

EECS 388



Introduction to Computer Security

Lecture 10:

HTTPS Attacks and Defenses

September 26, 2024

Prof. Halderman



Last week:

- The Web Platform
- Web Attacks and Defenses

This week:

- HTTPS and the Web PKI
- **HTTPS Attacks and Defenses**

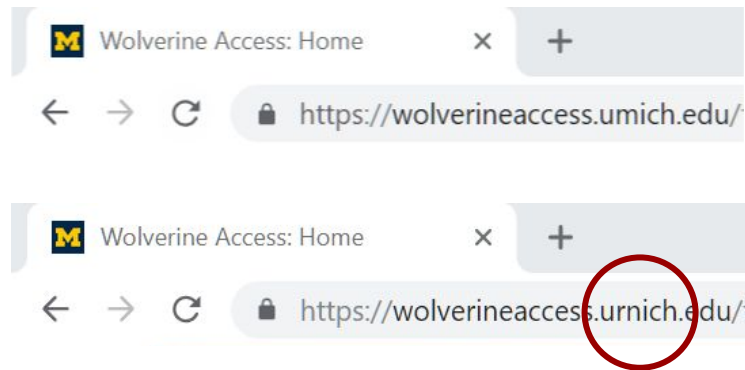
Next week:

- Networking 101
- Networking 102

Later:

- Network Defense
- User Authentication
- Online Privacy and Anonymity

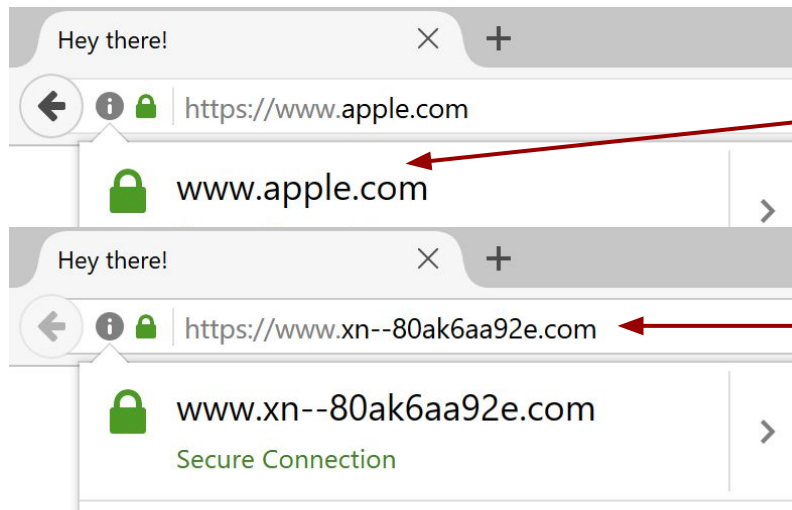
1. Fooling Users: Homographs



Homograph attacks use visually similar domain to trick users. It's simple but effective.

Attackers buy look-alike domain and acquire a legitimate cert, then set up a phishing site.

IDN homograph attacks use international characters for a pixel-perfect match.



Cyrillic letters that look just like "apple"

Punycode (ASCII) encoding of same name

Mitigation: Browsers use punycode to block many IDN homographs, but users beware!

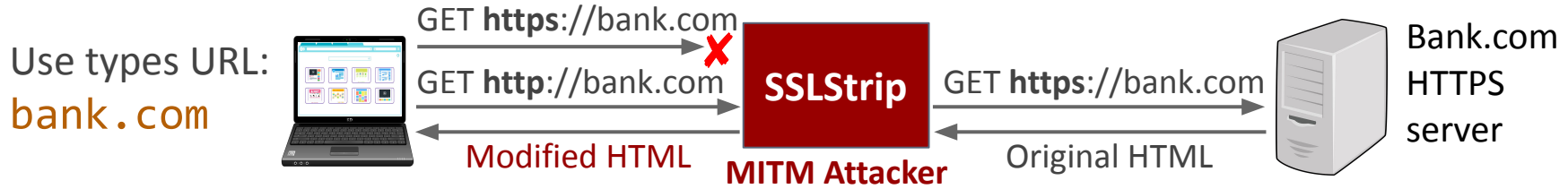
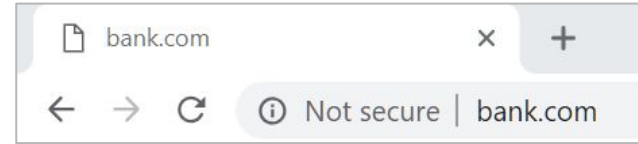
1. Fooling Users: Stripping



Browsers try HTTPS first, but **fall-back to HTTP** unless the URL specifies https://.

This enables the **SSLStrip attack**:

Users may not notice security indicator



Mitigation:

HTTP Strict Transport Security (HSTS)

Server sends special HTTP header:

```
Strict-Transport-Security: max-age=31536000
```

Browser will exclusively use HTTPS for the domain for *max-age* seconds and refuse to bypass certificate errors.

What about **first time** a user visits the domain?

Domains can get on **HSTS Preload List** shipped with browsers.

