

Attacker Intelligence Quiz

Attackers have three phases of intelligence gathering.
Match each phase to its description:

- B Footprinting (FP)
- A Scanning (S)
- C Enumeration (E)

- A. The attacker uses the internet to obtain information on specific IP addresses. The kind of information gathered is: O.S., services, and architecture of the target system
- B. The attacker gathers information about a target. The kind of information gathered is: DNS, email servers, and the IP address range.
- C. The attacker gathers information on network user and group names, routing tables, and simple network management protocol.



Internet Wide Security Scanning

- Expose new vulnerabilities
- Track adoption of defensive mechanisms
- Probing the **ENTIRE** address space with existing tools is both difficult and slow

E.g., Datasets of cryptographic keys + the entire address space =
Detection of vulnerabilities and understanding of the ecosystem



Internet-Wide Network Studies

- Mini-lead web in
- devices (2
- = SServato
- pse at
- ystem (2
- and Sur the
- Visi

- Vulnerabilities in: 5% HTTPS hosts and 10% SSH hosts
- 25 hours across 25 Amazon EC2 Instances (625 CPU-hours)
- 3 months on 3 Linux desktop machines (6500 CPU-hours)
- 3 months to complete ICMP census (2200 CPU-hours)



Zmap

- What if Internet surveys didn't require heroic efforts?
- What if we could scan the HTTPS ecosystem every day?
- What if we wrote a whole-Internet scanner from scratch?



- Open Source Tool
- Can port scan the entire IPv4 address space
- Can scan 45 minutes on a gigabit network at 97% the speed of gigabit Ethernet with 98% coverage



Zmap

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	ZMap
Reduce state by scanning in batches <ul style="list-style-type: none">• Time lost due to blocking• Results lost due to timeouts	Eliminate local per-connection state <ul style="list-style-type: none">• Full asynchronous components• No blocking except for network
Track individual hosts and retransmit <ul style="list-style-type: none">• Most hosts will not respond	Shotgun Scanning Approach <ul style="list-style-type: none">• Always send n probes per host
Avoid flooding through timing <ul style="list-style-type: none">• Time lost waiting	Scan widely dispersed targets <ul style="list-style-type: none">• Send as fast as network allows
Utilize existing OS network stack <ul style="list-style-type: none">• Not optimized for immense number of connections	Probe-optimized Network Stack <ul style="list-style-type: none">• Bypass inefficiencies by generating Ethernet frames



Addressing Probes

How do we randomly scan addresses without excessive state?

~~Overloading destination~~

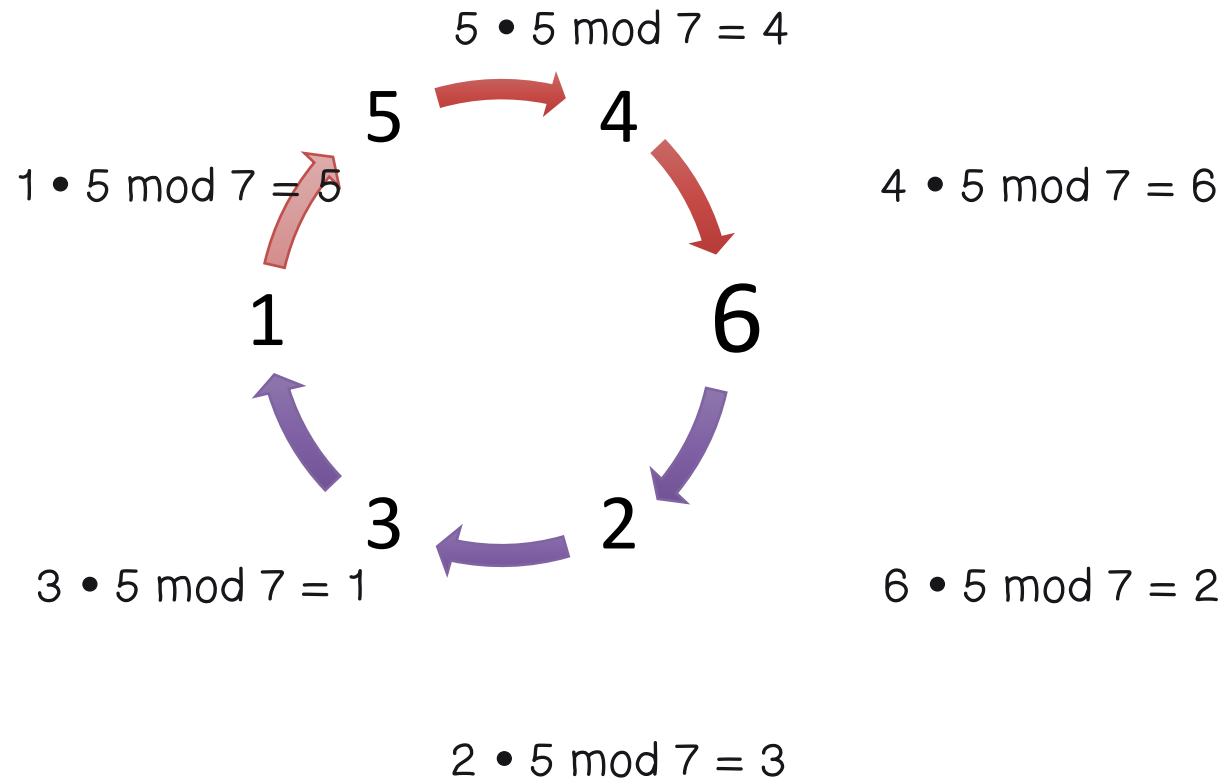
~~Track scanned hosts~~

Scan hosts according to random permutation

- Iterate over multiplicative group of integers modulo p , a prime slightly larger than 2^{32}



Addressing Probes



A simple example:

1. Primitive root or generator: 5
2. Starting number: 2
3. Multiply current number by 5 modulo 7 results in the next number; all number are enumerated/iterated



