CSE227 – Graduate Computer Security

TLS II

UC San Diego

Housekeeping

General course things to know

- Midpoint check-in document is due **5/8 at 11:59pm PT**
  - Introduction (frame the problem)
  - Related work section (should include ~5 – 10 relevant papers)
  - Research plan, current status, what's left to do
- Midpoint check-in meetings will happen week 7, so be prepared for that
- Resources should get finalized this week!
  - Most things have been ordered; provisioned, etc., some things remain!
- Arshia OH moving to Zoom —> link on website

Zoom Rules

- Cameras on when answering questions (and always if you can!)
- Stay muted when not speaking
- “Raise hand” to ask questions
    - But feel free to jump in and answer if you know the answer (we’ll see how this goes…)
- Unmute + answer cold calls :)
- I won’t be looking at chat, so don’t use it

Today's lecture

Learning Objectives

- Discuss the Ps and Qs paper, and why the paper is so significant
- Talk about TLS certificates, how we get them, and how it's decided
- Talk about Certificate Misissuance Paper

Where we left off…

Last time!

- We talked about the RSA construction
  - What are p and q, and how are they used in RSA?
  - What is the public modulus N?
  - Why is RSA hard to break?

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

A few words on this paper…

- This is a UCSD paper!
  - Nadia Heninger is one of the lead authors, done when she was a postdoc here at UCSD
  - The other lead was *my* postdoc advisor, done when he was a first-year graduate student (much like you!)
- This paper won best paper at USENIX Security 2012
- This paper won the USENIX Security Test-of-Time award in 2022
- Needless to say… it's a very important computer security paper. Why?

A few more words on this paper…

- IMO, one of the greatest paper titles of all time
- What does “minding your Ps and Qs” mean?

A few more words on this paper…

- IMO, one of the greatest paper titles of all time
- What does “minding your Ps and Qs” mean?
- So what does “mining your Ps and Qs” mean?
  - Factoring weak RSA public key information!

An RSA “vulnerability”

- This paper exploits a “vulnerability” in the RSA cryptosystem’s underlying assumptions. What assumption does it violate?

An RSA “vulnerability”

- This paper exploits a “vulnerability” in the RSA cryptosystem’s underlying assumptions. What assumption does it violate?
  - That public modulus N will **never share any factors with any other N.**

An RSA “vulnerability”

- This paper exploits a “vulnerability” in the RSA cryptosystem’s underlying assumptions. What assumption does it violate?
  - That public modulus N will **never share any factors with any other N.**
- Why is sharing factors so bad?

An RSA “vulnerability”

- This paper exploits a “vulnerability” in the RSA cryptosystem’s underlying assumptions. What assumption does it violate?
  - That public modulus N will **never share any factors with any other N.**
- Why is sharing factors so bad?
  - GCDs are efficiently computable (see Euclid’s algorithm)
  - If N1 and N2 share a factor *p*, the GCD between N1 and N2 would be *p*, and the key can be trivially broken

Who cares if you break someone’s private key?

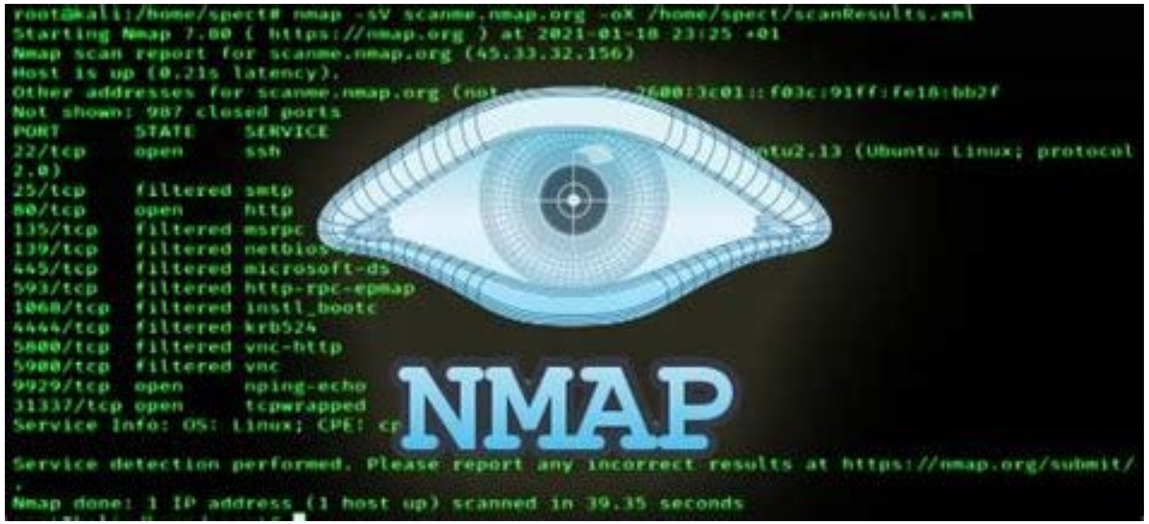
- What scenarios do the authors describe?

Who cares if you break someone's private key?

- What scenarios do the authors describe?
  - Passive attacker: If key exchange is RSA, then a passive attacker can decrypt **entire encrypted session** after the fact
  - Active attacker: If key exchange is Diffie-Hellman, passive adversary won't work, but active attacker (e.g., MiTM) can modify / decrypt traffic

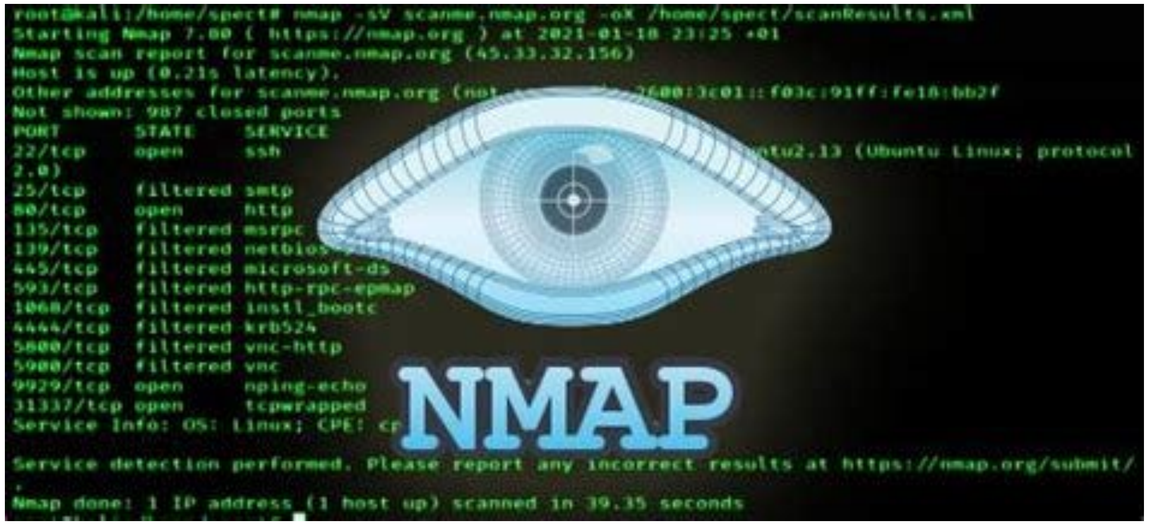
How do we find these vulnerable keys?

