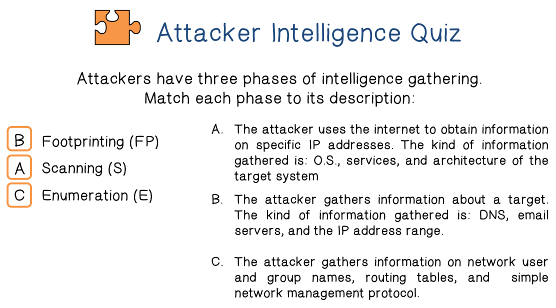
The recommended reading for this lesson is:
Zakir Durumeric, Eric Wustrow, and J. Alex Halderamn. ZMap: Fast Internet-Wide Scanning and its Security Applications.
In Proceedings of the 22nd USENIX Security Symposium

You can find short summary of the paper at the end of the document.

The Internet is a large ecosystem of networks. It is also a large ecosystem of vulnerabilities. To defend the network, we need to know it. In other words, we need to map the entire Internet. In this lesson, we will learn how to map the Internet efficiently and thoroughly, and survey vulnerabilities.

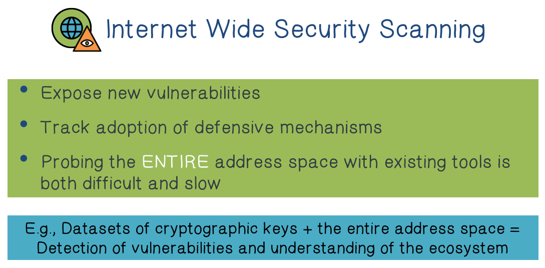


Attacker Intelligence Quiz

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

B Footprinting (FP)	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
A Scanning (S)	B. The attacker gathers information about a target. The kind of information gathered is: DNS, email servers, and the IP address range.
C Enumeration (E)	C. The attacker gathers information on network user and group names, routing tables, and simple network management protocol.

Let us start with a quiz on how attackers can gather intelligence. Match each phrase with its description.
In Footprinting, the attacker gathers information about target. The kind of information gathered is DNS, email servers, and the IP address range. In Scanning, the attacker uses the Internet to obtain information on specific IP addresses. The kind of information gathered is operating system, services, and architecture of the target system. In Enumeration, the attacker gathers information on network user and group names, routing tables and simple network management protocol.

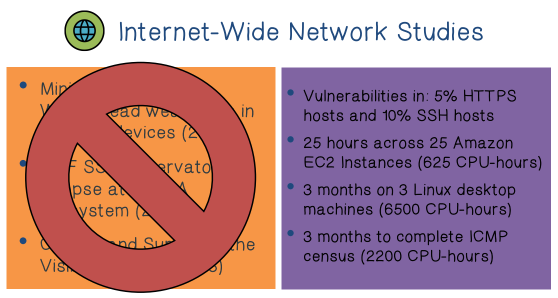


Internet Wide Security Scanning

- 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

There are a lot of benefits from Internet Wide Security Scanning. For example, we can discover new vulnerabilities, we can understand how a defensive mechanism has been adopted. On the other hand, Internet wide security scanning means that we have to probe the entire address space and with existing tools, this is a very difficult and slow task. Some of the security studies does require scanning of the entire address space. For example, if we can obtain the cryptographic keys in the entire address space, then we can understand the ecosystem of public key infrastructure and its vulnerabilities.

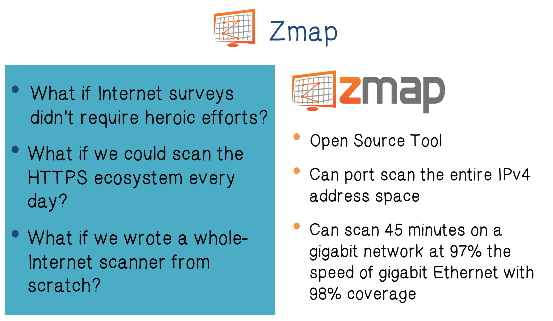


Internet-Wide Network Studies

- Mini… W… ead we… in … evices (2…
- … F SS… ervato… pse at… A … ystem (2…
- C… nd Su… he Visi… s)

- 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)

In fact, there have been quite a few influential Internet wide network studies. For example, in 2012, researchers at the University of Michigan released a paper describing the weak keys used in network devices. In particular, they found vulnerabilities in more than five percent of HTTPS hosts and 10 percent of SSH hosts, but the study required a lot of computing powers and time. Similarly, there was a study from the Electronic Frontier Foundation on public key certificates. Again, the study requires a lot of effort and a comprehensive scan of the whole Internet also requires months of efforts. All these projects require heroic effort just to perform initial data collection. As you can see, these scans require massive parallelization and typically they also require an extended period of time. Given the benefits of Internet-wide network security studies, we want to see a few more of them.