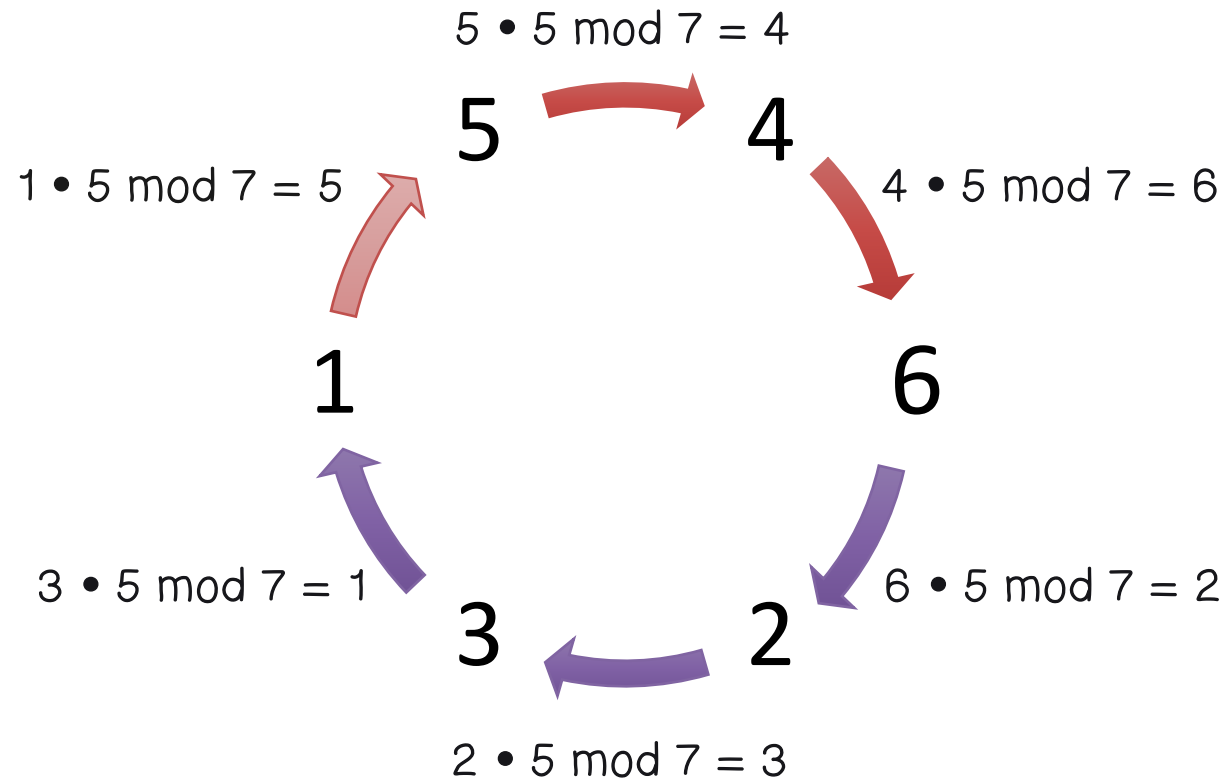
Addressing Probes

How do we randomly scan addresses without excessive state?

- Select a fresh random permutation of the address space for each scan
- Generate a primitive root, or, a generator, of the multiplicative group
- Choose a random starting address

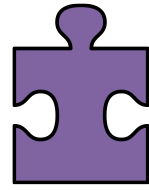


Addressing Probes



Negligible State:

1. Primitive root
2. Current location
3. First address



TCP IP Quiz

Step 1: Which protocol is used to break data into packets?

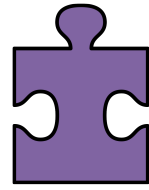
TCP

Step 2: Which protocol is used to move packets from router to router?

IP

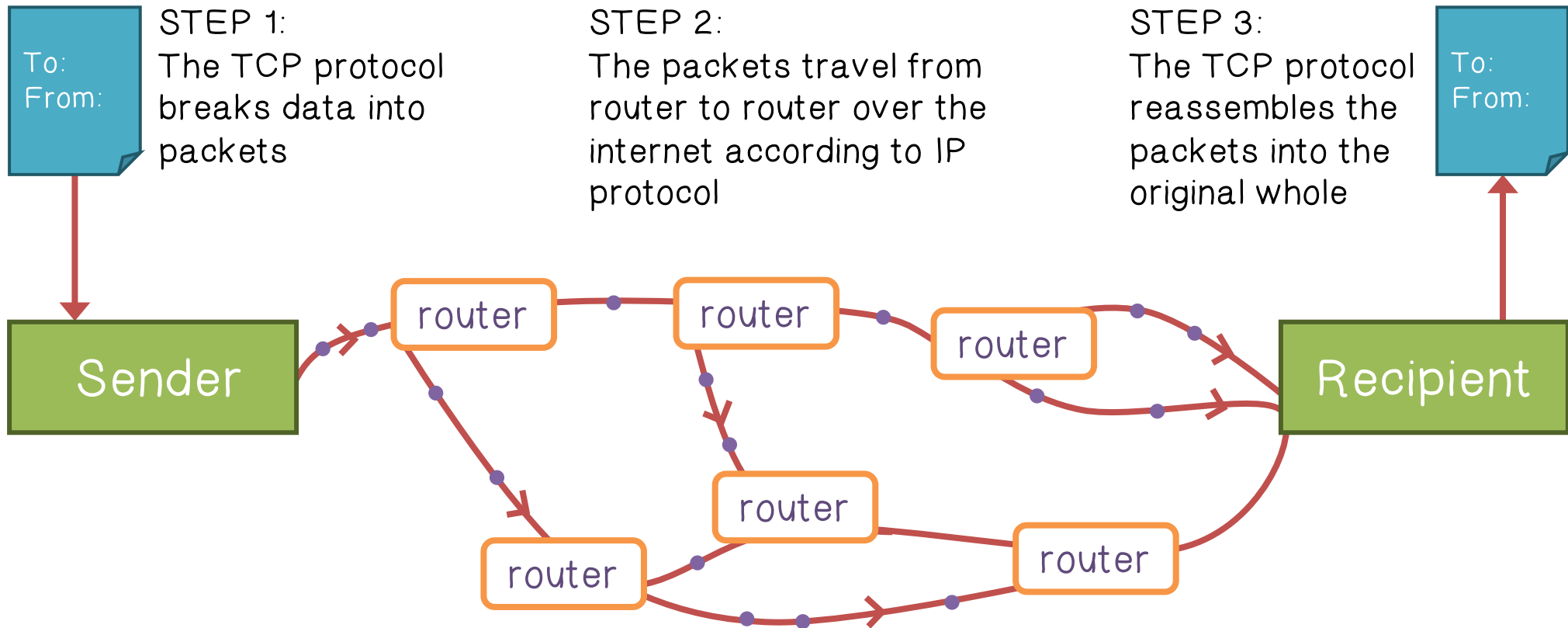
Step 3: Which protocol reassembles the data packets?

