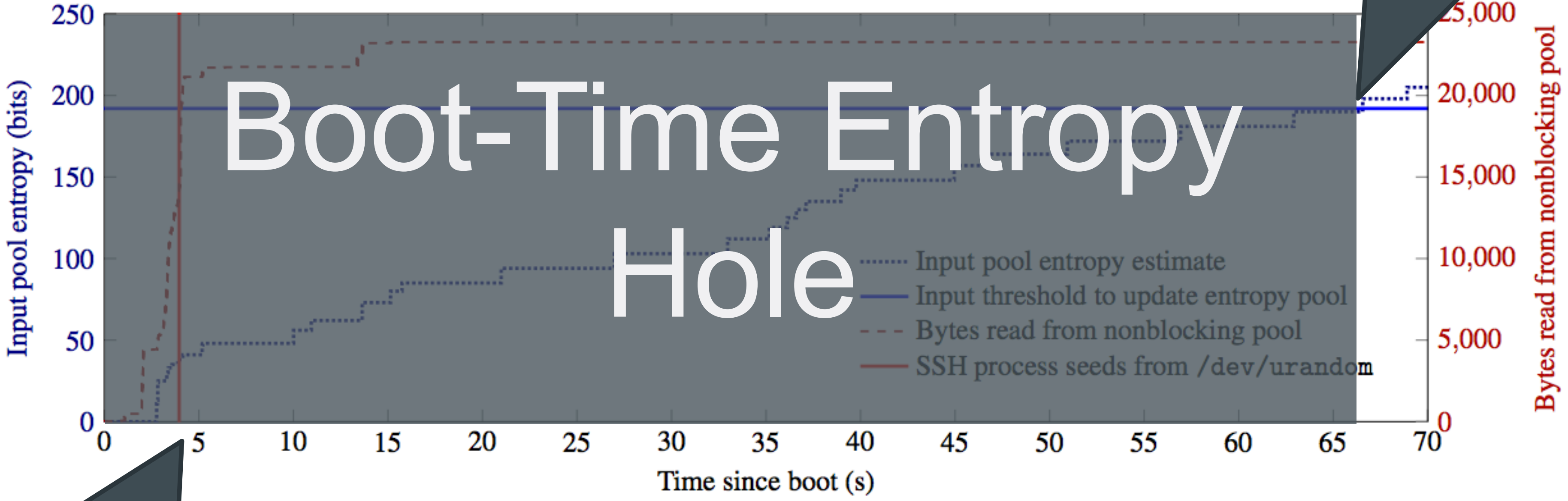
Only happens if Input Pool contains more than 192 bits…

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?