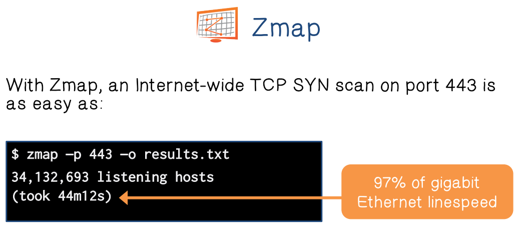
Zmap

- 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?

zmap

- 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

But in order to do that, Internet surveys should not require heroic efforts. For example, it would be nice if we can scan the whole HTTPS ecosystem every day. Given that existing tools are so slow, researchers at the University of Michigan developed ZMap, a new Internet scanner, from scratch. ZMap is a fully functional open-source network scanner that can scan a single port on the IPv4 address space. On a gigabit network, it can cover 98 percent of the IPv4 address space from a single machine in under 45 minutes.

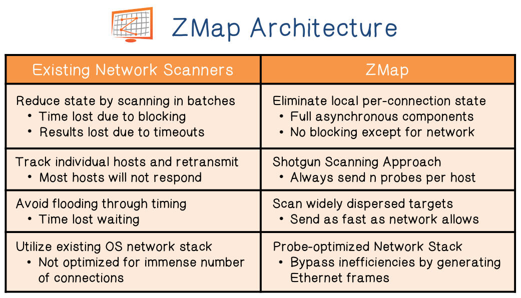


Zmap

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)
```

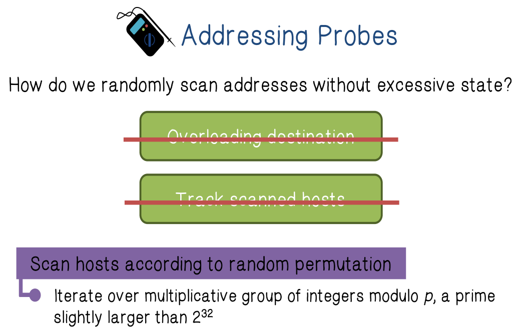
With ZMap, performing a scan on the Internet no longer requires hundreds of hours of coding or months to execute. Now, performing an Internet scan is a single command that could be finished within an hour.



ZMap Architecture

Existing Network Scanners	ZMap
Reduce state by scanning in batches • Time lost due to blocking • Results lost due to timeouts	Eliminate local per-connection state • Full asynchronous components • No blocking except for network
Track individual hosts and retransmit • Most hosts will not respond	Shotgun Scanning Approach • Always send n probes per host
Avoid flooding through timing • Time lost waiting	Scan widely dispersed targets • Send as fast as network allows
Utilize existing OS network stack • Not optimized for immense number of connections	Probe-optimized Network Stack • Bypass inefficiencies by generating Ethernet frames

While there are many excellent multi-purpose network scanners, these scanners were never intended or optimized for scanning the entire Internet. ZMap, on the other hand, was designed from the ground up for the specific purpose of completely scanning the whole Internet. Let us compare existing network scanners with ZMap. Existing network scanners need quite an immense amount of resources to keep track of state of the entire Internet. ZMap, on the other hand, eliminate local, per-connection state. Existing scanners also spend a lot of resources keeping track of which host is being scanned and which host have responded. On the other hand, ZMap uses a shotgun scanning approach and keeps minimum state information. Previous scanners have been attempting to be more polite by slowing down the scans. ZMap, on the other hand, distributes the scan across the Internet. This allows ZMap to scan at more aggressive rate without impacting the destination networks. Finally, ZMap also bypasses the inefficient network stack in the operating system and generate outgoing packets directly.



Addressing Probes

How do we randomly scan addresses without excessive state?

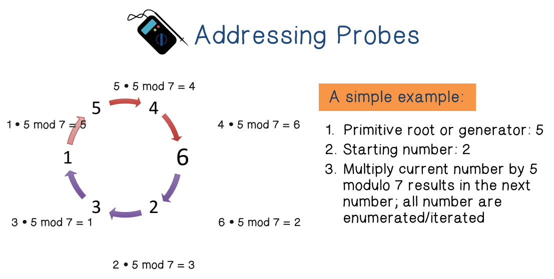
~~Overloading destination~~

~~Track scanned hosts~~

Scan hosts according to random permutation

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

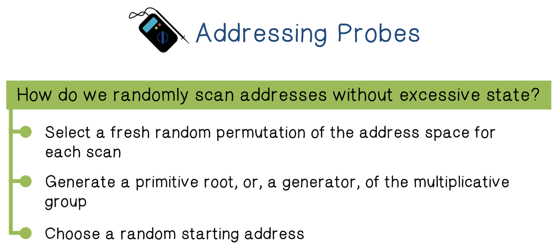
If we simply poll network addresses in numerical order, we will risk overloading the destination networks. However, we also do not want to have to track what host we have scanned or need to scan. In order to scan according to a random permutation, we select addresses according to a cyclic group. Specifically, we iterate over a multiplicative group of integers modulo a prime number slightly larger than 2^{32}.



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