TCP



TCP IP Quiz

How TCP/IP Works

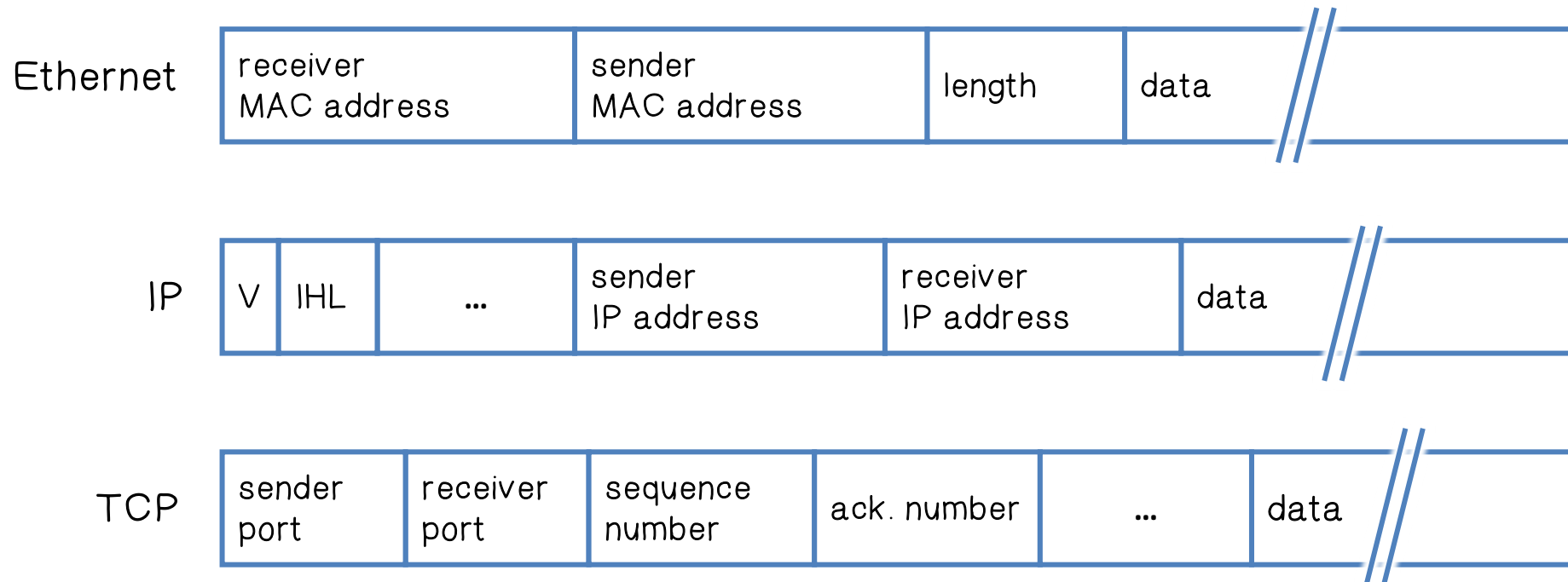




Validating Responses

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

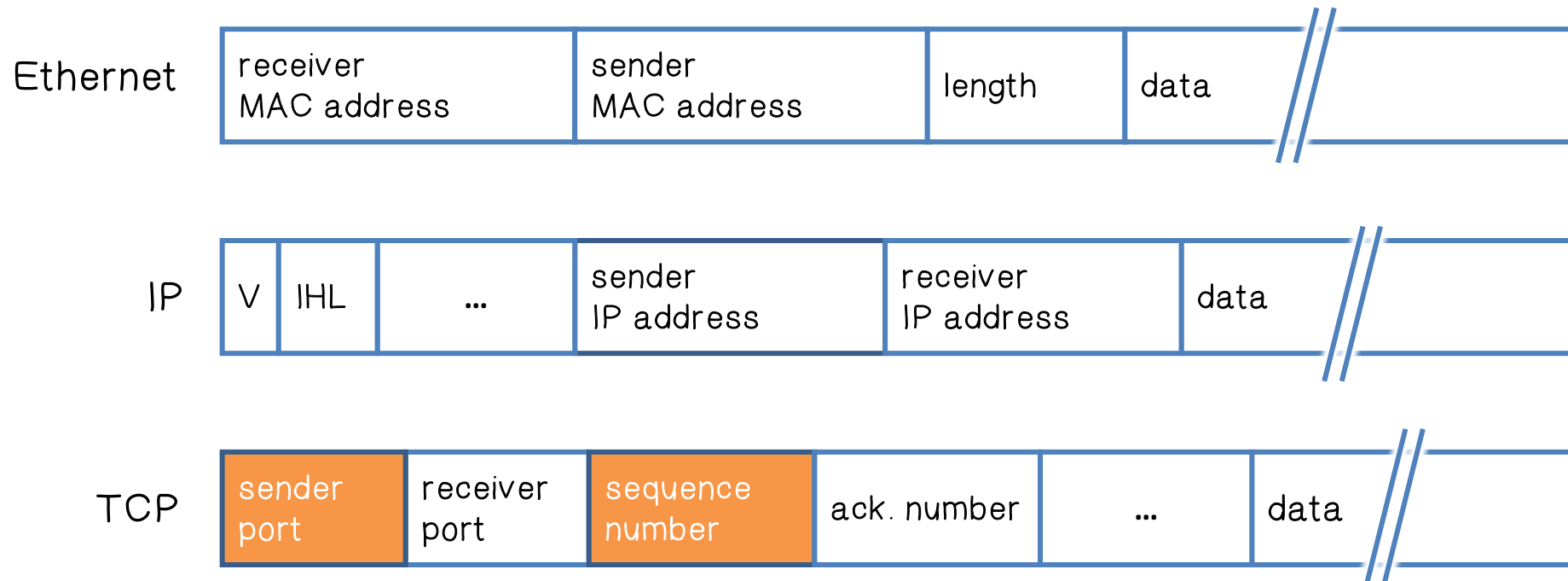




Validating Responses

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

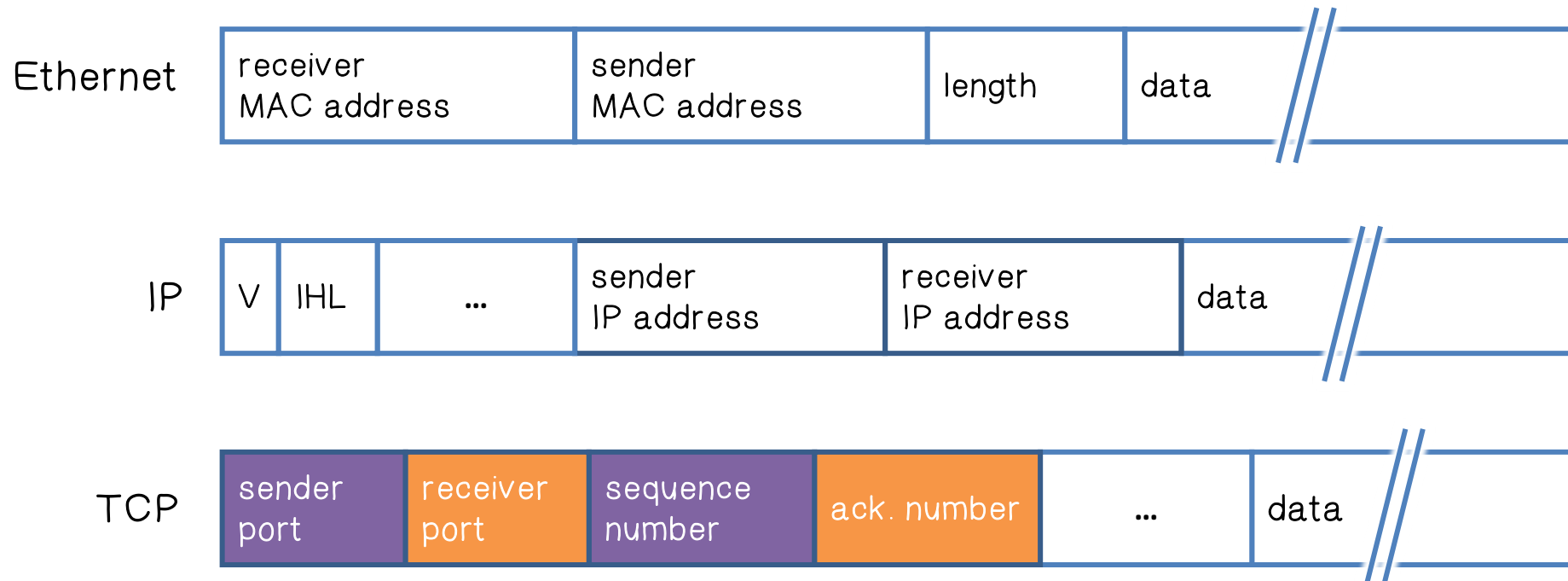




Validating Responses

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