- What is network scanning?



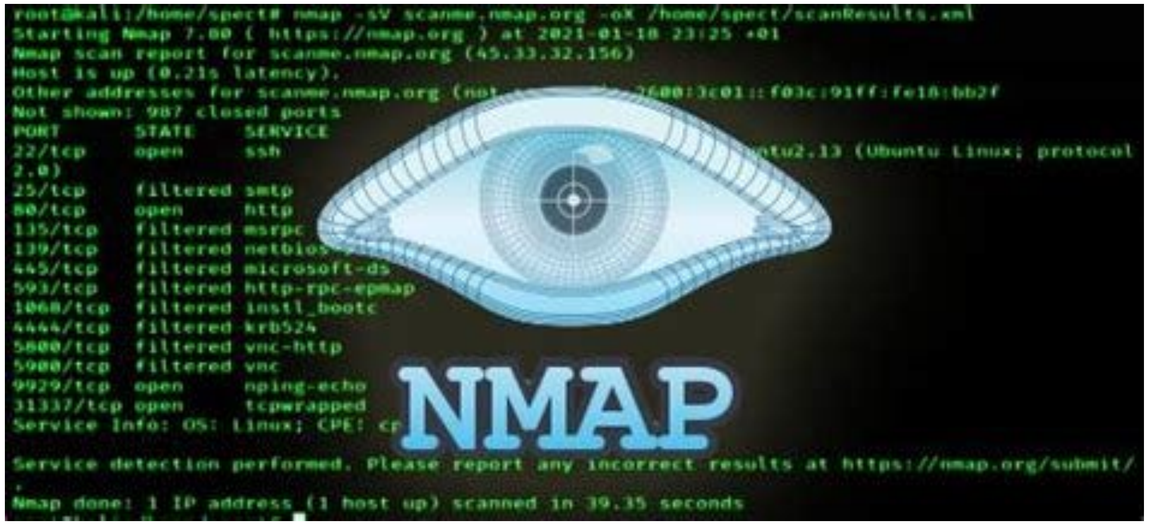
How do we find these vulnerable keys?

- What is network scanning?
- How does network scanning work to identify TCP hosts?



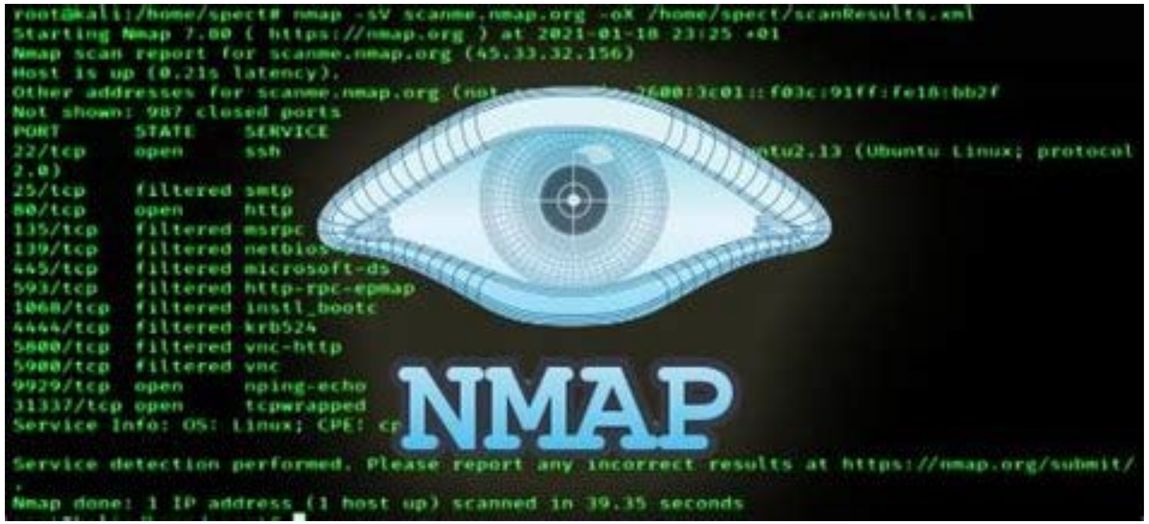
How do we find these vulnerable keys?

- What is network scanning?
- How does network scanning work to identify TCP hosts?
- What was the search space the authors searched in this paper?



How do we find these vulnerable keys?

- What is network scanning?
- How does network scanning work to identify TCP hosts?
- What was the search space the authors searched in this paper?
  - How long did it take the authors to scan the IPv4 Internet?

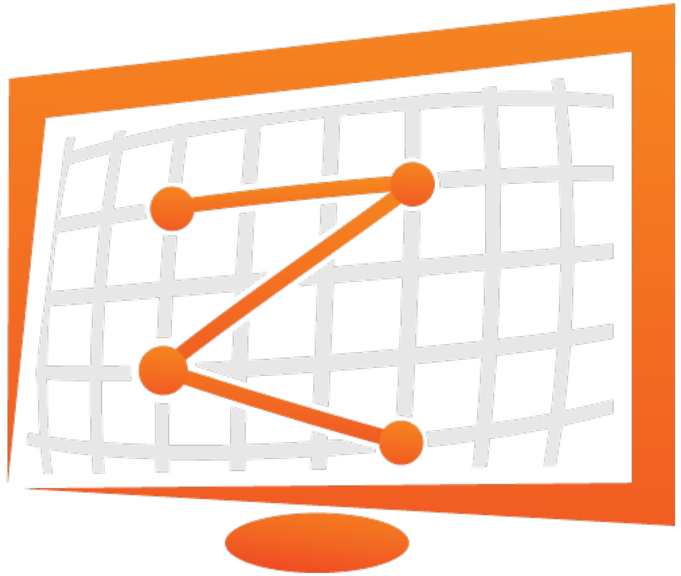


Scan Results

	SSL Observatory (12/2010)	Our TLS scan (10/2011)	Our SSH scans (2-4/2012)
Hosts with open port 443 or 22	≈16,200,000	28,923,800	23,237,081
Completed protocol handshakes	7,704,837	12,828,613	10,216,363
Distinct RSA public keys	3,933,366	5,656,519	3,821,639
Distinct DSA public keys	1,906	6,241	2,789,662
Distinct TLS certificates	4,021,766	5,847,957	—
Trusted by major browsers	1,455,391	1,956,267	—

Table 1: **Internet-wide scan results** — We exhaustively scanned the public IPv4 address space for TLS and SSH servers listening on ports 443 and 22, respectively. Our results constitute the largest such network survey reported to date. For comparison, we also show statistics for the EFF SSL Observatory's most recent public dataset [18].