Here is an example of how we randomly iterate through all the numbers in a group. The generator is 5, and we start with number 2. The process is that we multiply the current number by 5 mod 7, and the result is the next number, and all numbers will be enumerated. For example, we start with 2. 2 multiply by 5 mod 7 is 3. So, we move to 3. 3 times 5 mod 7 is 1, and so on. As you can see, we can iterate through all the numbers in a group.

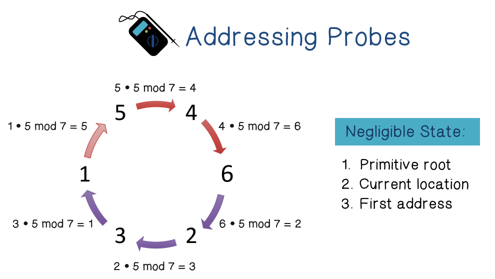


Addressing Probes

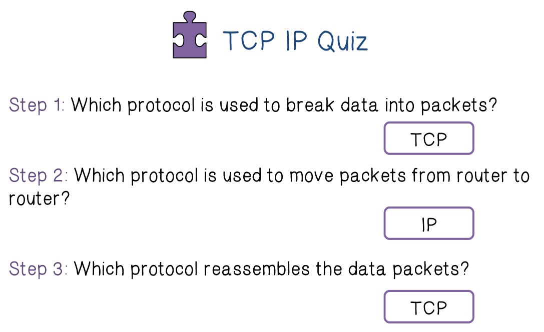
How do we randomly scan addresses without excessive state?

- 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

Following that simple example, here is how we can iterate through the addresses in address space. Each scan is essentially a random permutation of the address space. Again, we treat the address space as a multiplicative group. So, we first decide on the generator and then we choose a random starting address.



With this approach, we can guarantee that all addresses in the address space can be iterated. More importantly, there is very little state information we need to keep track of. Essentially, for each scan, we need to keep track of the primitive root or the generator and the current address because when the first address is visited again, we know that we have iterated all the addresses in the address space. As you can see, this is very little information to keep track of for each scan.



TCP IP Quiz

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

TCP

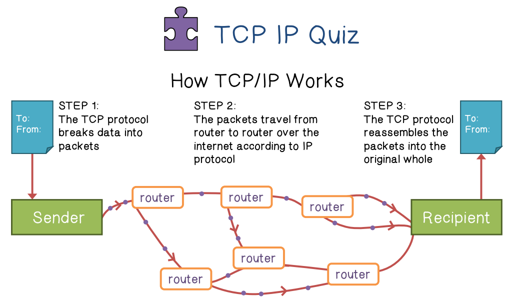
Step 2: Which protocol is used to move packets from router to router?

IP

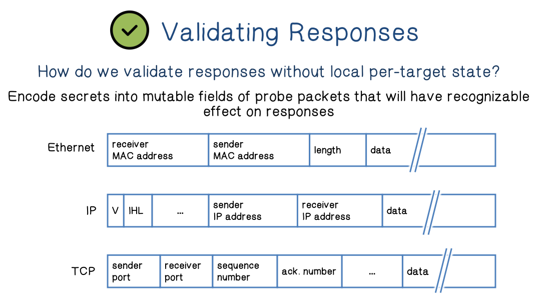
Step 3: Which protocol reassembles the data packets?

TCP

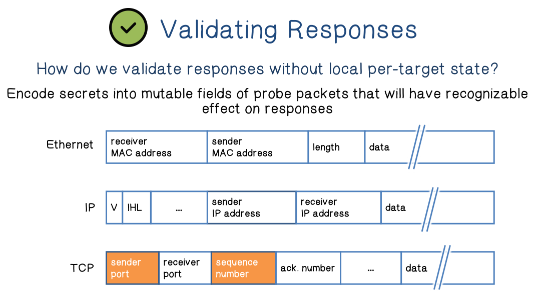
Now, let us do a quiz. Which protocol is used to break data into packets? It is TCP. Which protocol is used to move packets from router to router? It is IP, the Internet Protocol. Which protocol reassembles the data packets? Again, it is TCP.


TCP IP Quiz
How TCP/IP Works
To:
From:
STEP 1:
The TCP protocol breaks data into packets
STEP 2:
The packets travel from router to router over the internet according to IP protocol
STEP 3:
The TCP protocol reassembles the packets into the original whole
To:
From:
Sender
router
router
router
router
router
router
Recipient

And here is an illustration of the concepts in the quiz. On the sender's side, the TCP protocol breaks data into packets. When a packet comes out of the sender, the routers are responsible for sending a packet to the recipient. At the recipient the TCP protocol again resembles the packets back into the original data.



We have discussed how we send probe packets without storing what hosts we have scanned or need to scan, but how do we validate the responses? The idea here is that we can encode secrets into the fields of the probe packets. This is similar to the SYN cookies. Here are the formats of the Ethernet, IP and TCP packets, and some of the fields here can be used to encode the secrets.

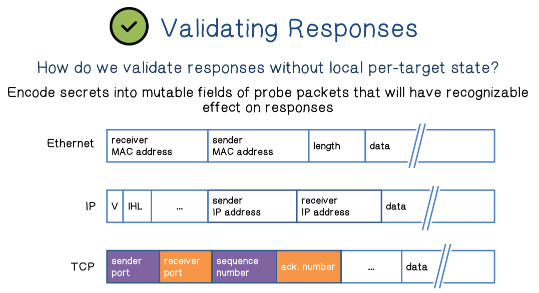


Validating Responses

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

In order to validate the packets that we receive, we encode a scan invocation and host-specific secret into mutable fields that will have a recognizable effect on the probe responses. Specifically, for each host to be scanned, ZMap computes a hash of the destination address keyed by a scan-specific secret. This MAC value, or Message Authentication Code, is then spread across any appropriate and available fields. In the case of using a TCP SYN packet for scanning, we encode the secret into the source port and sequence number. We know that the destination host will have to include the source address and port and acknowledge the sequence number to send a response.