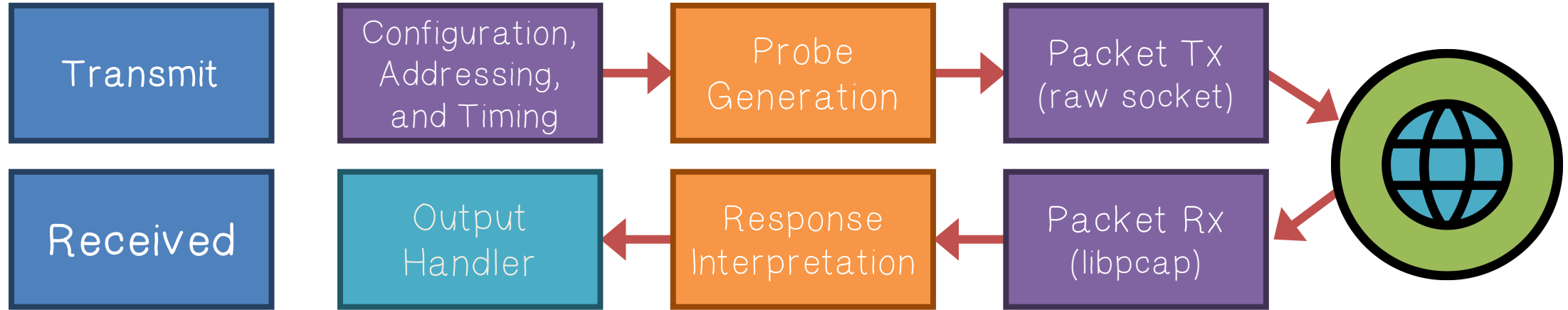
Packet Transmission and Receipt

How do we make processing probes easy and fast?

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



Packet Transmission and Receipt



1.4 million packets per second on a gigabit network

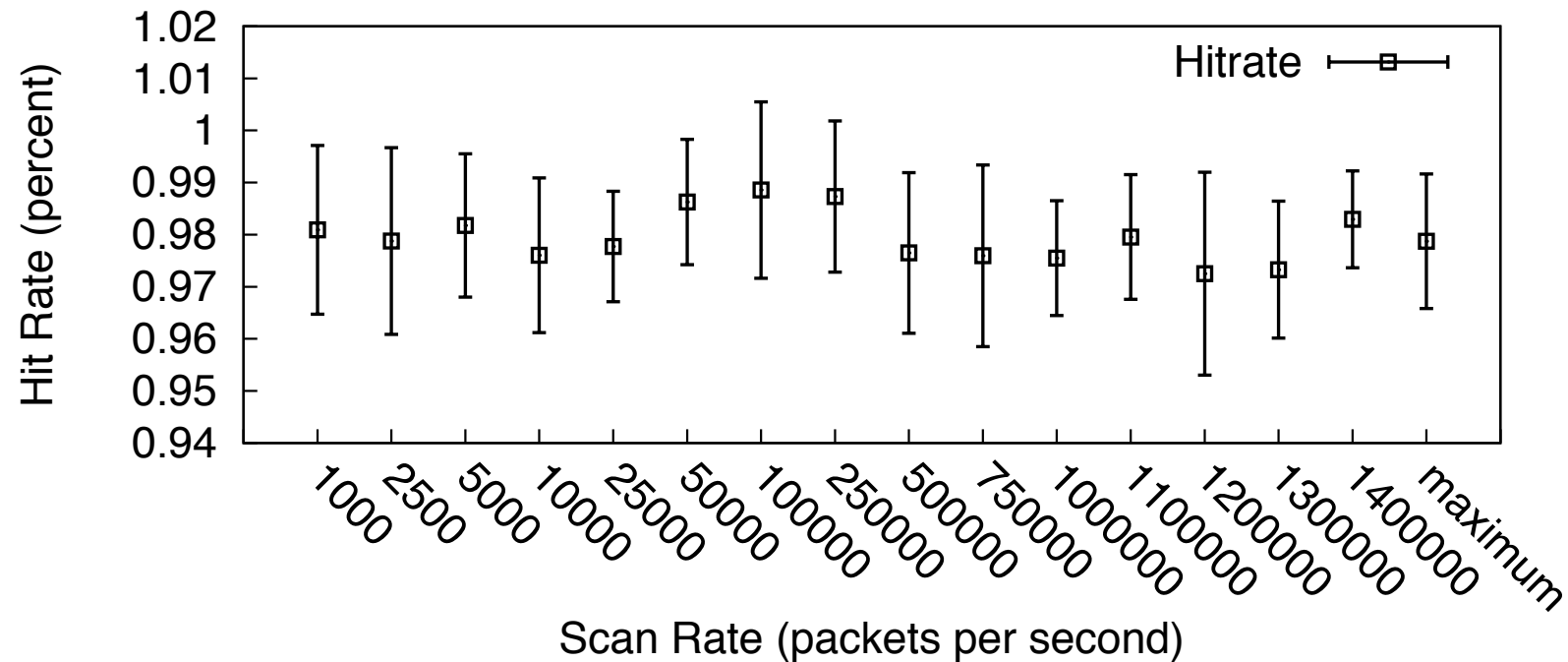
Zmap:

- An extensible framework
- Abstracts out configuration, timing, addressing, validation



Scan Rate

How fast is too fast?