Side note…



censys

Efficient Factorization

- Paper uses a combination of product trees and remainder trees to efficiently compute GCDs for their moduli
  - Computed GCDs for 11.1M moduli in about 5h, this would be much faster today

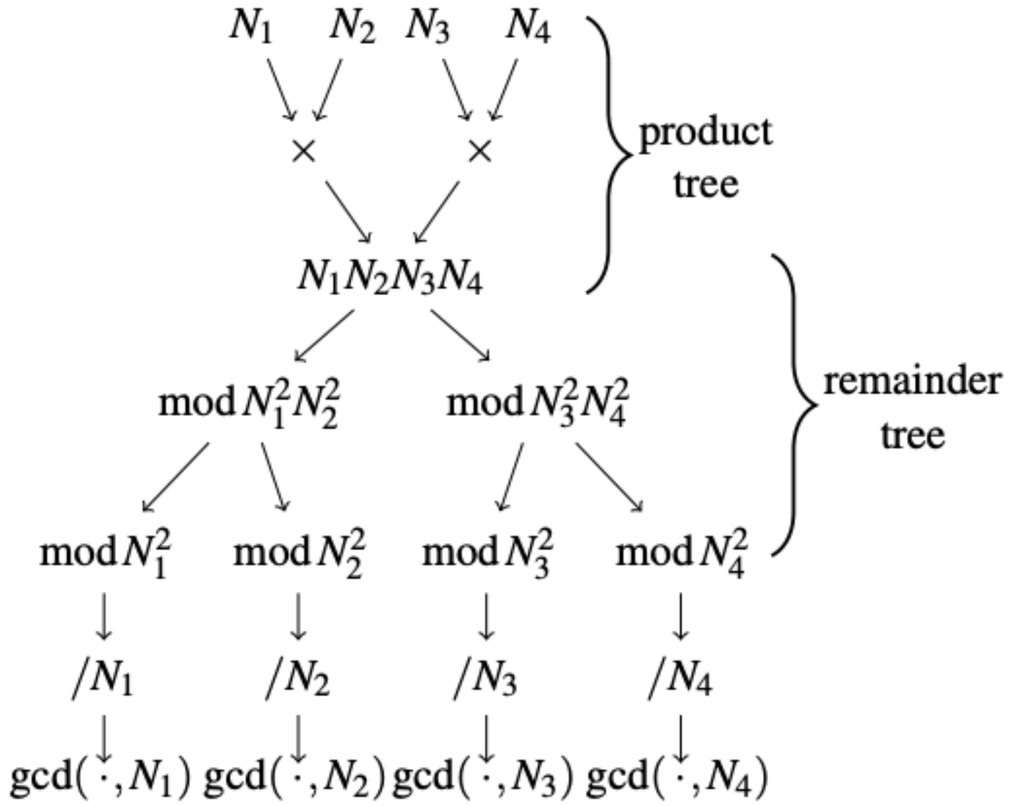


Figure 1: **Computing all-pairs GCDs efficiently** — We computed the GCD of every pair of RSA moduli in our dataset using an algorithm due to Bernstein [6].

Repeated Keys are Common

- Authors found 61% of TLS hosts **served the same key**
- Why does this happen in practice?

Repeated Keys are Common

- Authors found 61% of TLS hosts **served the same key**
- Why does this happen in practice?
  - Hosting providers might share keys for ease of deployment
  - Some keying material is embedded in firmware as *default* – **5.23%** of hosts were manufacturer defaults

Who is putting bad keys into the world?

- The authors identified many vulnerable device vendors and models. How?

Who is putting bad keys into the world?

- The authors identified many vulnerable device vendors and models. How?
  - There's **a lot** of information volunteered in TLS certificates…

Who is putting bad keys into the world?

- The authors identified many vulnerable device vendors and models. How?
  - There's **a lot** of information volunteered in TLS certificates…
- Authors identified vulnerable devices from 27 manufacturers. What were these manufacturers creating?

Who is putting bad keys into the world?

- The authors identified many vulnerable device vendors and models. How?
  - There's **a lot** of information volunteered in TLS certificates…
- Authors identified vulnerable devices from 27 manufacturers. What were these manufacturers creating?
  - Enterprise-grade routers, server management, VPN devices, security systems, consumer routers… things you *totally* want having bad crypto

What did the authors find?

	Our TLS Scan		Our SSH Scans	
Number of live hosts	12,828,613	(100.00%)	10,216,363	(100.00%)
. . . using repeated keys	7,770,232	(60.50%)	6,642,222	(65.00%)
. . . using vulnerable repeated keys	714,243	(5.57%)	981,166	(9.60%)
. . . using default certificates or default keys	670,391	(5.23%)		
. . . using low-entropy repeated keys	43,852	(0.34%)		
. . . using RSA keys we could factor	64,081	(0.50%)	2,459	(0.03%)
. . . using DSA keys we could compromise			105,728	(1.03%)
. . . using Debian weak keys	4,147	(0.03%)	53,141	(0.52%)
. . . using 512-bit RSA keys	123,038	(0.96%)	8,459	(0.08%)
. . . identified as a vulnerable device model	985,031	(7.68%)	1,070,522	(10.48%)
. . . model using low-entropy repeated keys	314,640	(2.45%)		

Table 2: **Summary of vulnerabilities** — We analyzed our TLS and SSH scan results to measure the population of hosts exhibiting several entropy-related vulnerabilities. These include use of repeated keys, use of RSA keys that were factorable due to repeated primes, and use of DSA keys that were compromised by repeated signature randomness. Under the theory that vulnerable repeated keys were generated by embedded or headless devices with defective designs, we also report the number of hosts that we identified as these device models. Many of these hosts may be at risk even though we did not specifically observe repeats of their keys.

What did the authors find?

	Our TLS Scan		Our SSH Scans	
Number of live hosts	12,828,613	(100.00%)	10,216,363	(100.00%)
. . . using repeated keys	7,770,232	(60.50%)	6,642,222	(65.00%)
. . . using vulnerable repeated keys	714,243	(5.57%)	981,166	(9.60%)
. . . using default certificates or default keys	670,391	(5.23%)		
. . . using low-entropy repeated keys	43,852	(0.34%)		
. . . using RSA keys we could factor	64,081	(0.50%)	2,459	(0.03%)
. . . using DSA keys we could compromise			105,728	(1.03%)
. . . using Debian weak keys	4,147	(0.03%)	53,141	(0.52%)
. . . using 512-bit RSA keys	123,038	(0.96%)	8,459	(0.08%)
. . . identified as a vulnerable device model	985,031	(7.68%)	1,070,522	(10.48%)
. . . model using low-entropy repeated keys	314,640	(2.45%)		

Table 2: **Summary of vulnerabilities** — We analyzed our TLS and SSH scan results to measure the population of hosts exhibiting several entropy-related vulnerabilities. These include use of repeated keys, use of RSA keys that were factorable due to repeated primes, and use of DSA keys that were compromised by repeated signature randomness. Under the theory that vulnerable repeated keys were generated by embedded or headless devices with defective designs, we also report the number of hosts that we identified as these device models. Many of these hosts may be at risk even though we did not specifically observe repeats of their keys.

Why is this happening?

- What is entropy?

Why is this happening?

- What is entropy?
  - “The amount of unpredictable randomness” in a physical system

Why is this happening?

- What is entropy?
  - “The amount of unpredictable randomness” in a physical system
- Where does entropy come from?

Why is this happening?

- What is entropy?
  - “The amount of unpredictable randomness” in a physical system
- Where does entropy come from?
  - Uninitialized contents of memory when the kernel starts, startup clock time, disk access timings, “old” entropy

Why is this happening?