Boot-Time Entropy Hole

Entropy first mixed into /dev/urandom

OpenSSH seeds from /dev/urandom

/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

Talk Roadmap

ZMap Scanner

1. Philosophy and Architecture of ZMap
2. Characterizing ZMap's Performance

Applications of High Speed Scanning

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

Certificate Authority Ecosystem

Nearly all secure web communication uses HTTPS

- online banking, e-commerce, e-mail, etc…

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

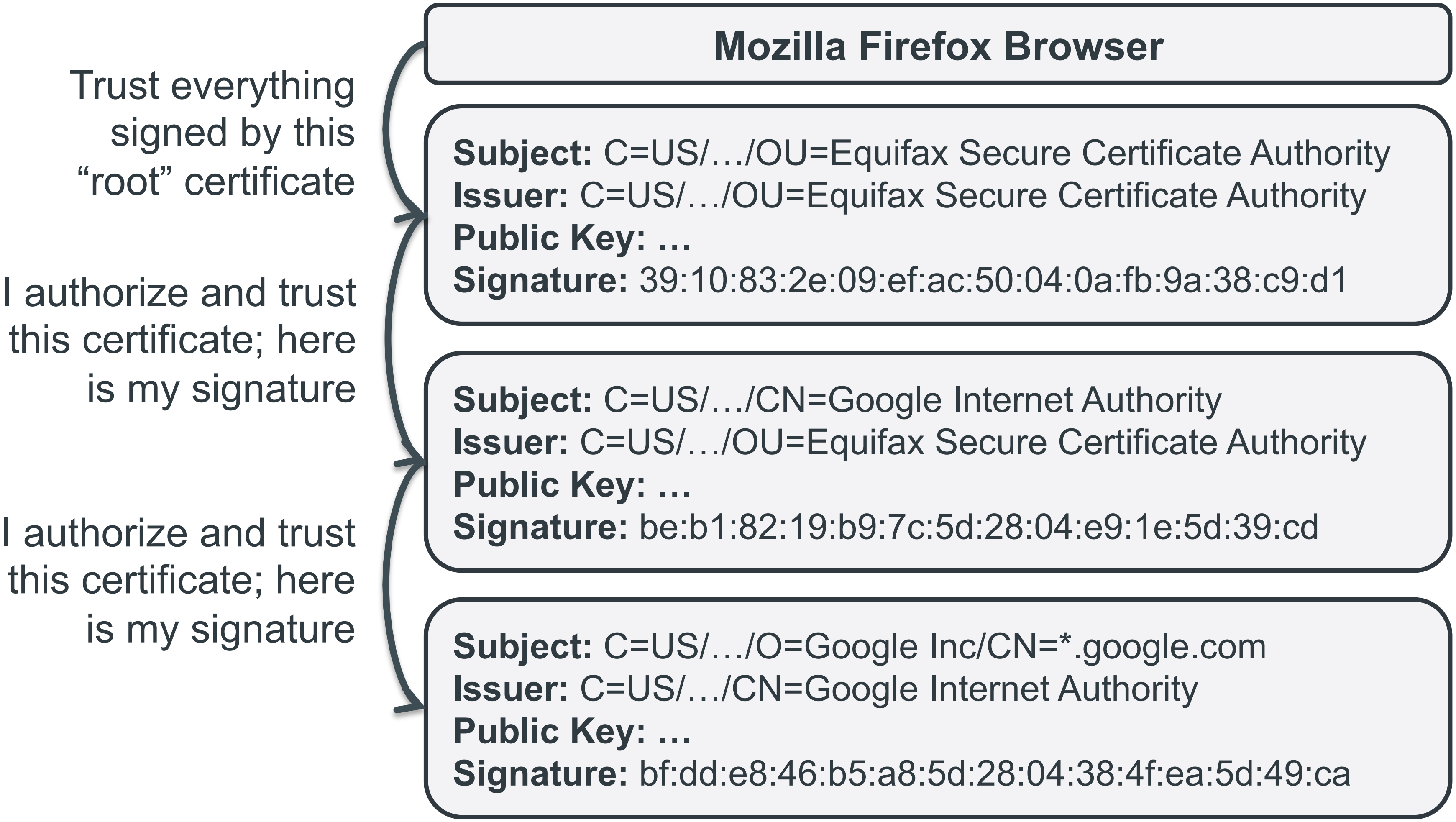
Every certificate authority can sign for *any* website