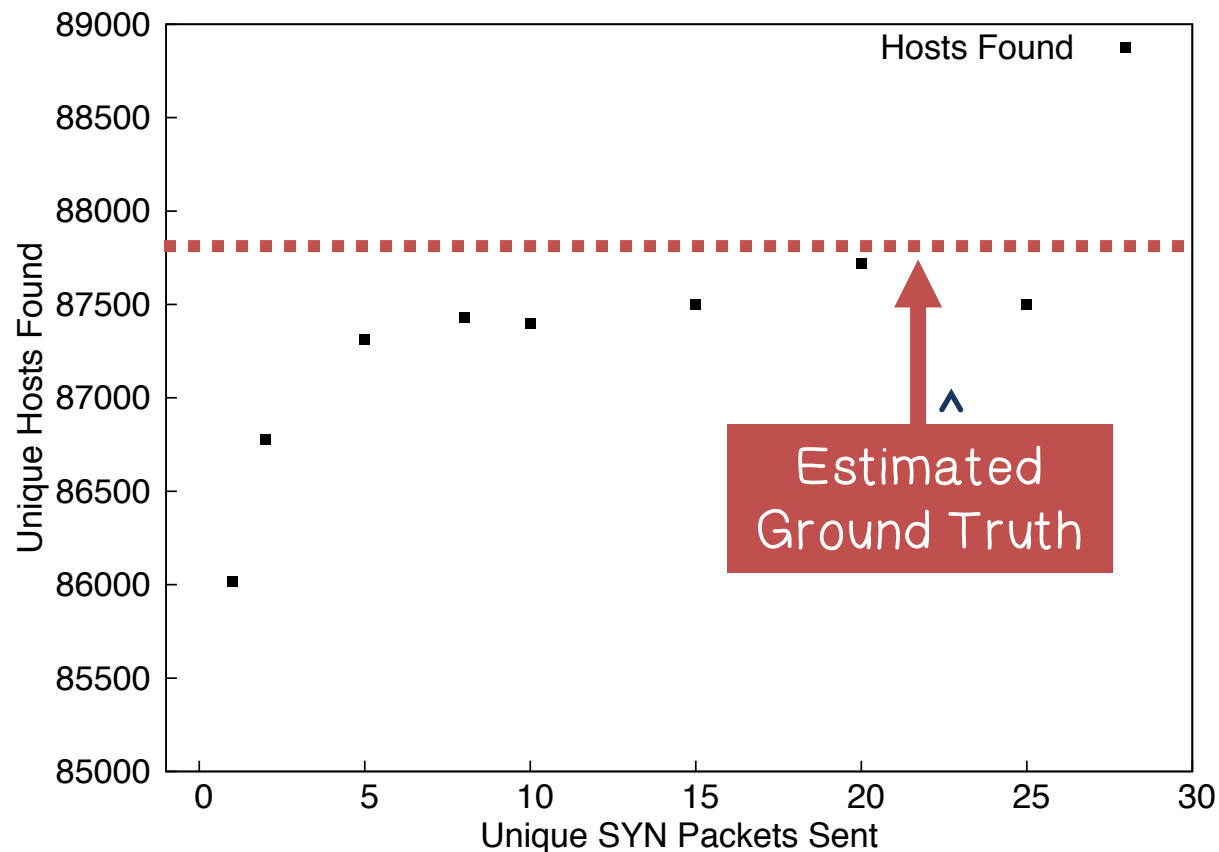


- No correlation between hit-rate and scan-rate.
- Slower scanning does not reveal additional hosts.



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

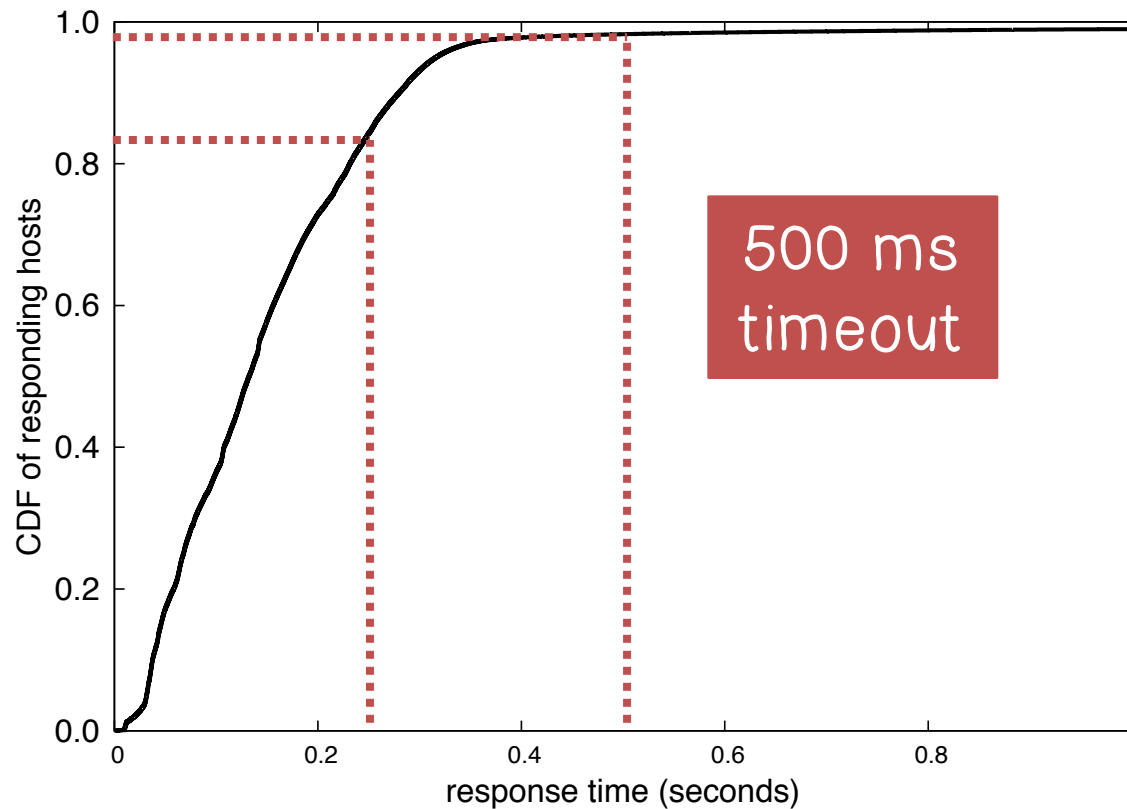
	Normalized Coverage	Duration (mm:ss)	Est. Internet Wide Scan
Nmap (1 probe)	81.4%	24:12	62.5 days
Nmap (2 probes)	97.8%	45:03	116.3 days
ZMap (1 probe)	98.7%	00:10	1:09:35
ZMap (2 probes)	100.0%	00:11	2:12:35

- 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.