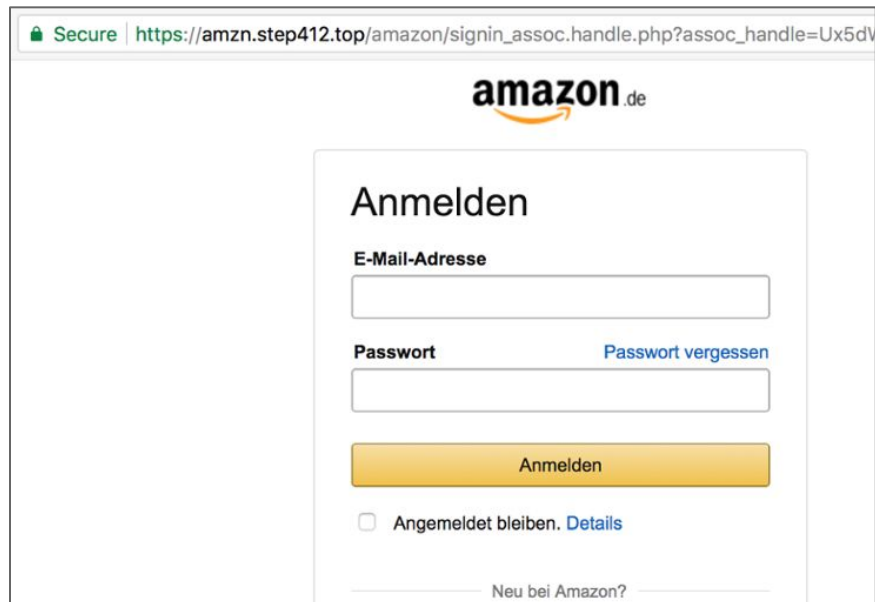
<https://hstspreload.org/>

1. Fooling Users: Phishing



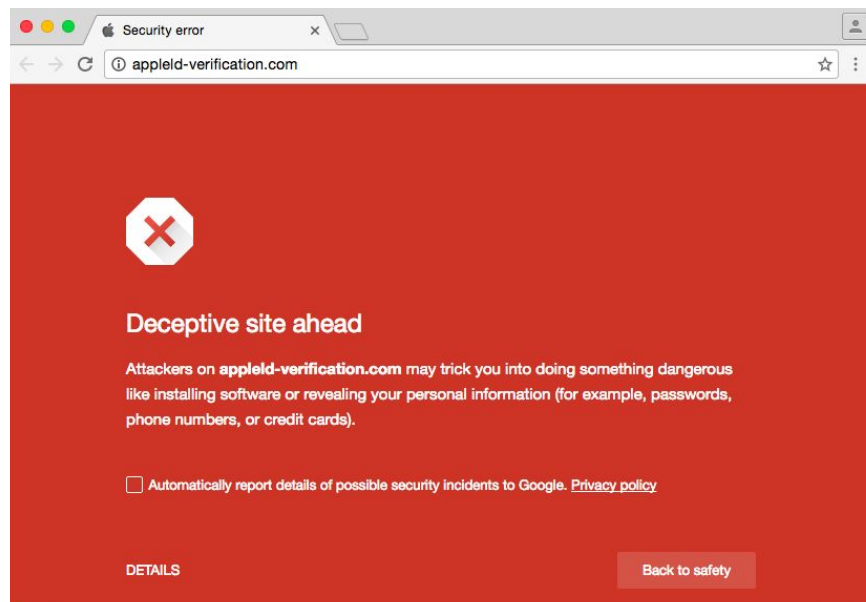
HTTPS cannot prevent phishing

Majority of phishing sites have valid certs.
CAs don't have visibility into site behavior.



Mitigation:

Google Safe Browsing uses ML to identify dangerous sites, warn users.

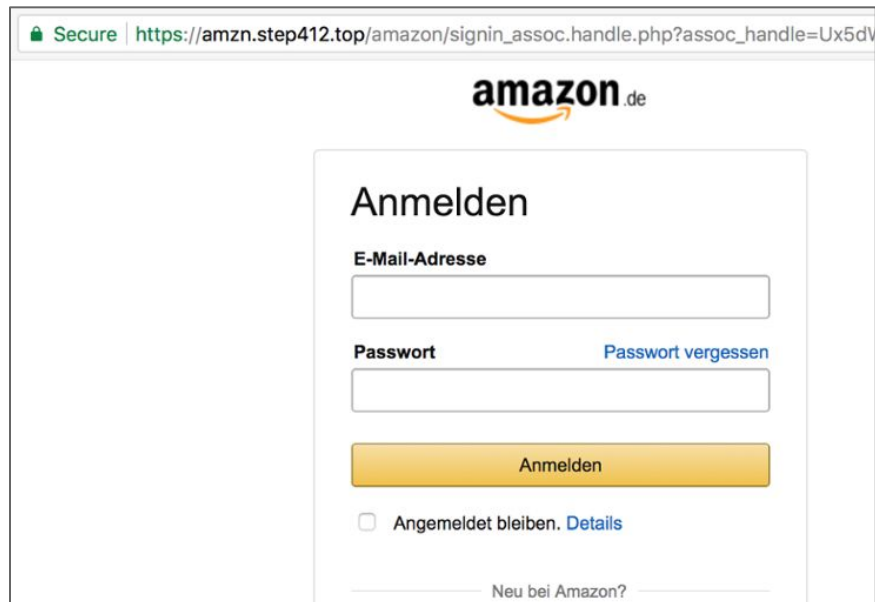


1. Fooling Users: Phishing



HTTPS cannot prevent phishing

Majority of phishing sites have valid certs.
CAs don't have visibility into site behavior.



Mitigation:

Phishing-resistant authentication
such as WebAuthN and Passkeys.