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



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

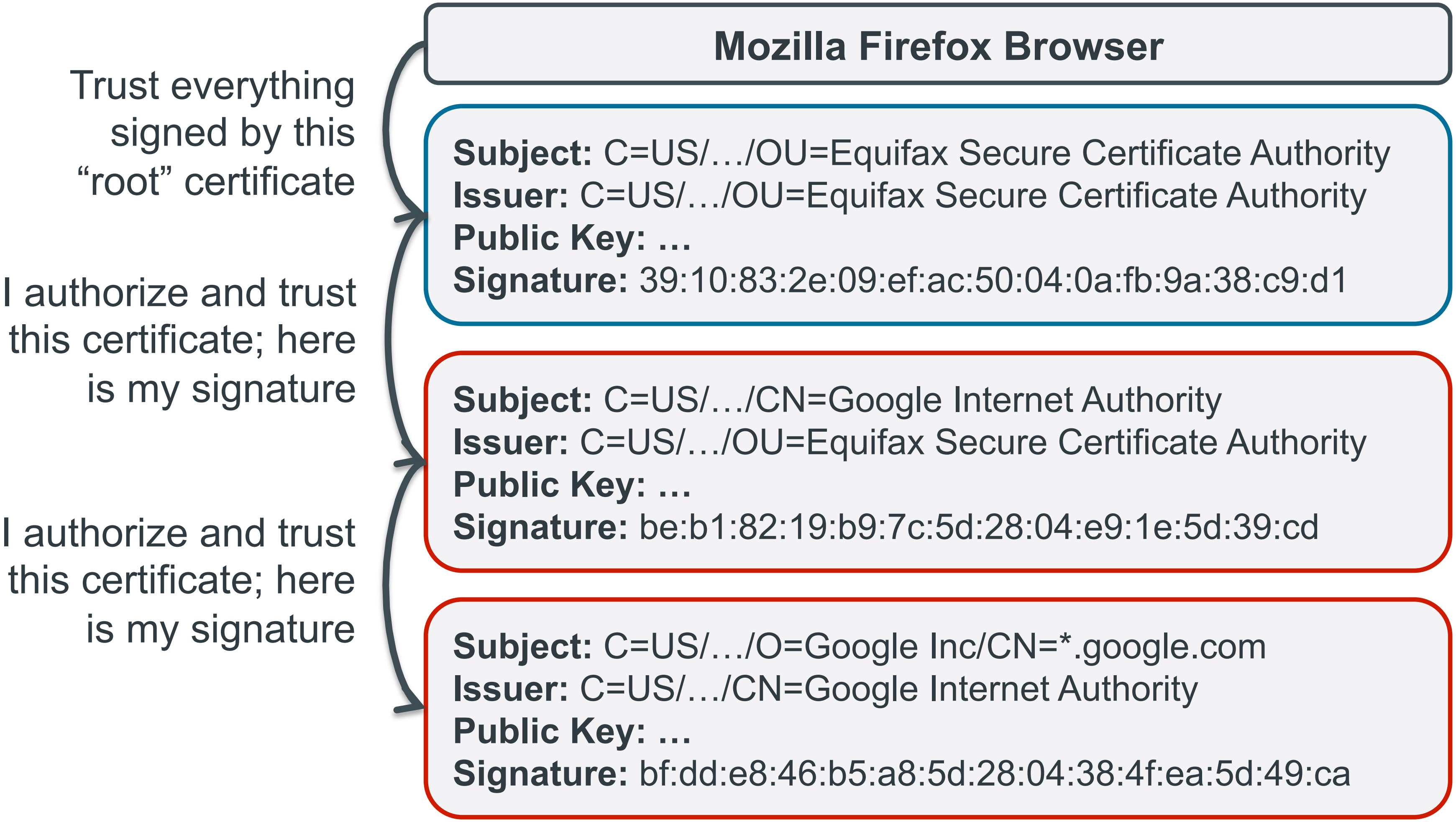
Subject: C=US/…/CN=Google Internet Authority
Issuer: C=US/…/OU=Equifax Secure Certificate Authority
Public Key: …
Signature: be:b1:82:19:b9:7c:5d:28:04:e9:1e:5d:39:cd

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

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

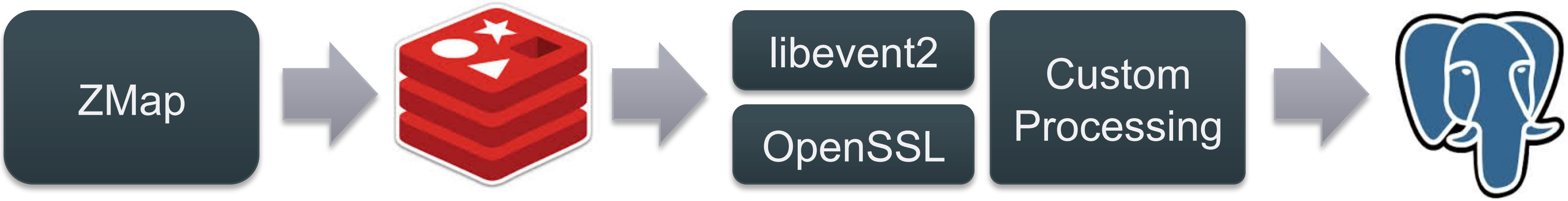
Subject: C=US/…/O=Google Inc/CN=*.google.com
Issuer: C=US/…/CN=Google Internet Authority
Public Key: …
Signature: bf:dd:e8:46:b5:a8:5d:28:04:38:4f:ea:5d:49:ca