Entropy Quiz

Fill in the blanks with the correct answers:

With regards to computing, what is entropy?

Randomness for use in cryptography or other applications that require random data.

What are the two sources of entropy?

Hardware sources and randomness generators

A lack of entropy will have a **negative** impact on performance and security.



Cryptographic Keys

Uncovering weak cryptographic keys and poor entropy collection

	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



Cryptographic Keys

Why are a large number of hosts sharing cryptographic keys?

5.6% of TLS hosts and 9.6% of SSH hosts share keys in a vulnerable manner

- Default certificates and keys
- Apparent entropy problems



Cryptographic Keys

Scanning at this frequency allows tracking high-profile certificates, signing certificates and identify cases of mis-issued CA certificates

- A signing certificate was accidentally issued to a Turkish Transit Provider
- We found 1,300 CA certificates that were issued by the Korean Government



Factoring RSA Public Keys

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

RSA Public Key: $n = p \bullet q$, 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

$$N_1 = p \bullet q_1 \text{ and } N_2 = p \bullet q_2$$

we can trivially compute

$$p = \text{GCD}(N_1, N_2)$$



Factoring RSA Public 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





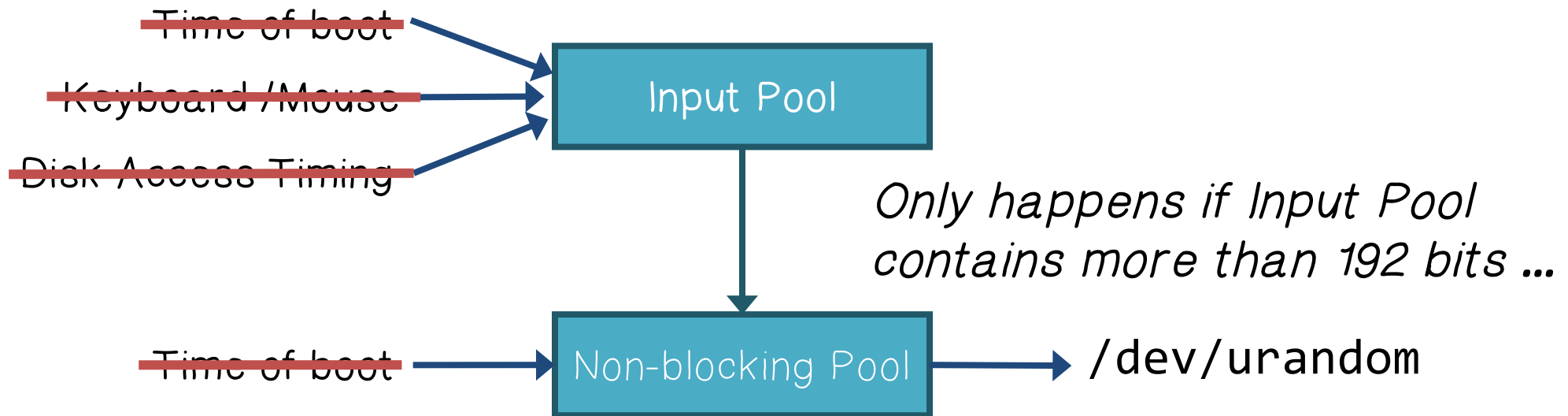
Embedded Systems

Linux /dev/urandom

Nearly everything uses /dev/urandom

Problem 1: Embedded devices may lack all these sources

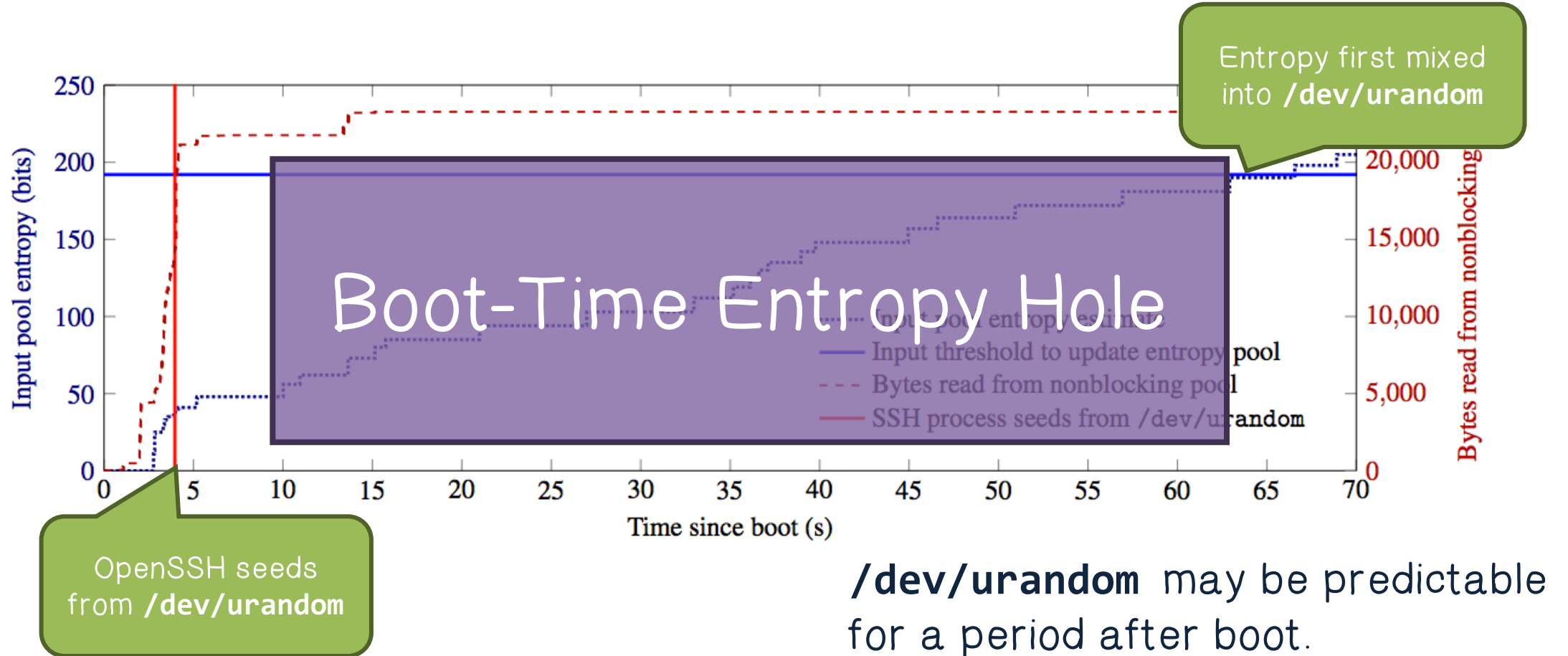
Problem 2: /dev/urandom can take a long time to “warm up”

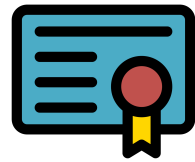




Embedded Systems

Why are embedded systems generating broken keys?