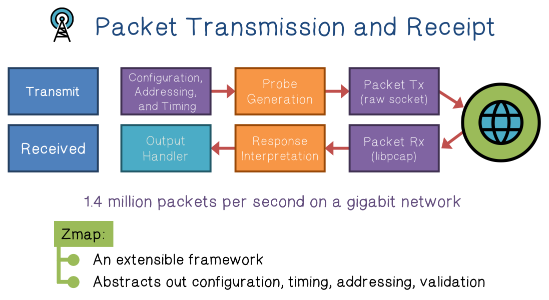


Validating Responses

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

For example, a sequence number will be encoded into the acknowledgement number field. More precisely, in the response packet the receiver port was the sender port in the scanning packet and the acknowledgement number is the sequence number of the scanning packet.

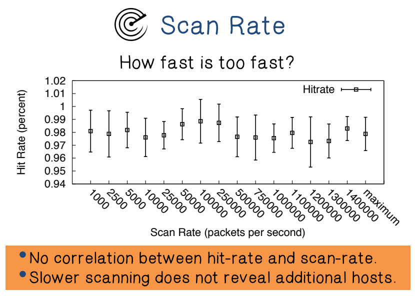


Packet Transmission and Receipt

1.4 million packets per second on a gigabit network

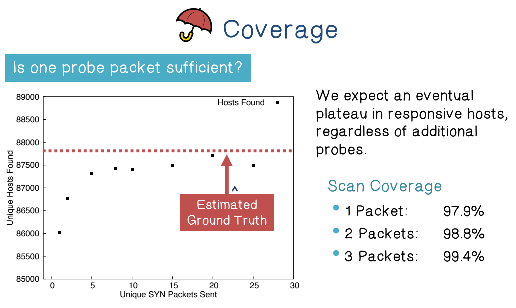
Zmap:
- An extensible framework
- Abstracts out configuration, timing, addressing, validation

Here are the high-level workflows of ZMap. To send scanning packets, ZMap will be configured, for example, with a random permutation of the address space. And then the probing packets will be sent to these addresses according to the random permutation. Since the majority of fields in a probe packet never change, ZMap performs all network operations at the Ethernet layer using a raw socket in order to cache values. This eliminates time lost to kernel operations such as route lookups. This allows ZMap to send probing packets at a very high speed and validate the responses. For example, checking the MAC value and the probing results can then be analyzed. This configuration allows ZMap to send probes at approximately 1.4 million packets per second on a gigabit network. ZMap is an extensible framework to allow any type of single packet scan such as a TCP SYN scan, ICMP echo requests or application specific UDP scan. The ZMap framework abstracts out details such as configuration, timing, addressing and validation.



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

The first major question is whether ZMap or our network can handle scanning at gigabit speed. In order to answer this question researchers at the University of Michigan performed 10 trials scanning one percent of the IPv4 address space at different rates ranging from 1,000 packets per second to the maximum 1.4 million packets per second. This maximum is determined by the NIC configuration. NIC stands for network interface card. As we can see there is no correlation between a scanning rate and the number of hosts that respond. One way to interpret this result is that even slower scanning rates does not produce additional response.

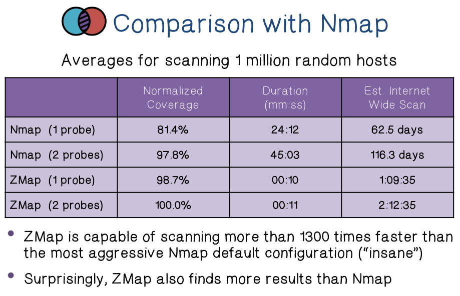


We expect an eventual plateau in responsive hosts, regardless of additional probes.

Scan Coverage

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

Another interesting question is, “Is one probe packet sufficient or is sending multiple packets beneficial?” This is difficult to answer directly because there is no ground truth for the number of live hosts on the Internet. In order to estimate ZMap coverage, multiples packets were sent to one percent samples of the IPv4 address space. If you look at the number of responses, we clearly see a plateau after a one-SYN packet scan. In fact, we should expect to see an eventual plateau in the number of responsive hosts regardless of additional SYN packets sent. Analyzing these results, we can estimate that with a single packet the coverage is about 97.9 percent and with three packets the coverage is 99.4 percent.



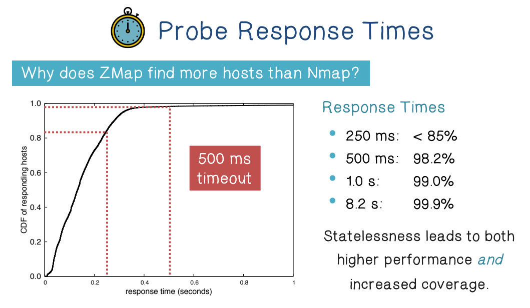
Comparison with Nmap

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

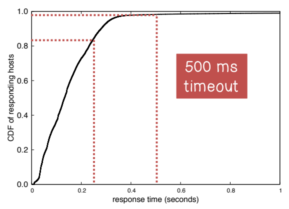