- What is entropy?
  - “The amount of unpredictable randomness” in a physical system
- Where does entropy come from?
  - Uninitialized contents of memory when the kernel starts, startup clock time, disk access timings, “old” entropy
- What did the authors discover about headless / embedded devices?

Implementations are tricky

- Linux provides two sources of randomness: /dev/random and /dev/urandom. What's the difference between the two?

Implementations are tricky

- Linux provides two sources of randomness: /dev/random and /dev/urandom. What's the difference between the two?
  - In 2012, /dev/random was **blocking**, /dev/urandom was **non-blocking,** now they're mostly the same after initialization

Implementations are tricky

- Linux provides two sources of randomness: /dev/random and /dev/urandom. What's the difference between the two?
  - In 2012, /dev/random was **blocking**, /dev/urandom was **non-blocking,** now they're mostly the same after initialization
- People preferred the **non-blocking** interface for randomness (even when the randomness was predictable). Why?

Meta-thoughts on the paper

- What do we think about this paper? Did we enjoy it, why or why not?
- Why do we think this paper won so many awards when the results only impacted such a small % of hosts?
- What were some limitations of the study you can think of?

Break Time + Attendance

https://tinyurl.com/cse227-attend

Codeword:
MindYourBusiness

Tracking Certificate Misissuance in the Wild

TLS Fundamentals

TLS provides three fundamental security properties:

TLS Fundamentals

TLS provides three fundamental security properties:

- Confidentiality. What is confidentiality, and how does TLS provide it?

TLS Fundamentals

TLS provides three fundamental security properties:

- Confidentiality. What is confidentiality, and how does TLS provide it?
- Integrity. What is integrity, and how does TLS provide it?

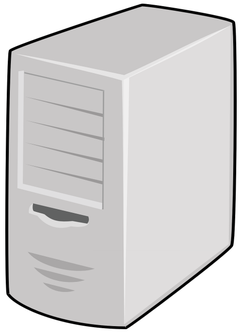
TLS Fundamentals

TLS provides three fundamental security properties:

- Confidentiality. What is confidentiality, and how does TLS provide it?
- Integrity. What is integrity, and how does TLS provide it?
- **Authenticity**
  - Validating the source or origin of data: i.e., *knowing who you are talking to*

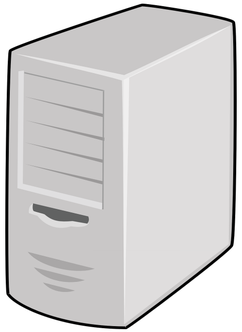
How do you get HTTPS on your website?

kumarde.com



How do you get HTTPS on your website?

kumarde.com



SECTIGO®

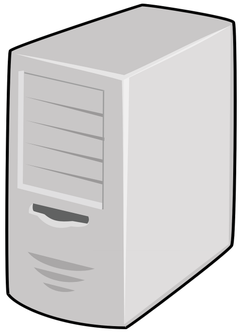
digicert®

Let's Encrypt

What is a Certificate Authority (CA)?

How do you get HTTPS on your website?

kumarde.com



SECTIGO®

digicert®

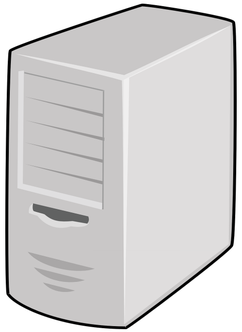
Let's Encrypt

What is a Certificate Authority (CA)?

"A trusted entity that issues digital certificates to verify the identity of individuals, companies, email addresses, and websites."

How do you get HTTPS on your website?

kumarde.com



certificate for kumarde.com pls

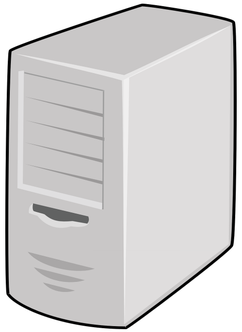
SECTIGO®

digicert®

Let's Encrypt

How do you get HTTPS on your website?

kumarde.com



certificate for kumarde.com pls

verify who you are!

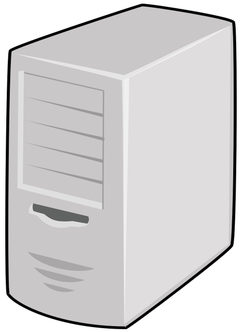
SECTIGO®

digicert®

Let's Encrypt

How do you get HTTPS on your website?

kumarde.com



certificate for kumarde.com pls

verify who you are!

SECTIGO®

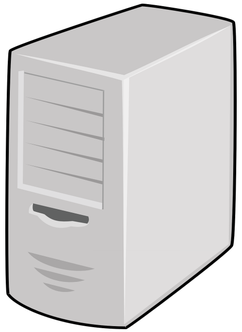
digicert®

Let's Encrypt

What are some ways that CAs verify your identity?

How do you get HTTPS on your website?

kumarde.com



certificate for kumarde.com pls →

verify who you are! ←

here is a cert *signed by us* ←

SECTIGO®

digicert®

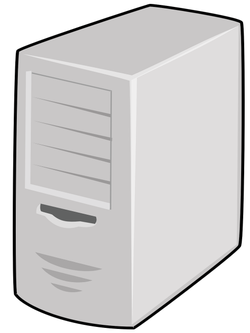
Let's Encrypt

Certs have lots of details, but most importantly they have your **public key,** thereby linking your verified identity to your cryptographic identity

How do you get HTTPS on your website?

Client

kumarde.com

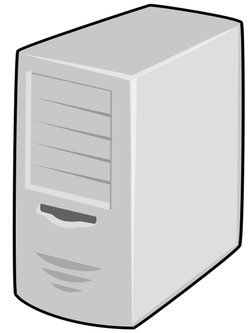


How do you get HTTPS on your website?