Needs three properties:

1. Non-forwardability

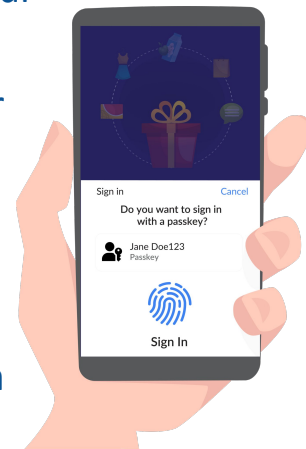
The credential can only be used once, and only used by the creator of the credential

2. Origin binding

The credential can only be used for a single web origin
(attacker.com vs. example.com)

3. Channel binding

The credential is linked to the physical device being used to log in



2. Problems with Site Design



Mixed Content (resources loaded over HTTP from inside an HTTPS page) can be modified or leak information

Browsers block HTTP scripts but do allow images to be loaded over HTTP

Mitigation: Use HTTPS for all resources

HTTPS reveals certain information about a site

TLS: sends **domain** (but not URL) in Client Hello message **Server Name Indication (SNI)** header

Privacy and censorship concerns

By default, **cookies** set over HTTPS will be **sent unencrypted** if the browser later requests an HTTP URL from the domain

Mitigation: Set the **Secure attribute** on cookies, so they will only be sent via HTTPS

```
Set-Cookie: session=<TOKEN>; Secure
```

Mitigation: Clients can use Tor or VPNs for privacy

Mitigation: Emerging standard for **Encrypted Client Hello** can let servers hide domain

3. CA Weaknesses: Fooling Validation



CA identity validation isn't foolproof.

If attacker can **falsely convince a CA** that they control a domain, they can obtain a certificate and fool browsers

Example: **Getting a cert for umich.edu**

Attack 1: (Email validation)

CA emails confirmation code to
`administrator@umich.edu`

Attack 2: (HTTP validation)

Attacker becomes MITM between CA and U-M (e.g., by route hijacking) and redirects CA's GET request to own site

Mitigation:

Domains can **limit which CAs** can issue certs for them by creating a DNS record called

Certification Authority Authorization (CAA):

`eecs388.org. IN CAA 0 issue "letsencrypt.org"`

Mitigation:

Sites must prohibit users from creating these email addresses: administrator, admin, webmaster, hostmaster, postmaster

Mitigation:

CAs such as Let's Encrypt are introducing **multi-perspective validation**, which makes MITM attacks on the CA more difficult

Email Validation



The screenshot displays the 'mcommunity.umich.edu' web application. At the top, a navigation bar includes the user name 'Carson Hoffman' and links for 'My Profile', 'My Groups', 'Log Out', and 'Help'. Below this is the 'M COMMUNITY' logo and a search bar with 'Search' and 'Advanced Search' buttons. The main content area is titled 'administrator' and features two tabs: 'Group Info' (selected) and 'Members'. The 'Group Info' tab contains two sections: 'Contact Information' and 'Group Details'. The 'Contact Information' section lists 'E-Mail: administrator@umich.edu' and 'Requests To: administrator-requests@umich.edu'. The 'Group Details' section includes 'Membership' (0 Members, Expires: May 13, 2022), 'Owners' (J Alex Halderman), and 'Rules Summary' (Owners must add members, Member list is visible to all, Messages can be sent to group by all, Sent to Google UMich: Yes). The footer of the page reads 'University of Michigan | Information and Technology Services © 2022 The Regents of the University of Michigan'.

mcommunity.umich.edu

Carson Hoffman | My Profile | My Groups | Log Out | Help

M COMMUNITY

Search Advanced Search

administrator Group Info Members

Contact Information

E-Mail: administrator@umich.edu
Requests To: administrator-requests@umich.edu

Group Details

Membership
0 Members
Expires: May 13, 2022

Owners
[J Alex Halderman](#)

Rules Summary
Owners must add members
Member list is visible to all
Messages can be sent to group by all
Sent to Google UMich: Yes

University of Michigan | Information and Technology Services
© 2022 The Regents of the University of Michigan

Single vs. Multi-Perspective Domain Validation



Single Perspective Domain Validation

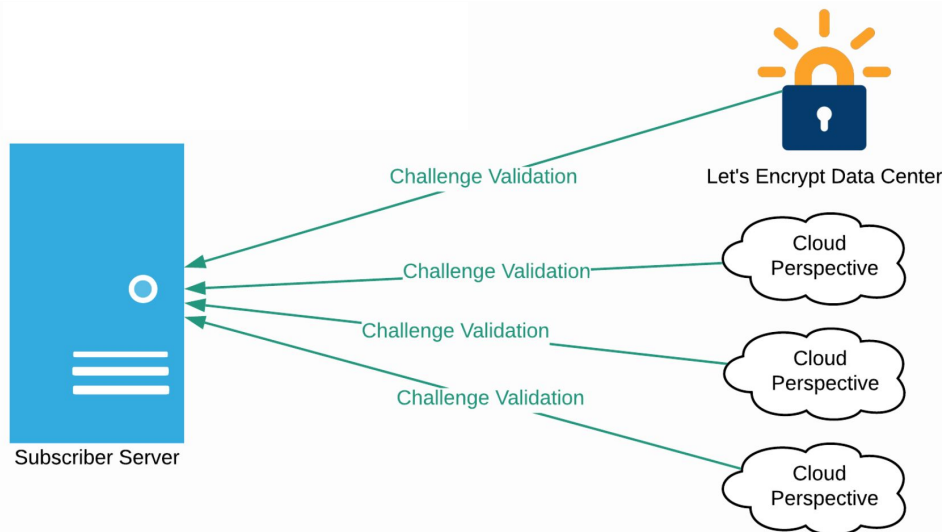
Site must satisfy one challenge validation check from a single CA data center



Multi-Perspective Domain Validation

Site must satisfy several challenge validation checks arriving from multiple network locations

To defeat, adversary would have to eavesdrop on all of the network paths



3. CA Weaknesses: Attacks on CAs



Web PKI uses a distributed architecture.
Thousands of intermediate CAs can issue certificates for any domain

Occasionally CAs get hacked.