- ZMap is capable of scanning more than 1300 times faster than the most aggressive Nmap default configuration (“insane”)
- Surprisingly, ZMap also finds more results than Nmap

Many previous projects have used the popular Nmap network scanner to perform scans. Therefore, we should compare ZMap with Nmap. The researchers performed several experiments focusing on the total time spent for scanned and scan coverage. In these experiments, one million addresses were scanned. In these experiments, Nmap used the most aggressive scan template called insane and with a minimal scan rate of 10,000 packets per second. As we can see from these results, ZMap is capable of scanning the IPv4 address space more than 1,300 times faster than Nmap. ZMap also has higher coverage than Nmap even when Nmap sends multiple probes. To be fair, Nmap is an excellent multi-purpose network scanner and it is optimized for completely different use cases than ZMap.



Probe Response Times

Why does ZMap find more hosts than Nmap?

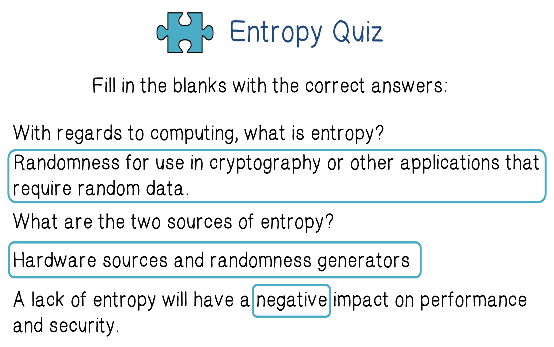


Response Times

- 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.

So why does ZMap have higher coverage than Nmap? We analyzed the response time of the scan responses. ZMap does not timeout hosts, but Nmap does. In fact, if Nmap sends one packet, it times out after 250 milliseconds. If it uses two packets, it times out after 500 milliseconds. As you can see, some responses arrive after Nmap has timed out. Ultimately, since ZMap uses stateless scanning and does not keep state, it has both increased 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.

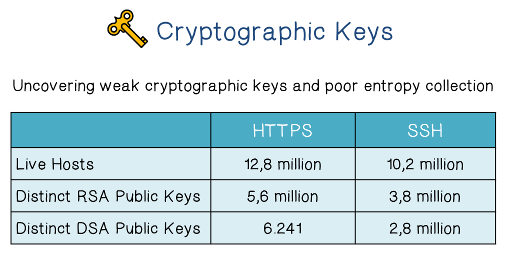
What are the two sources of entropy?

Hardware sources and randomness generators

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

Now let us do a quiz on entropy. Fill in the blanks with the correct answers.

With regards to computing. What is entropy? Entropy is randomness for use in cryptography or other applications that require random data. What are the two sources of entropy? Hardware sources or randomness generators. A lack of entropy will have a negative impact on performance and security.

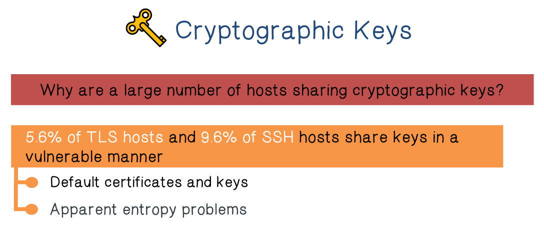


Cryptographic Keys

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

Now, let's discuss a few interesting Internet wide security studies using ZMap. In one study, the researchers scanned the HTTPS servers and the SSH servers and collected their public keys. As you can see, a large number of machines share their public keys. Although there are many legitimate reasons why machines would share their keys, we need to understand whether the keys happen to be the same without the intention of sharing because that is bad for security.



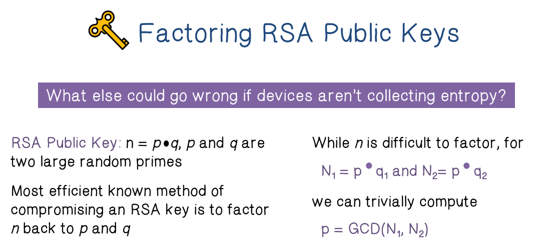
Cryptographic Keys

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

In particular, researchers find that some of these machines share their key in a vulnerable manner. For example, they simply use the default keys without creating their own keys and there is an apparent entropy problem that caused them to have the same keys.



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

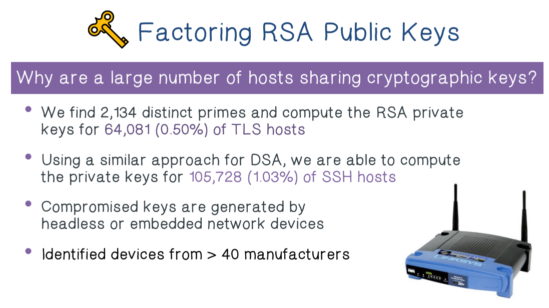
While n is difficult to factor, for

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

we can trivially compute

$p = GCD(N_1, N_2)$