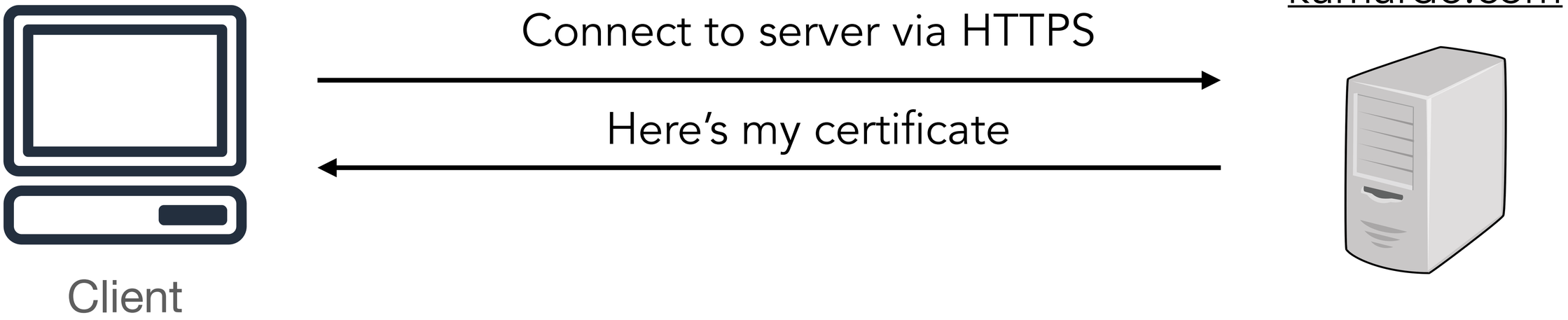
Client

Connect to server via HTTPS

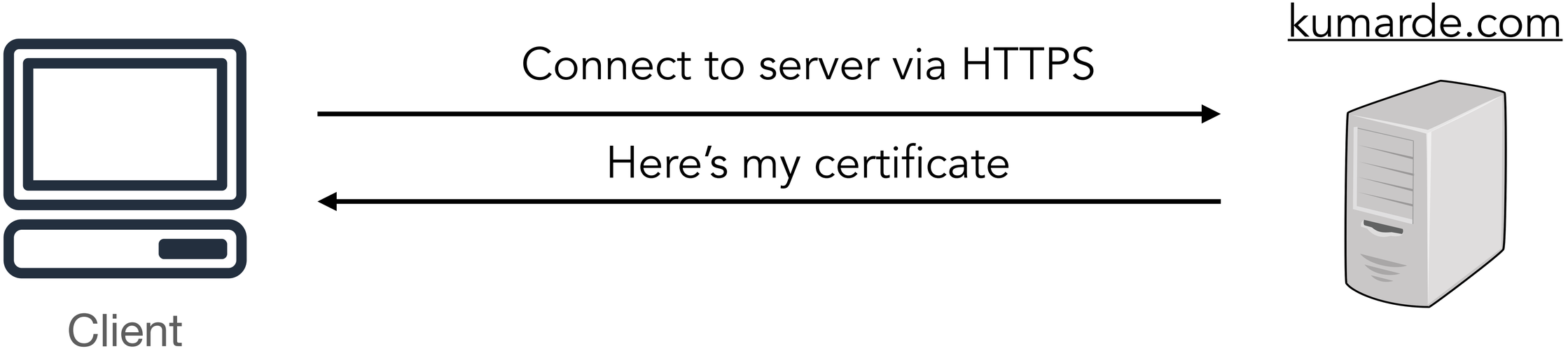
kumarde.com



How do you get HTTPS on your website?



How do you get HTTPS on your website?



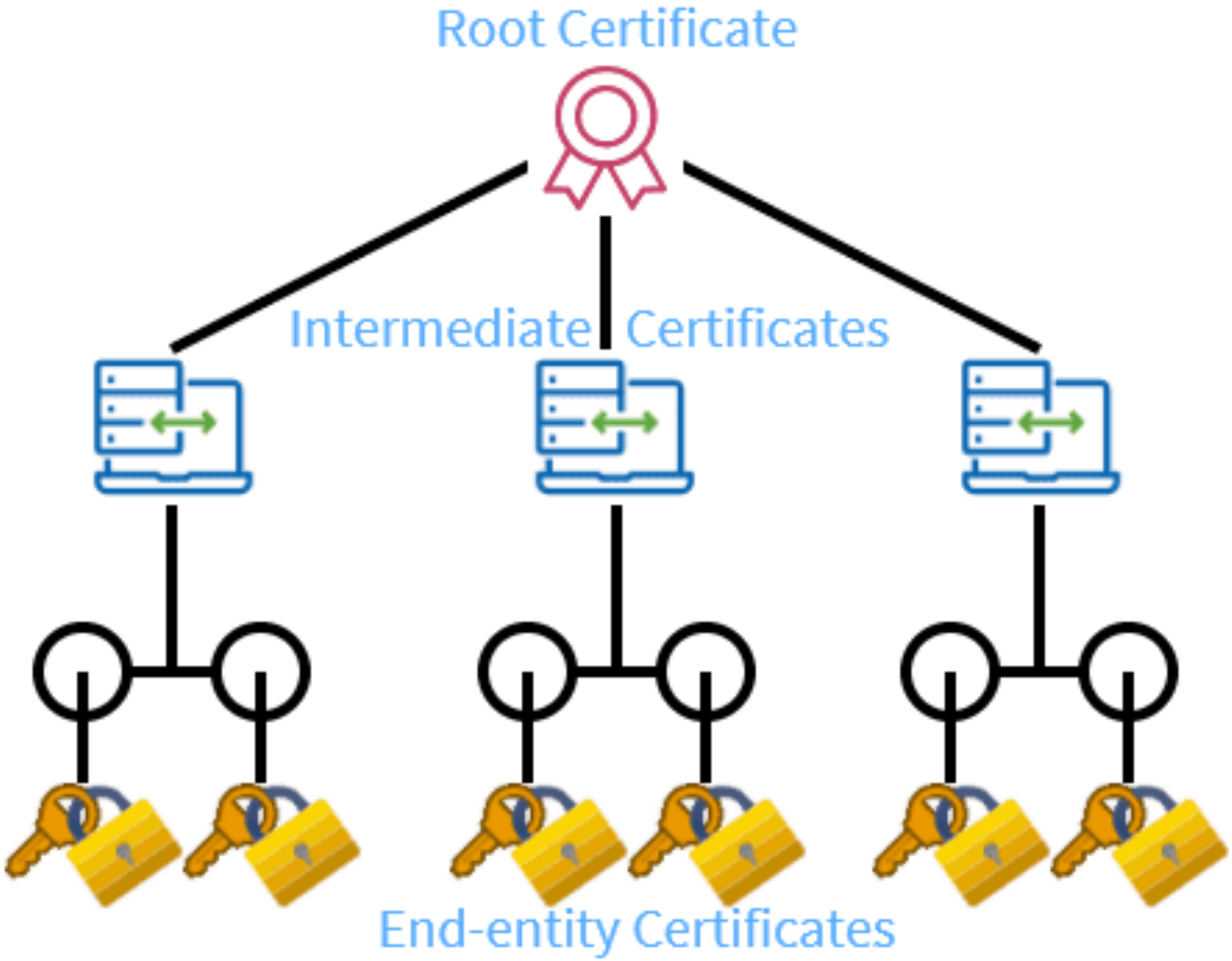
What does the client need to do with the certificate in order to trust the server on the other end?

Certificate validation

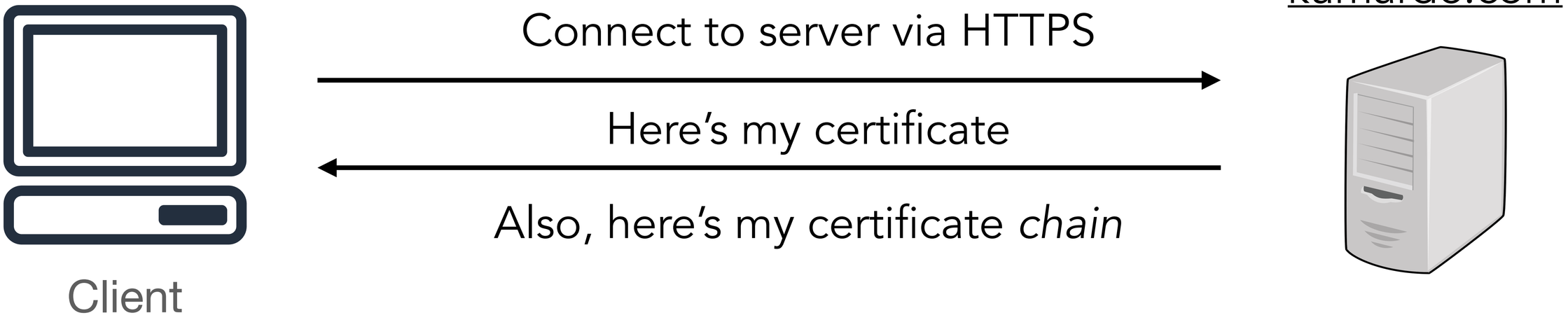
- Chain-of-trust verification
- Hostname verification
- Revocation checks

What is a chain of trust?