In 2011, a cert for *.google.com was issued to an attacker who broke into **DigiNotar**, a small Dutch CA

Cert was used for MITM attacks in Iran

Nobody noticed the attack until somebody found the certificate in the wild...

Google, Microsoft, Apple and Mozilla distrusted all DigiNotar certs, and the CA ceased operation

Mitigation:

CAs can **revoke** (i.e., cancel) certs they issue by adding them to a **Certificate Revocation List (CRL)** that they sign and publish

Browser companies crawl CRLs and push updates to users

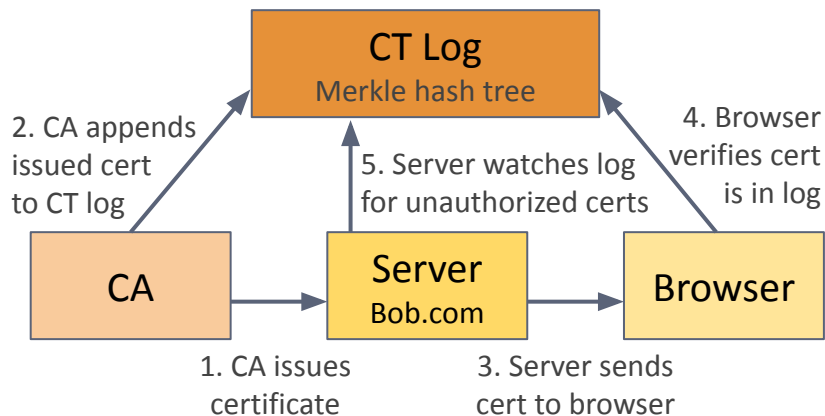
Mitigation:

Certificate Transparency helps ensure that compromises will be discovered quickly

Mitigation: Certificate Transparency



Browsers require CAs to record every cert they issue in a public ledger, called a **Certificate Transparency (CT) log**



Servers can **monitor** CT logs to detect if any CA issues an improper cert for their names

The screenshot shows the crt.sh Identity Search tool interface. The search criteria are set to "Identity" and "Match: ILIKE" with the search term "eecs388.org". The results table lists certificates with columns: crt.sh ID, Logged At, Not Before, Not After, Common Name, Matching Identities, and Issuer Name.

Certificates	crt.sh ID	Logged At	Not Before	Not After	Common Name	Matching Identities	Issuer Name
	6094181323	2022-01-31	2022-01-31	2022-05-01	project2.eecs388.org	project2.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6077263407	2022-01-31	2022-01-31	2022-05-01	project2.eecs388.org	project2.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6076922893	2022-01-31	2022-01-31	2022-05-01	files.eecs388.org	files.eecs388.org	C=US, O=Google Trust Services LLC, CN=GTS CA 1D4
	6076201879	2022-01-28	2022-01-28	2022-04-28	kvm.eecs388.org	kvm.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6063570494	2022-01-28	2022-01-28	2022-04-28	kvm.eecs388.org	kvm.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053130883	2022-01-26	2022-01-26	2022-04-26	test4.eecs388.org	test4.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053093352	2022-01-26	2022-01-26	2022-04-26	test4.eecs388.org	test4.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053178098	2022-01-26	2022-01-26	2022-04-26	project1.eecs388.org	project1.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053180470	2022-01-26	2022-01-26	2022-04-26	test3.eecs388.org	test3.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6052845260	2022-01-26	2022-01-26	2022-04-26	project1.eecs388.org	project1.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053093347	2022-01-26	2022-01-26	2022-04-26	test3.eecs388.org	test3.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053178054	2022-01-26	2022-01-26	2022-04-26	deploy.eecs388.org	deploy.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053177967	2022-01-26	2022-01-26	2022-04-26	stats.eecs388.org	stats.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6052845289	2022-01-26	2022-01-26	2022-04-26	deploy.eecs388.org	deploy.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053183559	2022-01-26	2022-01-26	2022-04-26	test2.eecs388.org	test2.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6052902674	2022-01-26	2022-01-26	2022-04-26	stats.eecs388.org	stats.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6053093285	2022-01-26	2022-01-26	2022-04-26	test2.eecs388.org	test2.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6042561717	2022-01-24	2022-01-24	2022-04-24	www.eecs388.org	www.eecs388.org	C=US, O=Google Trust Services LLC, CN=GTS CA 1D4
	5918401470	2022-01-04	2022-01-04	2022-04-04	test2.eecs388.org	test2.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5918395588	2022-01-04	2022-01-04	2022-04-04	test2.eecs388.org	test2.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5918401827	2022-01-04	2022-01-04	2022-04-04	test4.eecs388.org	test4.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5918401754	2022-01-04	2022-01-04	2022-04-04	test4.eecs388.org	test4.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5918401281	2022-01-04	2022-01-04	2022-04-04	test3.eecs388.org	test3.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5918401438	2022-01-04	2022-01-04	2022-04-04	test3.eecs388.org	test3.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5893708741	2021-12-31	2021-12-31	2022-03-31	project1.eecs388.org	project1.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5893708518	2021-12-31	2021-12-31	2022-03-31	project1.eecs388.org	project1.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5893741222	2021-12-24	2021-12-24	2022-03-24	project4.eecs388.org	project4.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5893740202	2021-12-24	2021-12-24	2022-03-24	project4.eecs388.org	project4.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5818836690	2021-12-17	2021-12-17	2022-03-17	deploy.eecs388.org	deploy.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5816279308	2021-12-17	2021-12-17	2022-03-17	deploy.eecs388.org	deploy.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5817458407	2021-12-17	2021-12-17	2022-03-17	eecs388.org	eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5814205519	2021-12-17	2021-12-17	2022-03-17	eecs388.org	eecs388.org	C=US, O=Let's Encrypt, CN=R3
	5807280072	2021-12-14	2021-12-14	2022-03-14	stats.eecs388.org	stats.eecs388.org	C=US, O=Let's Encrypt, CN=R3
	6706503233	2021-12-14	2021-12-14	2022-03-14	stats.eecs388.org	stats.eecs388.org	C=US, O=Let's Encrypt, CN=R3