Let us look at the effect of not having sufficient entropy. As a quick review, in RSA, the public key n is the product of p and q, where p and q are large random prime numbers. The security of RSA is based on the fact that factoring n back to p and q is very, very inefficient. That is, it is very hard for the attacker. On the other hand, if two private keys N_1 and N_2 share the same prime number, say p, then it is trivial to compute p because p is the greatest common denominator (GCD) of N_1 and N_2. The reason that two different public keys may have the same prime number can be that there is not sufficient entropy and therefore two machines may happen to generate the same large prime number p. There is also a very efficient algorithm due to D. J. Bernstein to compute the GCD of every pair of N_1 and N_2.

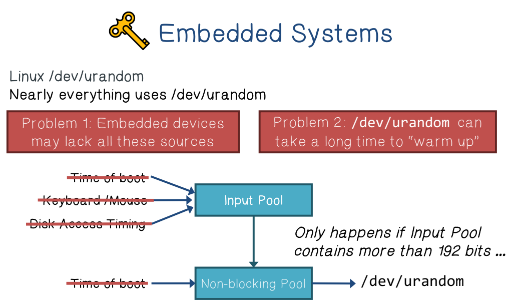


Factoring RSA Public Keys

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

With this efficient approach, the researchers were able to crack many keys on TLS and SSH machines. The majority of these machines are network devices.



Embedded Systems

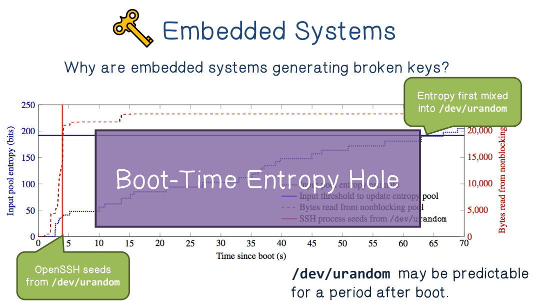
Linux /dev/urandom

Nearly everything uses /dev/urandom

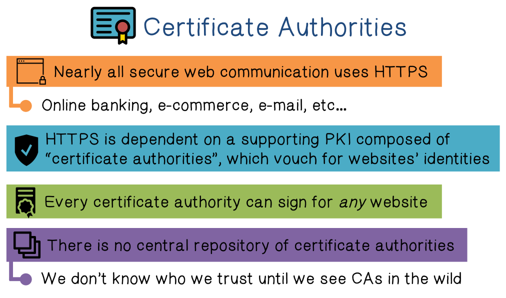
Problem 1: Embedded devices may lack all these sources

Problem 2: `/dev/urandom` can take a long time to "warm up"

Now, let us look at why these embedded systems or network devices generate broken keys. These embedded systems all run Linux and they all use urandom to generate cryptographic keys. Linux maintains several entropy pools. As entropy is gathered, it is stored in the input pool and is eventually mixed into the non-blocking pool which feeds urandom. However, many of these sources are not present on an embedded system. For example, the system may not have keyboard and many of them do not have spindle-based disks. In fact, most of these embedded systems do not even support real-time clocks. Therefore, with all of these random sources removed, we end up with a deterministic source of randomness. Furthermore, entropy is a mix from the input pool to the non-blocking pool until there are more than 192 bits of entropy that have been collected. Therefore, even if a small amount of entropy has been collected on a device, none of it is available until the pool reaches this threshold.



This graph shows the build-up of entropy on the first boot of a typical Ubuntu desktop server. No randomness is added from the input pool to urandom until 192 bits of entropy is available. From the graph, we can tell that this did not happen until fairly late in the boot process. This is the so-called Boot-Time Entropy Hole problem, which means that urandom may be predictable for a period after boot. Unfortunately, cryptographic keys may be based on this predictable urandom.



Certificate Authorities

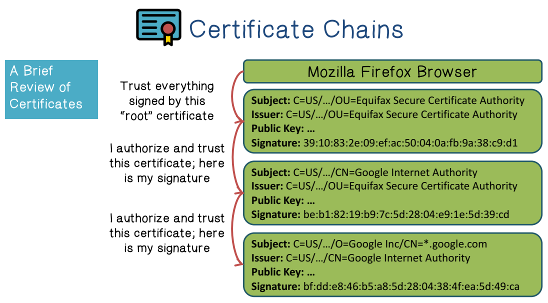
Nearly all secure web communication uses HTTPS
- Online banking, e-commerce, e-mail, etc…

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

Another interesting study is on the certificates issued by certificate authorities. This is an important topic because HTTPS underlines all secure Web communications and HTTPS is depending on the security of certificate authorities. The problem is that there are many certificate authorities and every one of them can sign for any website. In fact, we do not even know all the certificate authorities, until we see them.



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

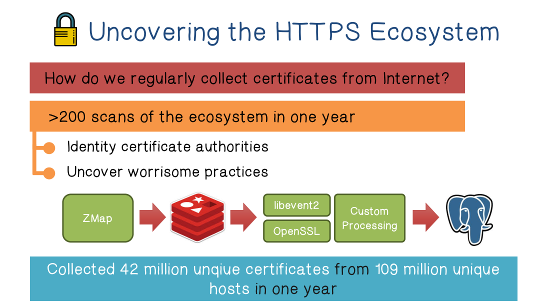
Mozilla Firefox Browser

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