What is a chain of trust?



How do you get HTTPS on your website?



Who decides who the roots of trust are?

Who decides who the roots of trust are?

CA/BROWSER FORUM

Public-Key Infrastructure (HTTPS)

Otherwise known as the Web PKI

- The web PKI is the hardware, software, policies, processes, and procedures that exist to manage, distribute, and revoke web certificates and public-keys
- Every machine or software that communicates with HTTPS **comes with roots installed** – these are explicitly trusted by browsers, OSes, etc.
- Chock-full of arbitrary and complicated rules that have evolved over time due to “problems”
  - E.g., Roots can’t sign leaf certificates, roots need to have intermediates that can sign leaf certificates

Certificate Authorities, Redux

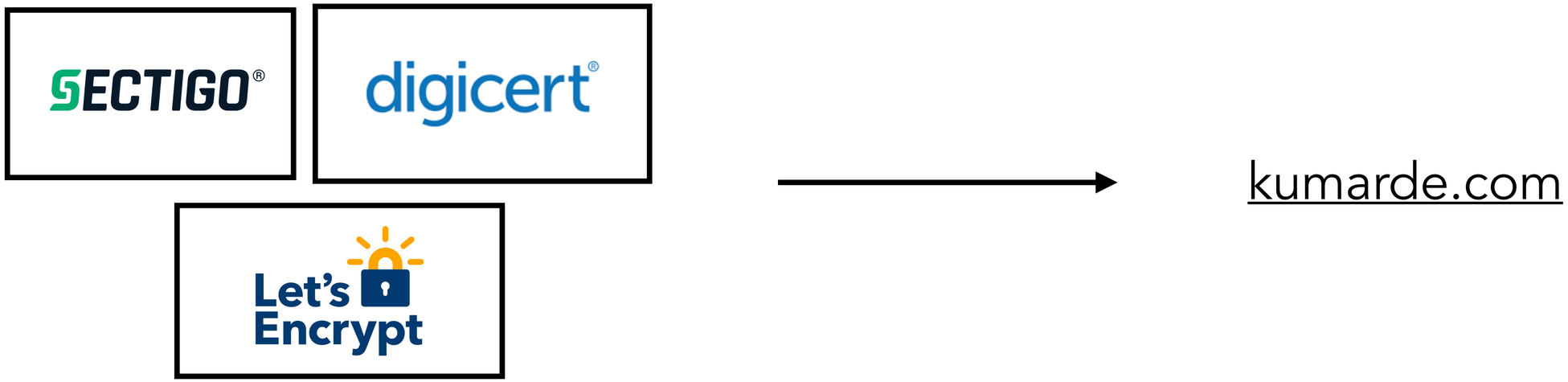
- There is a fundamental “vulnerability” with the way the certificate authority ecosystem is set up. What is it?

Certificate Authorities, Redux

- There is a fundamental “vulnerability” with the way the certificate authority ecosystem is set up. What is it?
  - ***Anyone* trusted can sign for *anyone* on the Internet!**

Certificate Authorities, Redux

- There is a fundamental “vulnerability” with the way the certificate authority ecosystem is set up. What is it?
  - ***Anyone* trusted can sign for *anyone* on the Internet!**



Iranian Man-in-the-Middle Attack Against Google Demonstrates Dangerous Weakness of Certificate Authorities

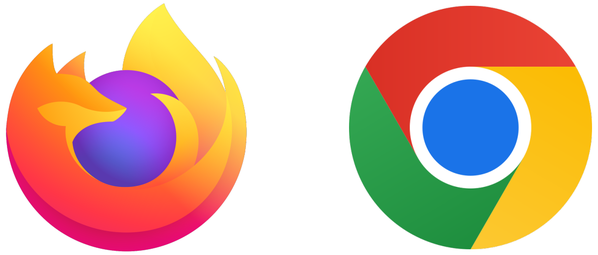
The TURKTRUST SSL certificate fiasco – what really happened, and what happens next?

Google Blocks Fraudulent Certificates Used by French Government

Revoking Trust in one CNNIC Intermediate Certificate

The rules that govern the PKI

- Two major sets of policies that CAs must follow:
  - CA/B Baseline Requirements
  - RFC 5280 – X509 Certificate Standard



ZLint: An X.509 Certificate Linter

- This paper a certificate **linter**, called ZLint, that measures if a certificate is *misissued*
  - How do the authors identify if a certificate is *misissued*?

ZLint: An X.509 Certificate Linter

- This paper a certificate **linter**, called ZLint, that measures if a certificate is *misissued*
  - How do the authors identify if a certificate is *misissued*?

“It's 2017 - it's both time to stop making excuses and time to recognize that the ability of CAs to adhere to the rules is core to their trustworthiness. Technical rules are but a proxy for procedure rules.” - Ryan Sleevi

How it really happened...

I lost my mind the summer of 2017

Network Working Group
Request for Comments: 5280
Obsoletes: 3280, 4325, 4630
Category: Standards Track

D. Cooper
NIST
S. Santesson
Microsoft
S. Farrell
Trinity College Dublin
S. Boeyen
Entrust
R. Housley
Vigil Security
W. Polk
NIST
May 2008

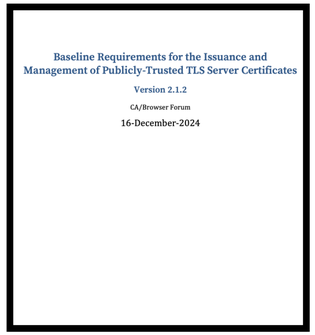
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This memo profiles the X.509 v3 certificate and X.509 v2 certificate revocation list (CRL) for use in the Internet. An overview of this approach and model is provided as an introduction. The X.509 v3 certificate format is described in detail, with additional information regarding the format and semantics of Internet name forms. Standard certificate extensions are described and two Internet-specific extensions are defined. A set of required certificate extensions is specified. The X.509 v2 CRL format is described in detail along with standard and Internet-specific extensions. An algorithm for X.509 certification path validation is described. An ASN.1 module and examples are provided in the appendices.