crt.sh CT log search tool

Be aware: Due to CT, your domain will be publicly visible as soon as you acquire a cert

4. Bugs in TLS Implementations



TLS is a complicated protocol.
Security bugs can be hard to
spot during routine testing

Apple Goto Fail (2014)

Apple TLS libraries skipped certificate
checking for almost a year due to a
stray goto statement

Mozilla BERsek (2014)

Bug in verifying certificate signatures
allowed spoofing certs, probably
since the beginning...

```
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(ctx,
    ctx->peerPubKey,
    dataToSign,
    dataToSignLen,
    signature,
    signatureLen);
if (err) {
    sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
        "returned %d\n", int(err))
    goto fail;
}
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

4. Bugs in TLS Implementations



TLS is a complicated protocol.
Security bugs can be hard to spot during routine testing

Apple Goto Fail (2014)

Apple TLS libraries skipped certificate checking for almost a year due to a stray goto statement

Mozilla BERsek (2014)

Bug in verifying certificate signatures allowed spoofing certs, probably since the beginning...

Null Prefix Attack (2009)

Most browsers use C-style strings (null terminated), but X.509 uses Pascal-style strings (length field, then value)

What if a domain name in a certificate contains “\0”?

`gmail.com\0.badguy.com`

CA validates `badguy.com` (Pascal string)

Browser thinks cert is for `gmail.com` (C string)!

Mitigation:

Formal verification techniques can be used to check software correctness using a proof

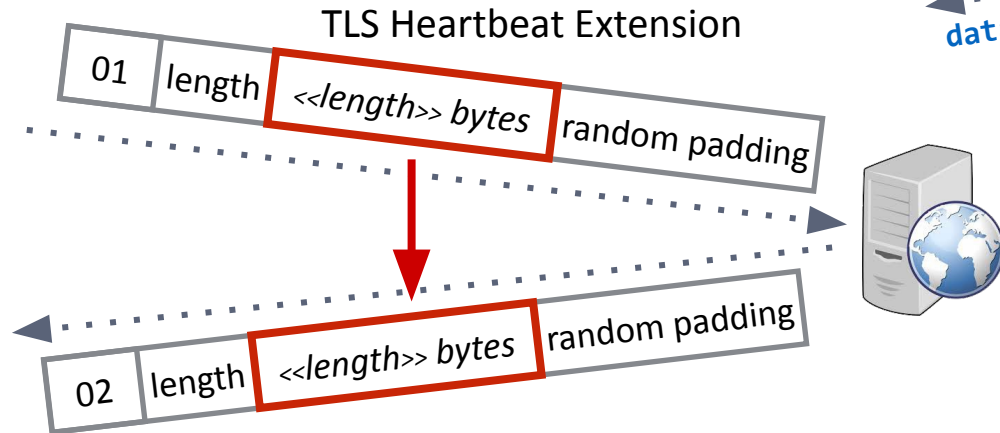
miTLS is a formally verified TLS library

4. Bugs in TLS Implementations

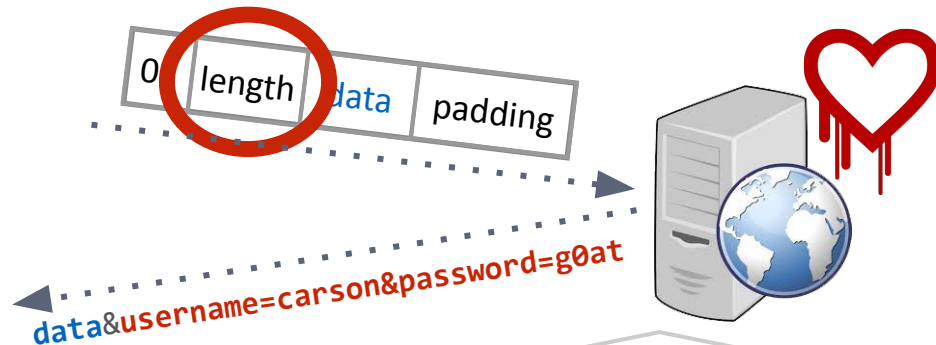


OpenSSL Heartbleed (2014)

Catastrophic bug in widely used TLS library allowed attackers to **dump process's memory**.
Leaked private keys, passwords, other secrets
24–55% of all HTTPS websites were vulnerable



Vulnerability: OpenSSL trusted user-provided length field and echoed back memory following request data