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. Identity 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

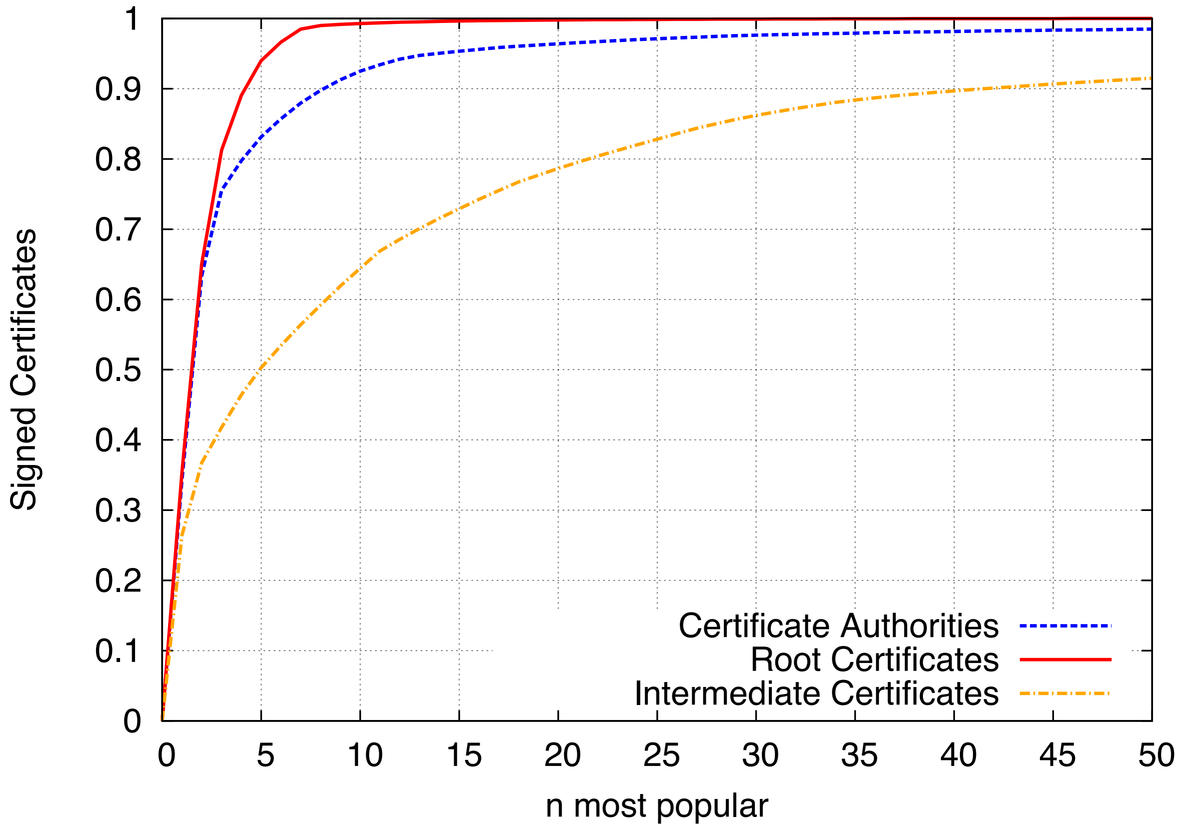
- Including religious institutions, libraries, non-profits, financial institutions, governments, and hospitals
- More than 80% of organizations controlling a CA certificate aren't commercial certificate authorities

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!