Baseline Requirements for the Issuance and Management of Publicly-Trusted TLS Server Certificates

Version 2.1.2

CA/Browser Forum

16-December-2024

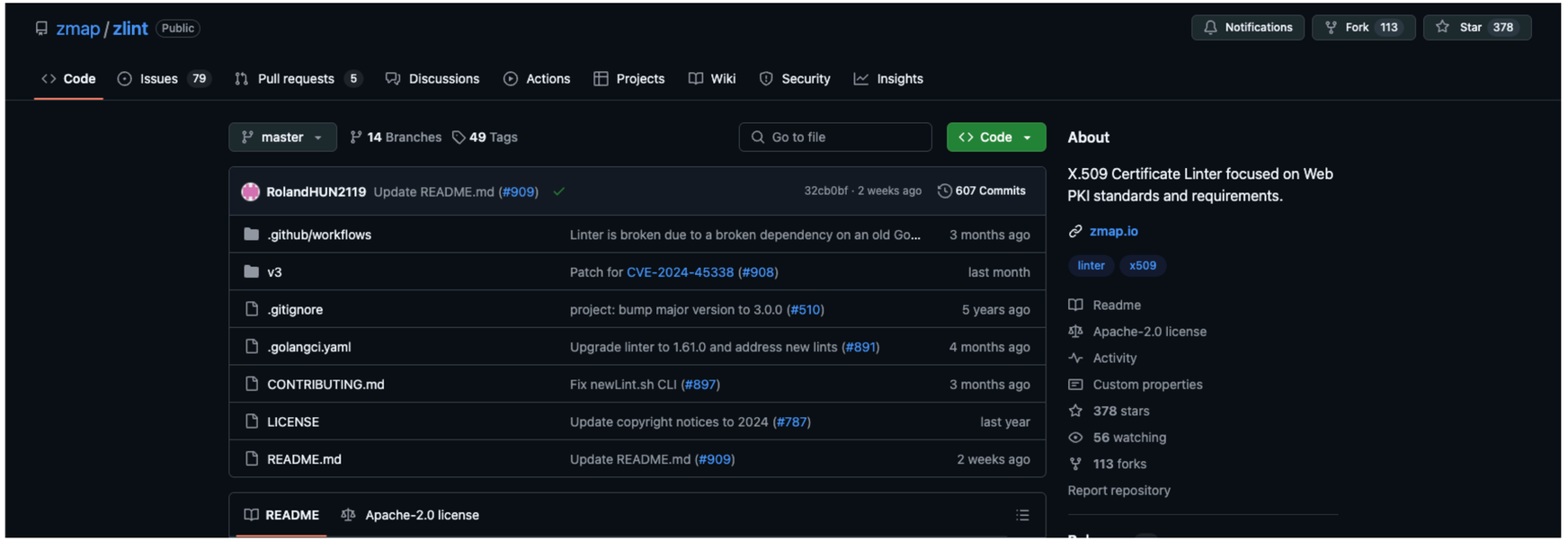
Rules

PhD Me

```go
import (
	"strings"

	"github.com/zmap/zcrypto/x509"
	"github.com/zmap/zlint/v3/lint"
	"github.com/zmap/zlint/v3/util"
)

type DNSNameUnderscoreInTRD struct{}

func init() {
	lint.RegisterCertificateLint(&lint.CertificateLint{
		LintMetadata: lint.LintMetadata{
			Name:        "w_rfc_dnsname_underscore_in_trd",
			Description:   "DNSName MUST NOT contain underscore characters",
			Citation:      "RFC5280: 4.1.2.6",
			Source:        lint.RFC5280,
			EffectiveDate: util.RFC5280Date,
		},
		Lint: NewDNSNameUnderscoreInTRD,
	})
}

func NewDNSNameUnderscoreInTRD() lint.LintInterface {
	return &DNSNameUnderscoreInTRD{}
}

func (l *DNSNameUnderscoreInTRD) CheckApplies(c *x509.Certificate) bool {
	return util.IsSubscriberCert(c) && util.DNSNamesExist(c)
}

func (l *DNSNameUnderscoreInTRD) Execute(c *x509.Certificate) *lint.LintResult {
	parsedSANDNSNames := c.GetParsedDNSNames(false)
	for i := range c.GetParsedDNSNames(false) {
		if parsedSANDNSNames[i].ParseError != nil {
			return &lint.LintResult{Status: lint.NA}
		}
		if strings.Contains(parsedSANDNSNames[i].ParsedDomain.TRD, "_") {
			return &lint.LintResult{Status: lint.Warn}
		}
	}

	return &lint.LintResult{Status: lint.Pass}
}
```

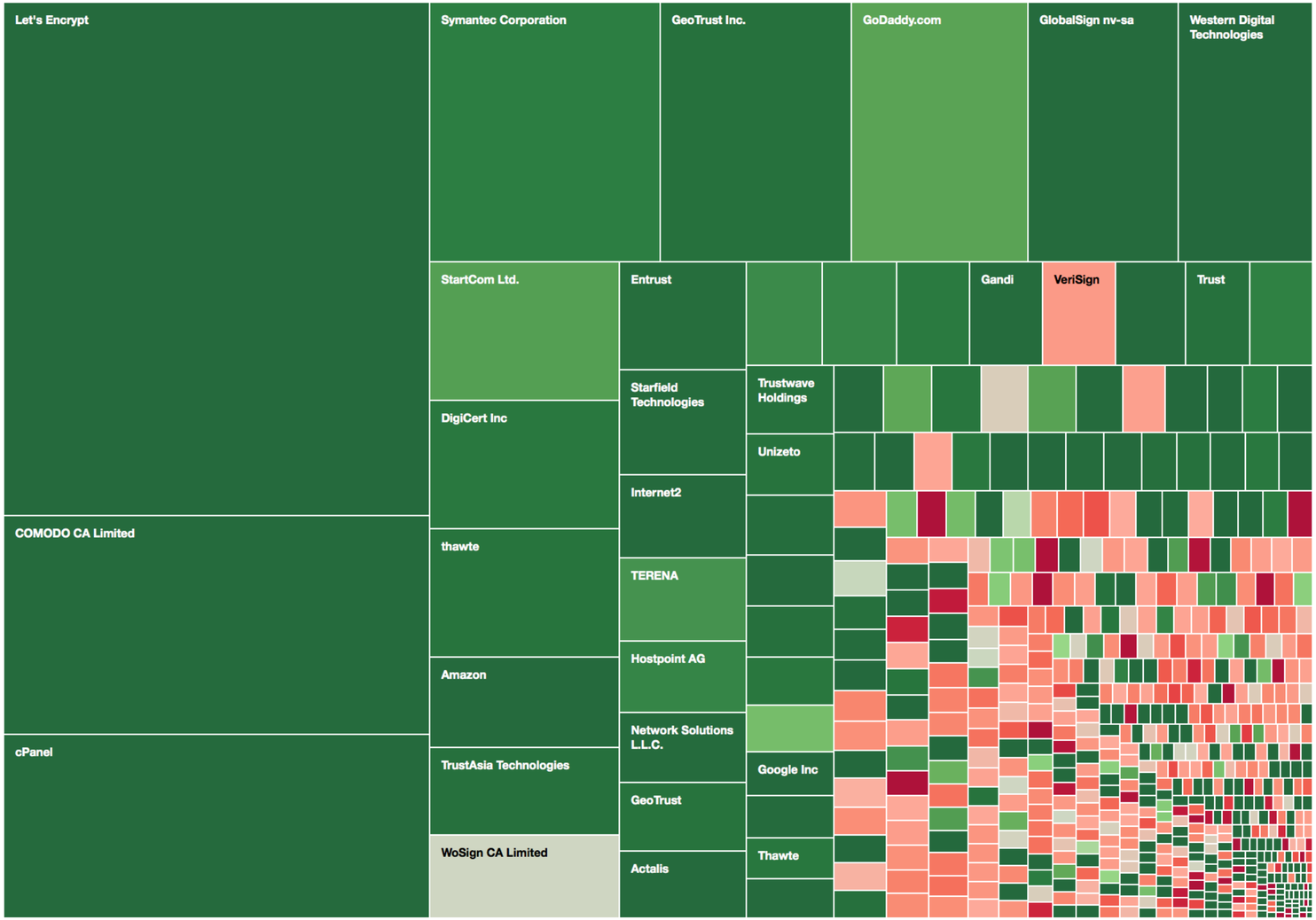
Code

Side note: ZLint remains an active OSS project!