Certificate Authorities



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

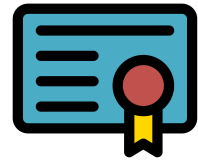


Every certificate authority can sign for *any* website



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

Trust everything signed by this "root" certificate

I authorize and trust this certificate; here is my signature

I authorize and trust this certificate; here is my signature

Mozilla Firefox Browser

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

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

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

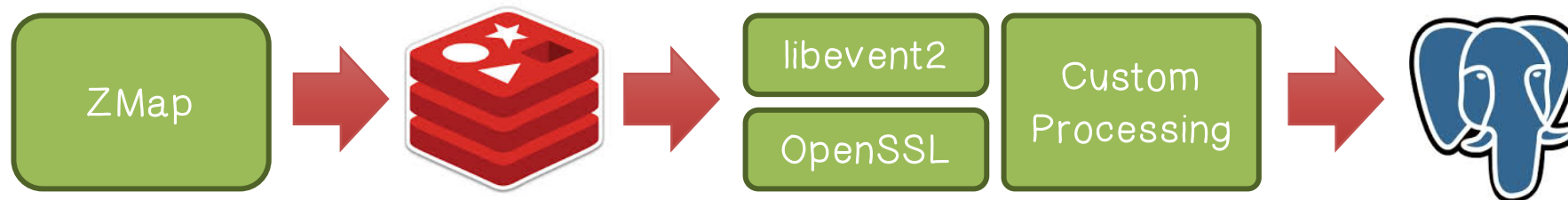


Uncovering the HTTPS Ecosystem

How do we regularly collect certificates from Internet?

>200 scans of the ecosystem in one year

- Identity certificate authorities
- Uncover worrisome practices



Collected 42 million unique certificates from 109 million unique hosts in one year

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

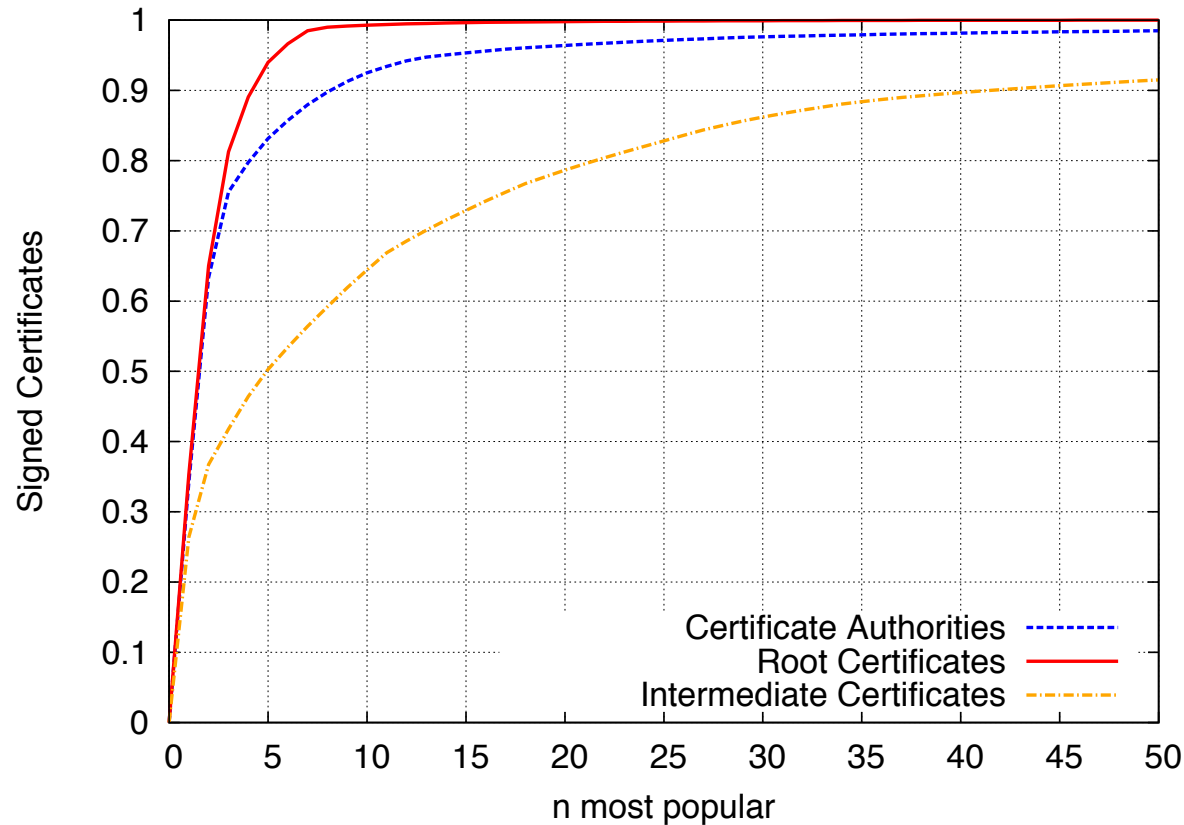
More than half of the certificates were provided by the German National Research and Education Network (DFN)

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



Identifying Certificate Authorities

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!