Let us have a quick review of how certificates are used in Web browsing. Browsers such as Firefox decide whom they trust. These certificates are stored as part of the browser or the operating system. Browsers typically support a couple hundred root certificates, such as shown in this figure. These root certificate authorities then agree to sign other certificates. For example, Equifax is a root certificate authority and agrees to sign Google with authority. And this can go on involving multiple intermediates and we end up having a certificate chain. That is, if you look at Google.com, it is signed by a chain of certificates and a top certificate is just a self-signed certificate. This entire chain is presented by Web browser to the client and the client can verify the signatures by traversing the certificate chain up to the browser trusted certificate, which is self-signed.



Uncovering the HTTPS Ecosystem

How do we regularly collect certificates from Internet?

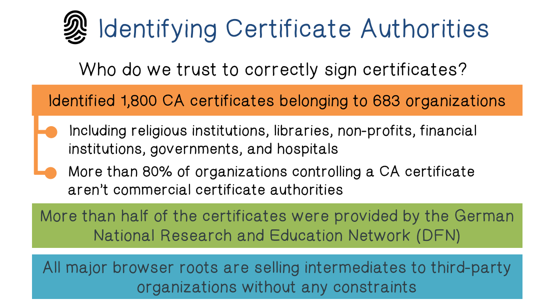
>200 scans of the ecosystem in one year

- Identity certificate authorities
- Uncover worrisome practices

ZMap
libevent2
OpenSSL
Custom Processing

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

The researchers at the University of Michigan performed regular scans to track the use of certificates. They observed 3,700 browser trusted certificates in one year. They also discovered two cases of misused certificates. In one case, a signing certificate was accidentally issued to a Turkish transit provider. In another case, 1,300 certificates were issued by the Korean government.



Identifying Certificate Authorities

Who do we trust to correctly sign certificates?

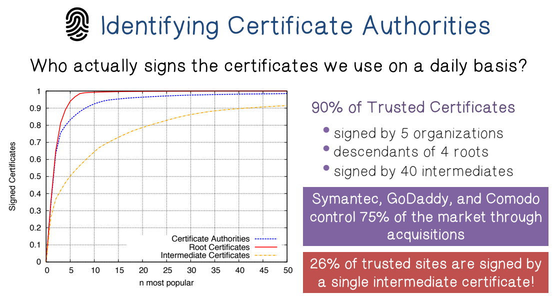
Identified 1,800 CA certificates belonging to 683 organizations

- 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

The researchers found 1,800 signed certificates belonging to 683 organizations and these include various kinds of institutions. The most worrisome finding is that all major browser roots are selling intermediates to third parties without any constraints.



Identifying Certificate Authorities

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

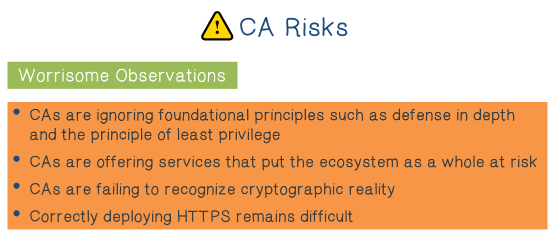
90% of Trusted Certificates

- 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!

By analyzing the certificate chains, we see that 90 percent of the certificates are signed by five organizations. They are sent from four roots and signed by 40 intermediates and there are only a few big players. Another interesting fact is that 26 percent of the trusted sites are signed by only 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

At a high level, there are several worrisome observations. First, the certificate authorities are ignoring defense in depth and least privilege. They are offering services that put the whole ecosystem at risk and they are using some weak keys and deploying HTTPS remains difficult. Let us go on with these observations next.

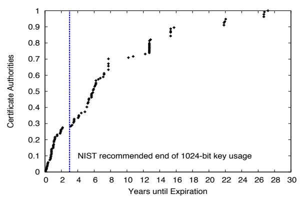
CA Risks

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

For defense in depth, there are several technical practices already in place for limiting the scope of a signing certificate including setting name or path length constrains and distributing leave certificates among a large number of intermediate certificates. There are clear cases for using these restrictions but the vast majority of the time, the CAs do not utilize these options. As another example, local domain names are not fully qualified and intended resource is ambiguous and there is no identifiable owner. As such, these local domain names frequently appear on more than one certificate. In one example, there are about 1,218 browser trusted certificates for the domain mail owned by organizations ranging from the U.S. Department of Defense to small companies.

CA Risks