```
8t30s05eRv8ilIFhm4qpdc8t9xTTBZdata&  
username=carson&password=g0atlnI1c9  
rX7ZayyY2N0H72MngCOUuWIogpPuRab293j
```

Mitigation: Update server software.
Prefer TLS libraries written in
memory-safe languages

5. TLS Protocol Vulnerabilities



TLS versions <1.3 have many known weaknesses

RC4-related Attacks

Biases in RC4 stream cipher can be used to leak cookies and passwords

CBC-related Attacks

BEAST and POODLE exploit weaknesses in TLS's use of CBC to leak data

Compression-related Attacks

TLS and HTTP support compression, but TLS exposes length of the plaintext. The combination can leak data

Example: Attacker causes user's browser to visit crafted URLs on a target site. If URLs match user's cookie, length of TLS data will be shorter. Use many requests to iteratively leak cookie

Export-related Attacks

FREAK, Logjam, and DROWN exploit weaknesses in 1990s-era "export-grade" cryptography. Allowed MITM attacks against $\approx 30\%$ of popular modern sites

Mitigation: Safeguarding TLS <1.3 requires updates to servers and/or browsers

TLS Compression Attack



HTTP Request

GET /abcd1234

Cookie: session=wxyz5678

Compression



GET /abcd1234 Cookie: session=wxyz5678

Encryption



2e47f5729b2d09a1b1f037bd5451d912b3aa69

Transmission



GET /wxyz1234

Cookie: session=wxyz5678



GET /wxyz1234 Cookie: session=wxyz5678



279d834608a0ab3a9ed6e4c6c19b1b1c883f



GET /wxyz5678

Cookie: session=wxyz5678



GET /wxyz5678 Cookie: session=wxyz5678



b523e908c59c81236c2095ef100809e7b



Length of the ciphertext can leak information about the plaintext!



Until 2000, U.S. law limited strength of cryptography sold overseas to:

- 64-bit symmetric keys
- 512-bit RSA or DH

1990s browsers came in U.S. versions and “export” versions, with weakened crypto. SSL design let servers and clients negotiate the strongest keys they both supported

Limits were lifted following legal challenge by cryptographer Dan Bernstein and EFF, arguing that code is a form of free speech