CA Risks

Worrisome Observations

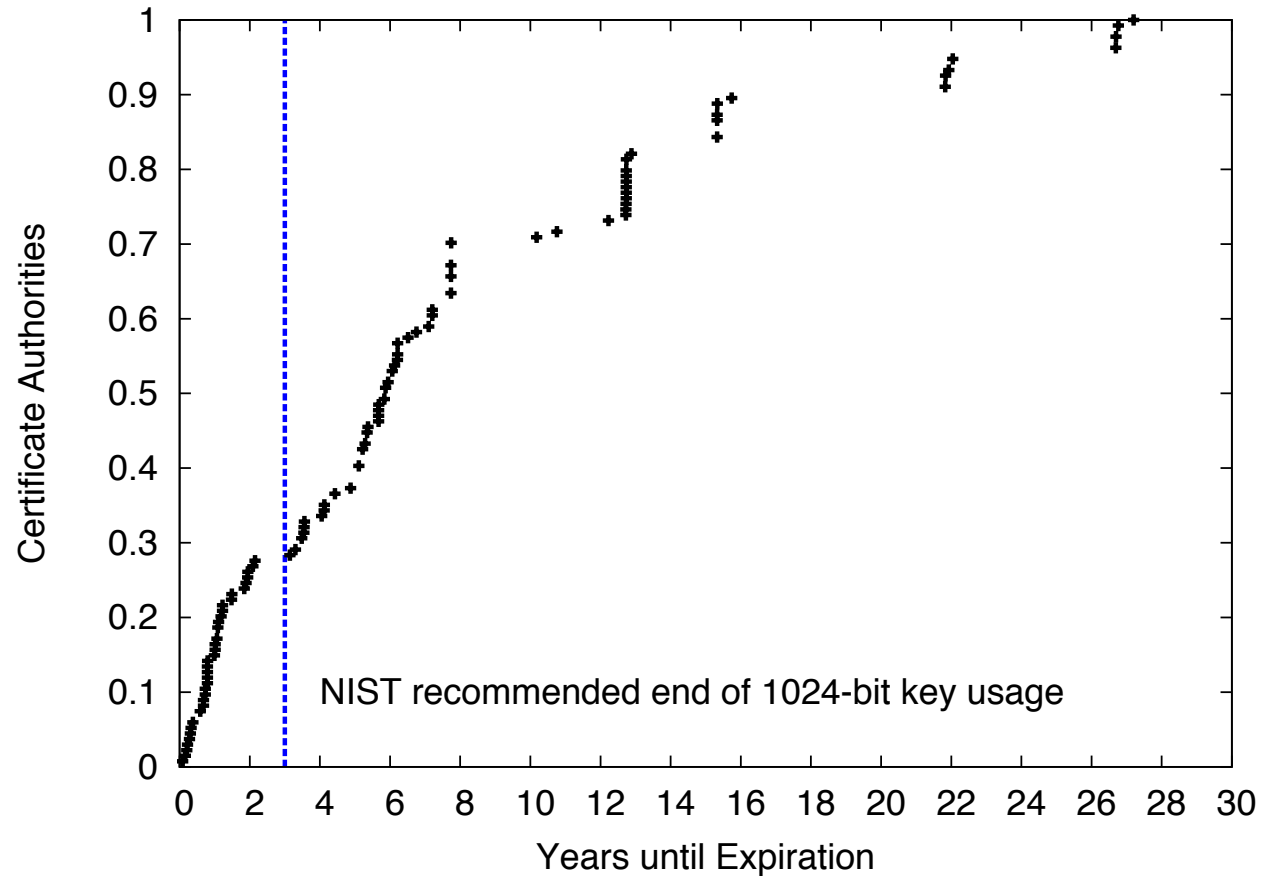
- CAs are ignoring foundational principles such as defense in depth and the principle of least privilege
- CAs are offering services that put the ecosystem as a whole at risk
- CAs are failing to recognize cryptographic reality
- Correctly deploying HTTPS remains difficult

CA Risks

Ignoring Foundational Principles

- 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

⚠ CA Risks

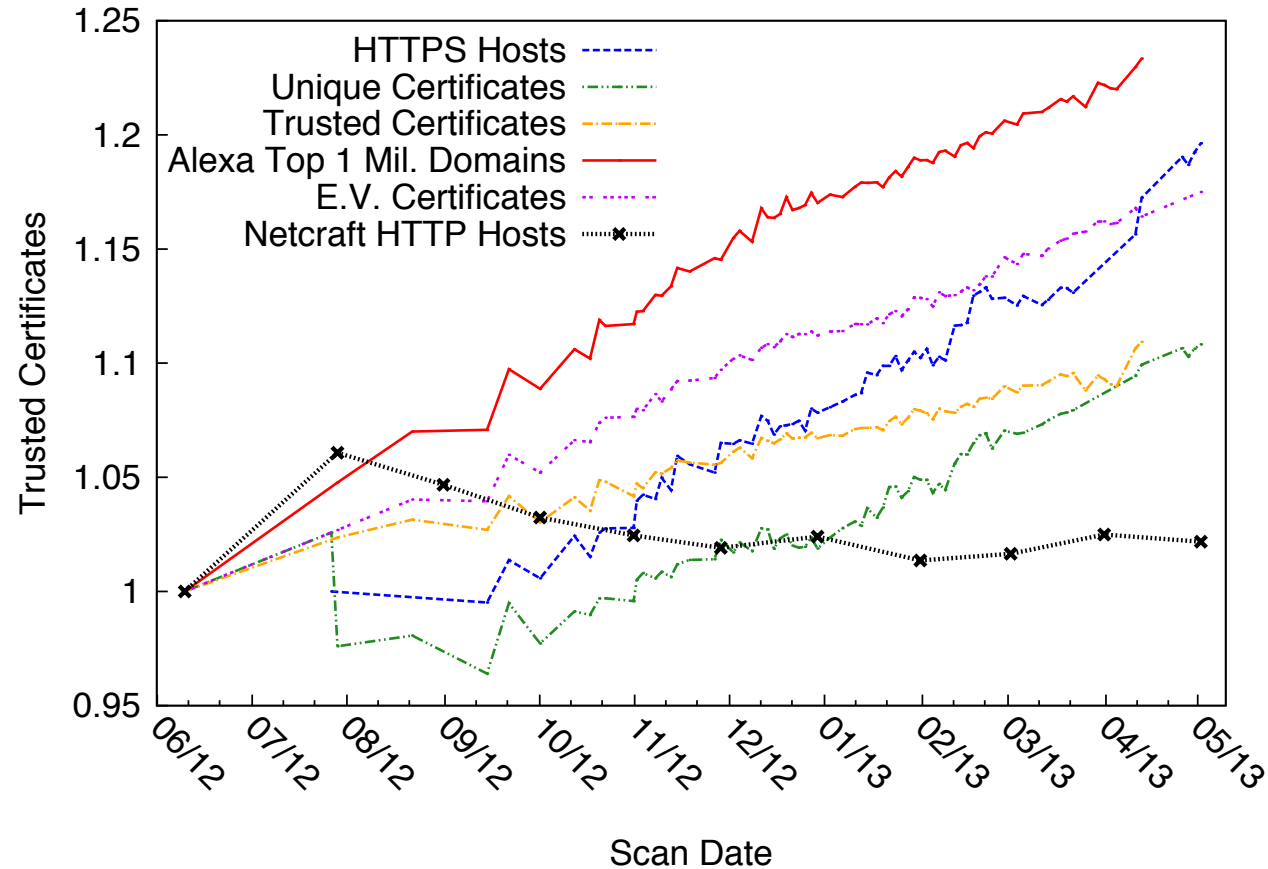


Cryptographic Reality




- 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



June 2012–May 2013

- 10%  HTTPS servers
- 23%  Use on Alexa Top-1M sites
- 11%  Browser-trusted certificates



ZMap Open Source

Releasing ZMap as a fully documented open source project

Downloaded it from <https://zmap.io>

Scanning the Internet *really* is as simple as:

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

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>