Used by almost every major CA on the planet: Let’s Encrypt, Google, Digicert, etc.

TL;DR Big CAs are pretty good, small CAs are terrible



Some of my favorite takeaways: Nestle is a CA?

Organization	Misissued	
Nestle (1)	968	100%
PSCProcert (1)	39	100%
Giesecke and Devrient (1)	18	100%
Unizeto Sp. z o.o. (1)	18	100%
CertiPath LLC (1)	9	100%
Helsana Gruppe (1)	8	100%
Chunghwa Telecom Co. (1)	7	100%
TSCP Inc. (1)	5	100%
Dell Inc. (1)	4	100%
DigitPA (1)	2	100%

(a) Highest Misissuance Rate

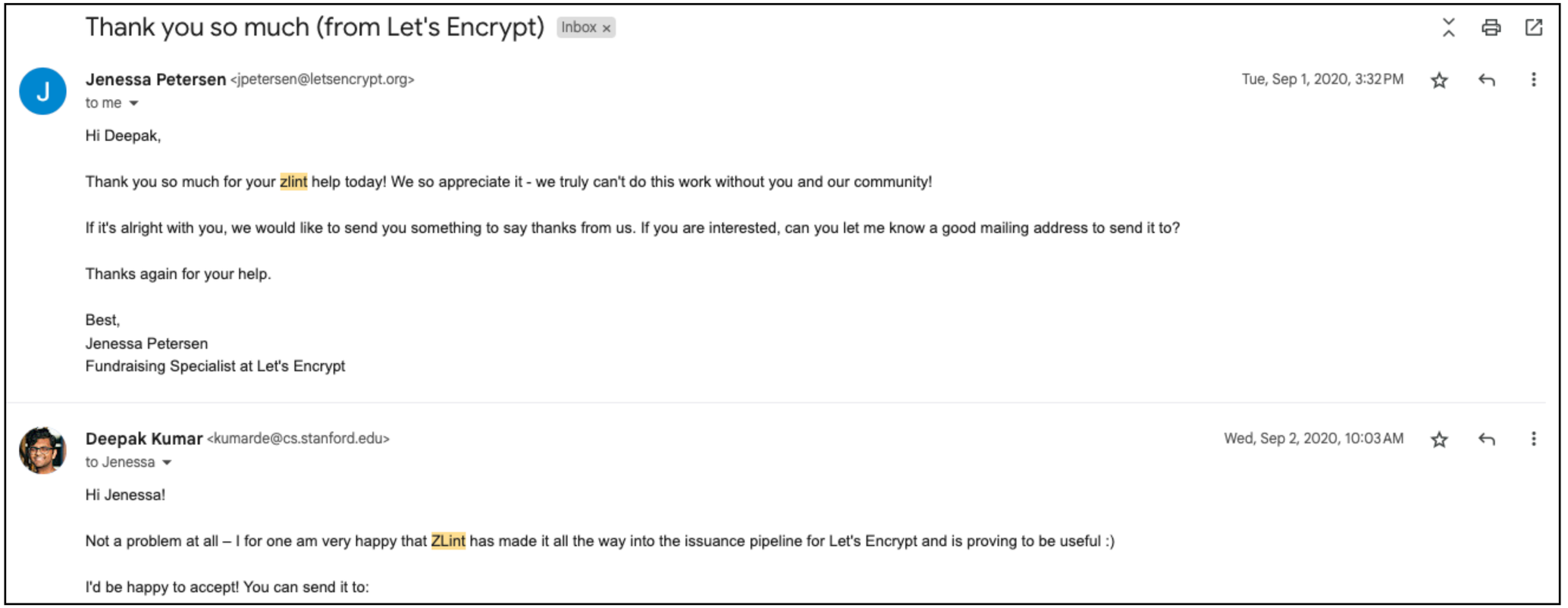
Organization	Misissued	
Consorci Catalunya (2)	1,117	58.8%
RHRK (2)	1,171	35.6%
KPN Corporate BV (2)	1,933	34.5%
DFN-Verein (5)	1,689	29.8%
Universitaet Stuttgart (1)	1,830	29.2%
AC Camerfirma S.A. (1)	2,725	25.9%
VeriSign (10)	42,622	23.1%
Trend Micro Inc (1)	6,374	19.8%
AlphaSSL (1)	3,848	17.2%
Uni Erlangen Nuernberg (1)	1,115	14.4%

(b) Highest Misissuance Rate (>1K certificates)

Organization	Misissued	
GoDaddy.com (3)	38,215	2.4%
Symantec Corp.† (22)	23,053	0.8%
StartCom Ltd.‡ (17)	11,617	2.1%
WoSign CA Lmtd.‡ (39)	9,849	5.0%
VeriSign† (10)	9,835	23.1%
GeoTrust Inc.† (22)	5,694	0.3%
Comodo Ltd. (30)	3,219	0.1%
DigiCert (43)	2,597	0.1%
Thawte† (4)	1,751	0.4%
TERENA (9)	1,405	1.7%

(c) Most Misissued Certificates

Sometimes, people are nice



Thank you so much (from Let's Encrypt) Inbox ×

Jenessa Petersen <jpetersen@letsencrypt.org> Tue, Sep 1, 2020, 3:32 PM

to me

Hi Deepak,

Thank you so much for your zlint help today! We so appreciate it - we truly can't do this work without you and our community!

If it's alright with you, we would like to send you something to say thanks from us. If you are interested, can you let me know a good mailing address to send it to?

Thanks again for your help.

Best,
Jenessa Petersen
Fundraising Specialist at Let's Encrypt

Deepak Kumar <kumarde@cs.stanford.edu> Wed, Sep 2, 2020, 10:03 AM

to Jenessa

Hi Jenessa!

Not a problem at all – I for one am very happy that ZLint has made it all the way into the issuance pipeline for Let's Encrypt and is proving to be useful :)

I'd be happy to accept! You can send it to:

Sometimes, people are mean

Meta-thoughts on the paper

- Certificate Misissuance is no longer really a problem…. so what did we learn from this paper?
- What about this paper surprised you? What didn’t surprise you?
- Why do we think there are so many small CAs? Is there anything to do about them? How do we make our system more resilient against these types of threats?

Next time…

- Moving back down the stack a little to focus on network attacks – DDoS and botnets!
- Midpoint check-in due soon… keep up the good work!