No modern clients support export-grade crypto, but remnants remained in protocol design and implementations until TLS 1.3

Court Hears Appeal in Encryption Case

By JOHN MARKOFF DEC. 9, 1997

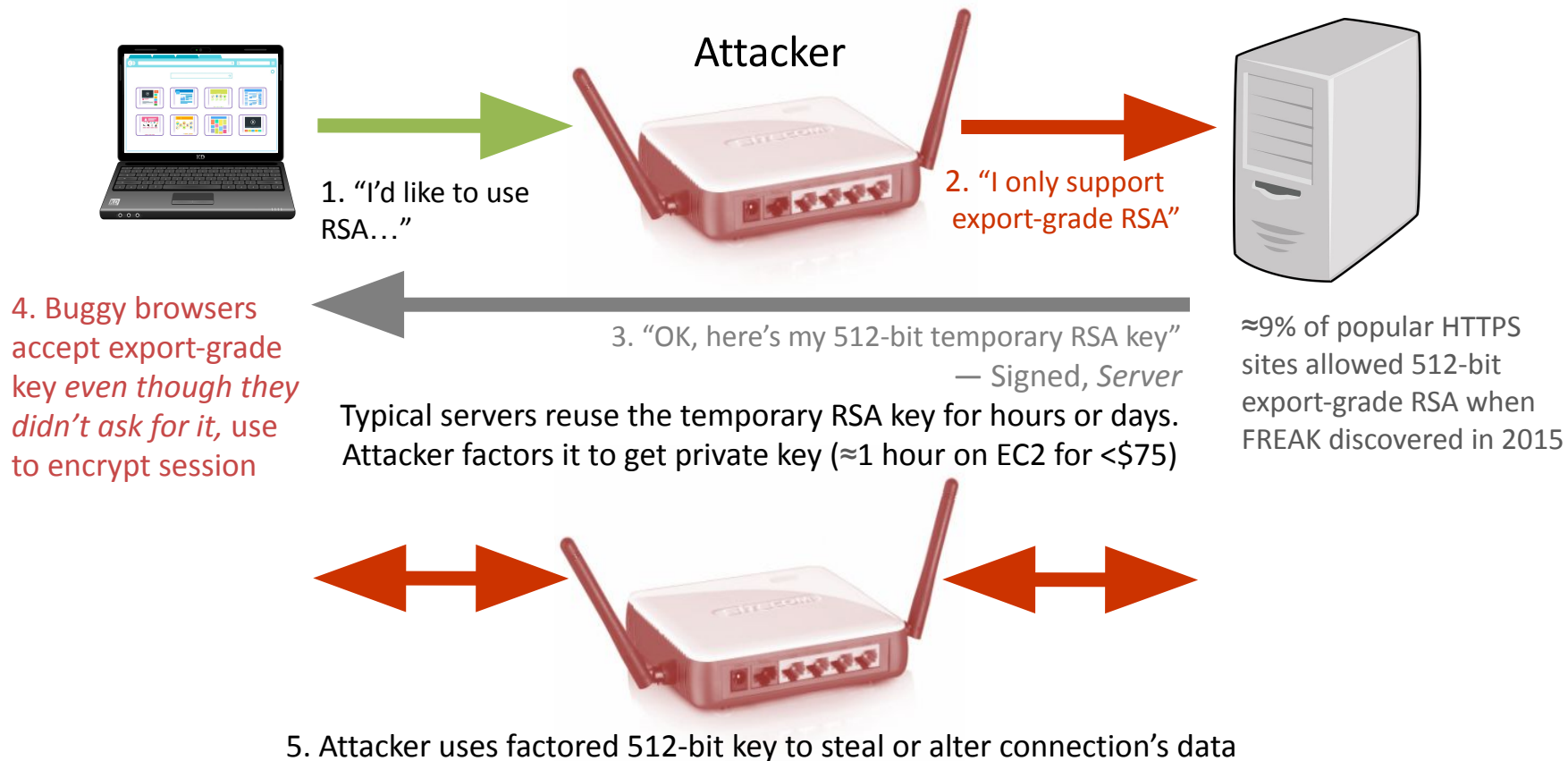
The power of the Internet to transcend national borders pitted personal rights against national security today when a Federal appeals court heard arguments in a suit contending that Government export controls on data-scrambling software illegally restrict free speech.

In a case that could have a profound effect on the future of electronic commerce and banking, a three-judge panel of the Court of Appeals for the Ninth Circuit here heard the Government's appeal of a December 1996 decision in which Judge Marilyn Hall Patel of Federal District Court ruled that Government attempts to control the export of encryption software were unconstitutional.

Her ruling came in a suit filed in February 1995 by Daniel J. Bernstein, then a graduate student at the University of California at Berkeley, after State Department officials said he would be required to register as a munitions dealer and secure an arms-trading license to export an electronic version of

The FREAK Attack

Deliberately weakened crypto from the 1990s harmed global security in 2015



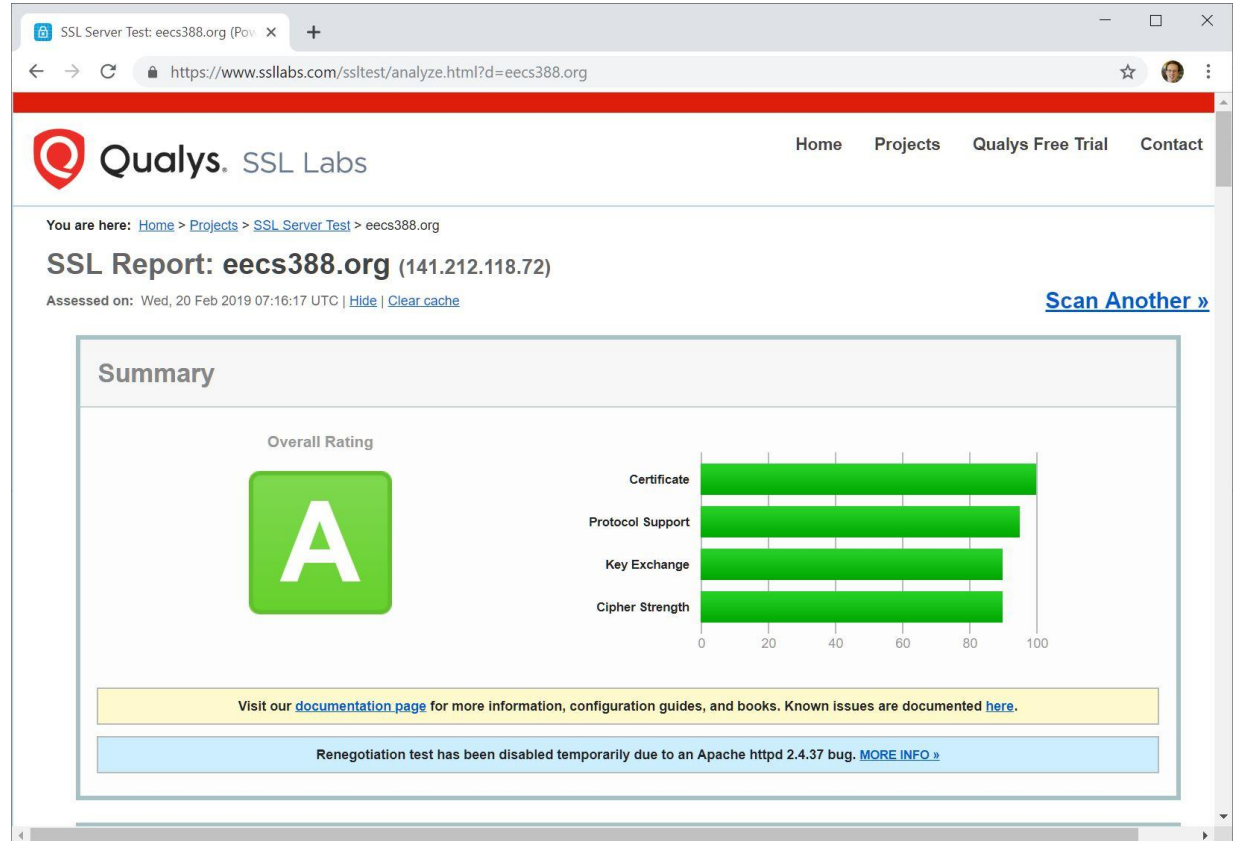
5. TLS Protocol Vulnerabilities



TLS server software is often insecure in its default configuration. Common speedups, like session resumption, can also weaken security.

Mitigation:

Use tools such as **SSL Labs** to test your server, and apply mitigations.



6. Server Vulnerabilities



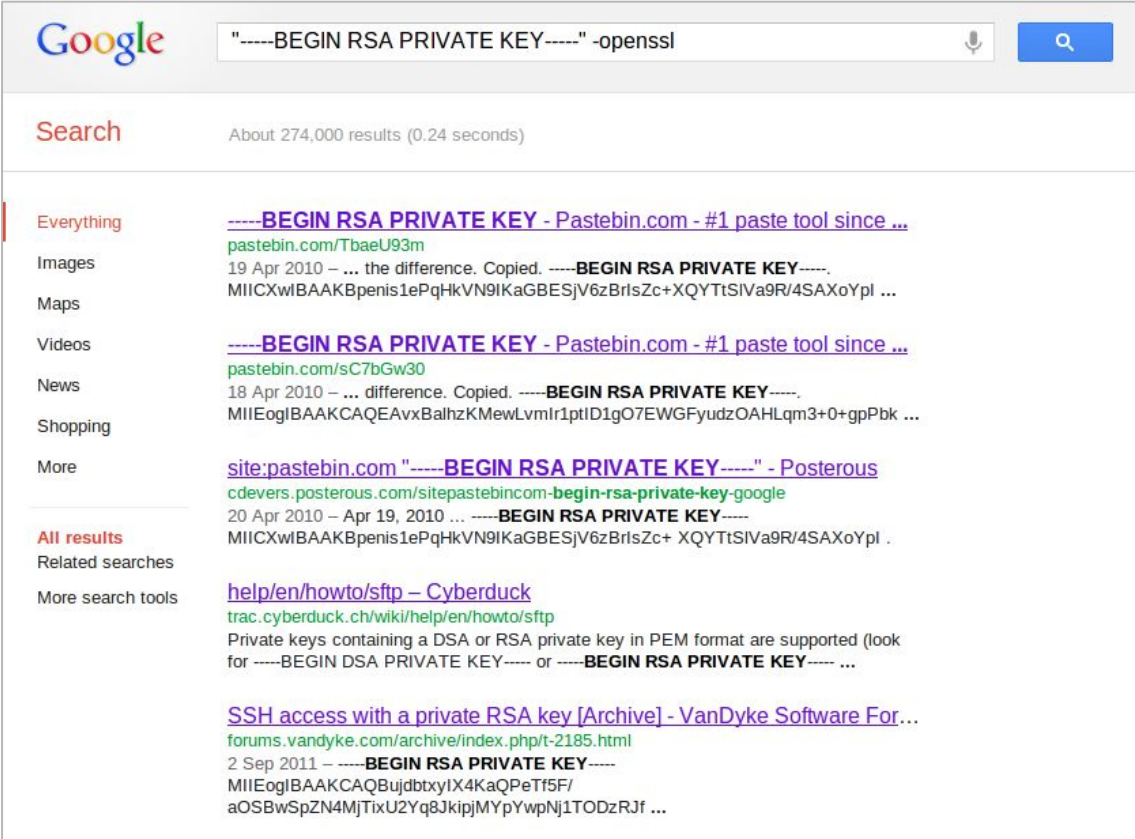
TLS servers must
protect their private keys

File permission errors,
unencrypted backups,
and intrusions can result in
theft of private keys

Mitigation:

Apply best practices of
isolation and access control.

If keys leak, contact CA to
have certificate revoked.