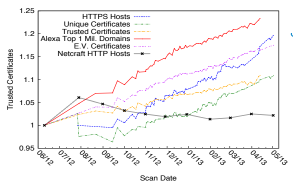


Cryptographic Reality

- 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!

In terms of the keys used, the researchers also found that 90 percent of the certificates use a 2048-bit or 4096-bit RSA keys. But 50 percent of the certificates were rooted in the 1024 bit key. More importantly, more than 70 percent of these would expire after 2016 and many still signed using MD5.

Growth in HTTPS Adoption



June 2012–May 2013

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

The researchers also found that within a year, the number of HTTPS servers only increased by 10% and there was a 23% increase in the number of Alexa top 1 million sites using the HTTPS. And there was 11% increase in the raw number of browser trusted certificates. Here is an example of how we use Internet wide scan to check the adoption of a technology.

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!

The researchers at the University of Michigan had released ZMap as an open source project. You can download ZMap and use it for your own Internet wide security study. Of course, please remember to be a good citizen on the Internet.

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

There is also a repository of data from previous scans.

ZMap: Fast Internet-wide Scanning and Its Security Applications

Zmap is a modular single packet network scanner designed for interne-wide network surveys. On a computer with a gigabit connection, Zmap can scan the entire public IPv4 address space in under 45 minutes. Although scanning is usually associated with botnets and worms, it can also be used as a valuable methodology in security research. It can be used to reveal all kinds of vulnerabilities, monitor deployments of mitigations and throw light on previously opaque distributed ecosystems.

The paper starts by giving an introduction to Zmap and all of its advantages, it compares Zmap to Nmap, which is a widely used network mapping tool. Zmap performs much better than Nmap for the following reasons (page 1 and page 2):

1. Optimized probing

2. No per-connection states

3. No retransmission

Section 2 describes the working of Zmap. It starts by explaining how the scan addresses are decided using randomized permutations of the address space and how it excludes certain address spaces. The packet transmission and receipt are designed in such a way that it maximizes the number of probes it can send based on the source's CPU and NIC. The probe modules are responsible for filling in the body of the probe and validate the incoming packets.

This section is followed by a set of experiments performed on Zmap and their results. It compares if the scan rate of Zmap has any effects on the hit rate and checks if coverage is an issue at such speeds (Section 3.1 and 3.2). Then there is an extensive comparison of Zmap with Nmap and other previous methods that have been used for Internet scanning. The comparisons include time taken, the coverage and timeouts (Section 3.4 and 3.5).

The next section explores the research possibilities with the ability to scan the entire IPv4 address space. This can be used for increased visibility into distributed systems, tracking how successful the adoption of new protocols is, enumerating vulnerable hosts and discovering unadvertised services that were previously available only with explicit knowledge of host name or address (Sections 4.1 to 4.5). This is followed by suggestions for other researchers who are performing fast Internet wide scans, these are guidelines that ensure that these tools are not used with the wrong intentions.