Worrisome Observations

What are authorities doing that puts the ecosystem at risk?

1. CAs are ignoring foundational principles such as *defense in depth* and the principle of least privilege

2. CAs are offering services that put the ecosystem as a whole at risk

3. CAs are failing to recognize cryptographic reality

4. Correctly deploying HTTPS remains difficult

Ignoring Foundational Principles

What are authorities doing that puts the ecosystem at risk?

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

2. 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

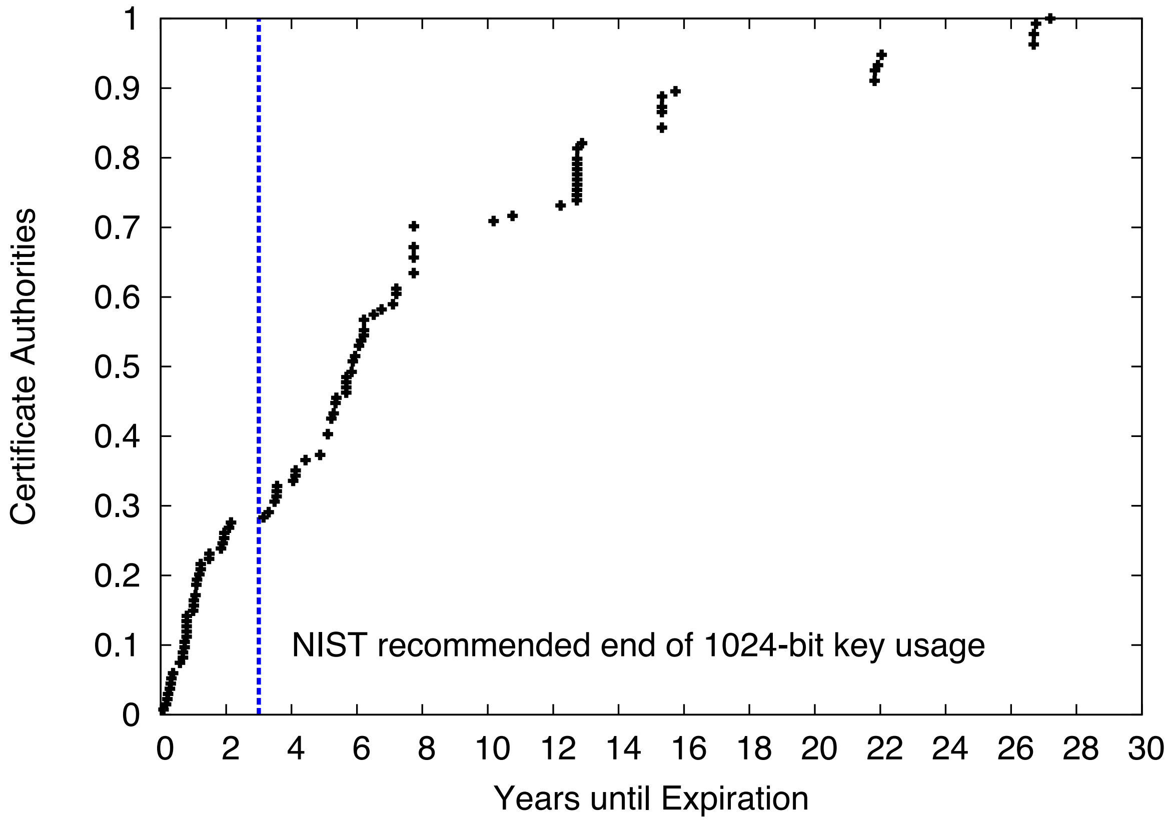
3. Lack of constraints allowed a rogue CA certificate in 2012, but in another case prevented 1,400 invalid certificates