The screenshot shows a Google search interface with the query "-----BEGIN RSA PRIVATE KEY-----" -openssl. The search results page displays several links, including multiple references to Pastebin.com and Posterous, as well as a link to Cyberduck's help page. The results indicate that private keys in PEM format are supported and provide examples of the key format.

Google

"-----BEGIN RSA PRIVATE KEY-----" -openssl

Search

About 274,000 results (0.24 seconds)

Everything

Images

Maps

Videos

News

Shopping

More

All results

Related searches

More search tools

-----BEGIN RSA PRIVATE KEY - Pastebin.com - #1 paste tool since ...
pastebin.com/TbaeU93m
19 Apr 2010 - ... the difference. Copied. -----BEGIN RSA PRIVATE KEY-----.
MIICXwlBAAKBpenis1ePqHkVN9lKaGBESjV6zBrlsZc+XQYTtSIVa9R/4SAXoYpl ...

-----BEGIN RSA PRIVATE KEY - Pastebin.com - #1 paste tool since ...
pastebin.com/sC7bGw30
18 Apr 2010 - ... difference. Copied. -----BEGIN RSA PRIVATE KEY-----.
MIIEoglBAAKCAQEAvxBalzKMewLvmIr1ptlD1gO7EWGFyudzOAHlQm3+0+gpPbk ...

site:pastebin.com "-----BEGIN RSA PRIVATE KEY-----" - Posterous
cdevers.posterous.com/sitepastebincom-begin-rsa-private-key-google
20 Apr 2010 - Apr 19, 2010 ... -----BEGIN RSA PRIVATE KEY-----
MIICXwlBAAKBpenis1ePqHkVN9lKaGBESjV6zBrlsZc+ XQYTtSIVa9R/4SAXoYpl .

help/en/howto/sftp - Cyberduck
trac.cyberduck.ch/wiki/help/en/howto/sftp
Private keys containing a DSA or RSA private key in PEM format are supported (look
for -----BEGIN DSA PRIVATE KEY----- or -----BEGIN RSA PRIVATE KEY----- ...

SSH access with a private RSA key [Archive] - VanDyke Software For...
forums.vandyke.com/archive/index.php/t-2185.html
2 Sep 2011 - -----BEGIN RSA PRIVATE KEY-----
MIIEoglBAAKCAQBujdbtxylX4KaQPETf5F/
aOSBwSpZN4MjTixU2Yq8JkipjMYpYwpNj1TODzRjf ...

Coming Up



Reminders:

Lab Assignment 2 due TODAY at 6 PM

Project 2 due next Thursday, October 3, at 6 PM

Midterm Exam is Friday, October 18, 7-8:30 PM

Tuesday

Networking 101

Protocol layers, Ethernet, IP, routing

Thursday

Networking 102

TCP, UDP, and DNS attacks