4. 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

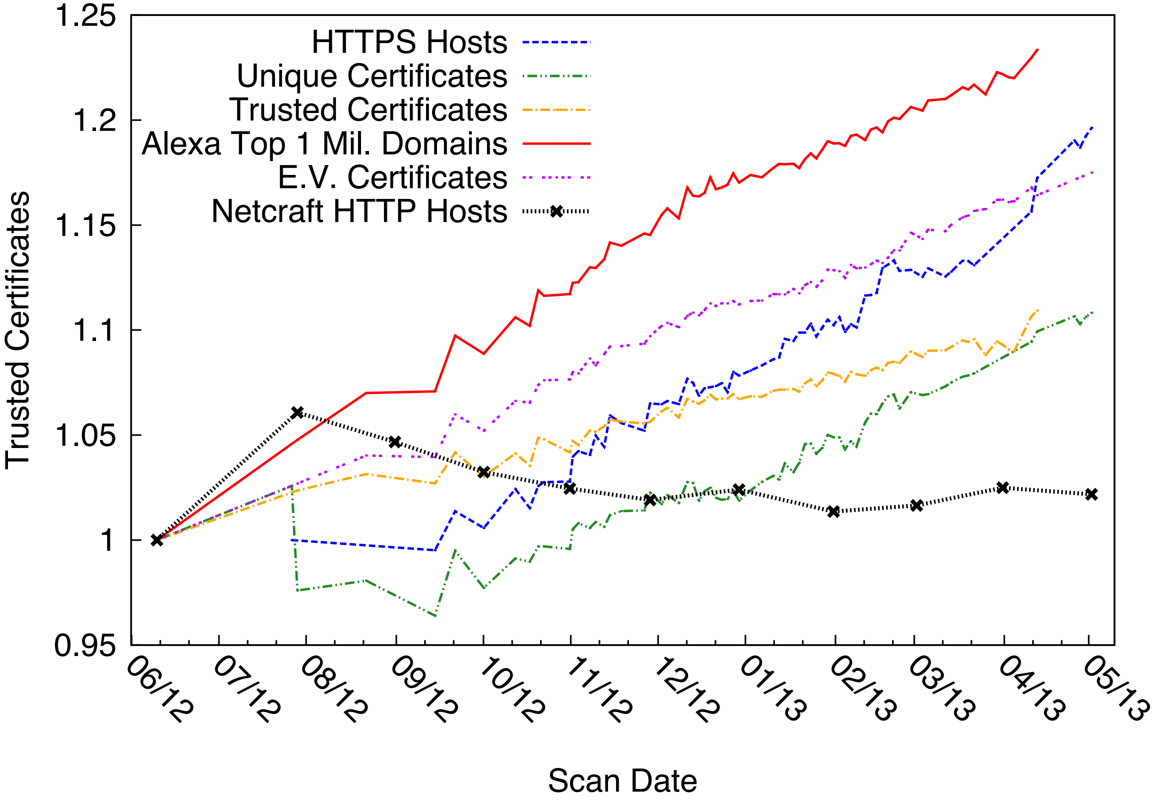
50% of certificates are rooted in a 1024-bit key

More than 70% of these will expire after 2016

Still signing certificates using MD5!

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.

Future Work

10gigE Network Surveys

TLS Server Name Indication

Scanning Exclusion Standards

IPv6 Scanning Methdology?

Use ZMap to do great research!

ZMap Public Release

Releasing ZMap as a fully documented open source project

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

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

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

Be sure you have adequate bandwidth and be a good Internet neighbor!

Scans.IO Data Repository

How do we share all this scan data?

University of Michigan is hosting a repository of data gathered from Internet-wide scans

https://scans.io

Includes our HTTPS datasets and data from Rapid7

Working with other organizations to post data

Conclusion

Living in a unique period

IPv4 can be quickly, exhaustively scanned

IPv6 has not yet been widely deployed

ZMap lowers barriers of 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

Ultimately hope that ZMap enables future research

ZMap | Fast Internet-Wide Scanning, Weak Keys and the HTTPS Certificate Ecosystem

https://zmap.io

Zakir Durumeric, Michael Bailey

University of Michigan