

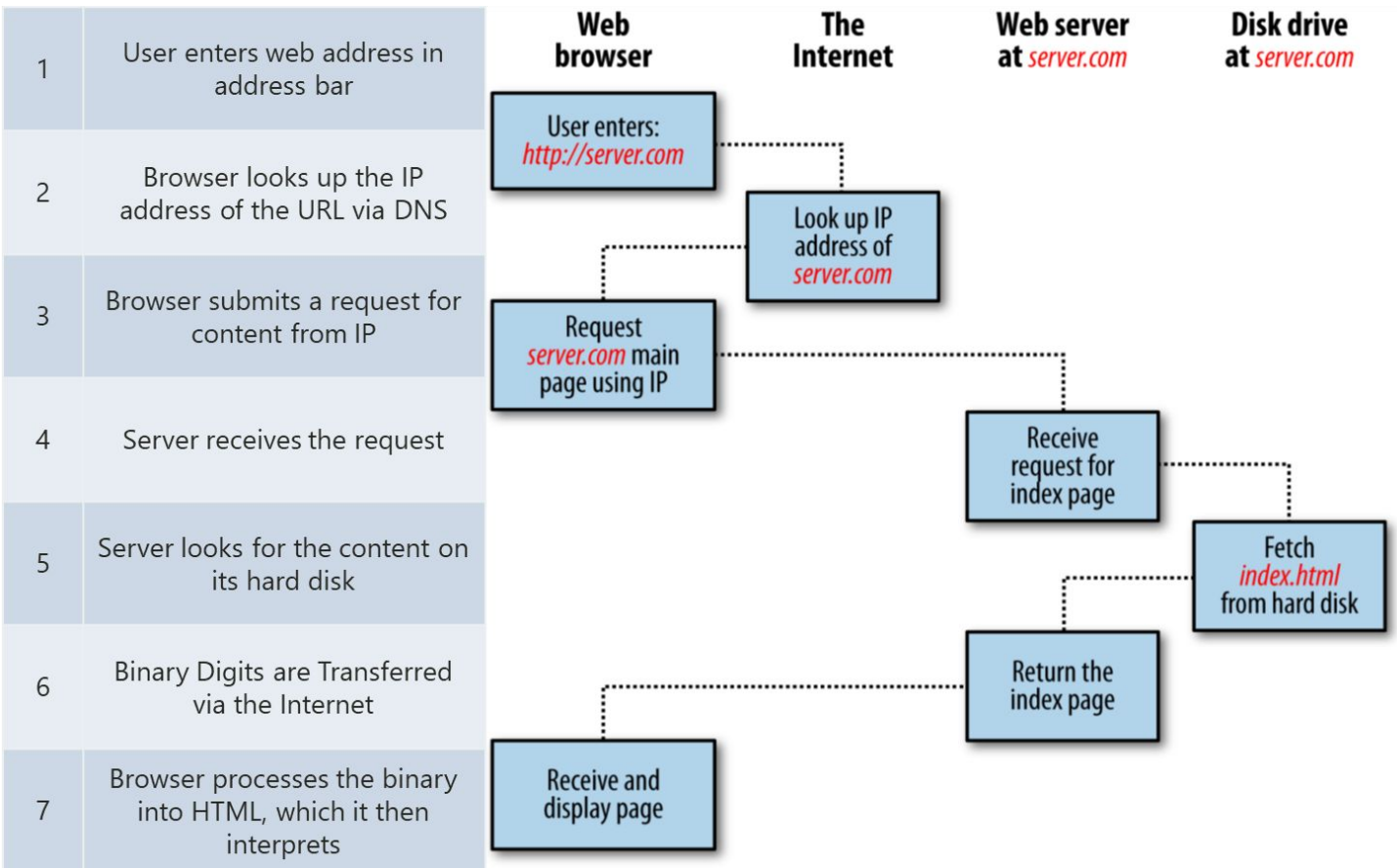


CSC 405

SSL/TLS & HTTPS

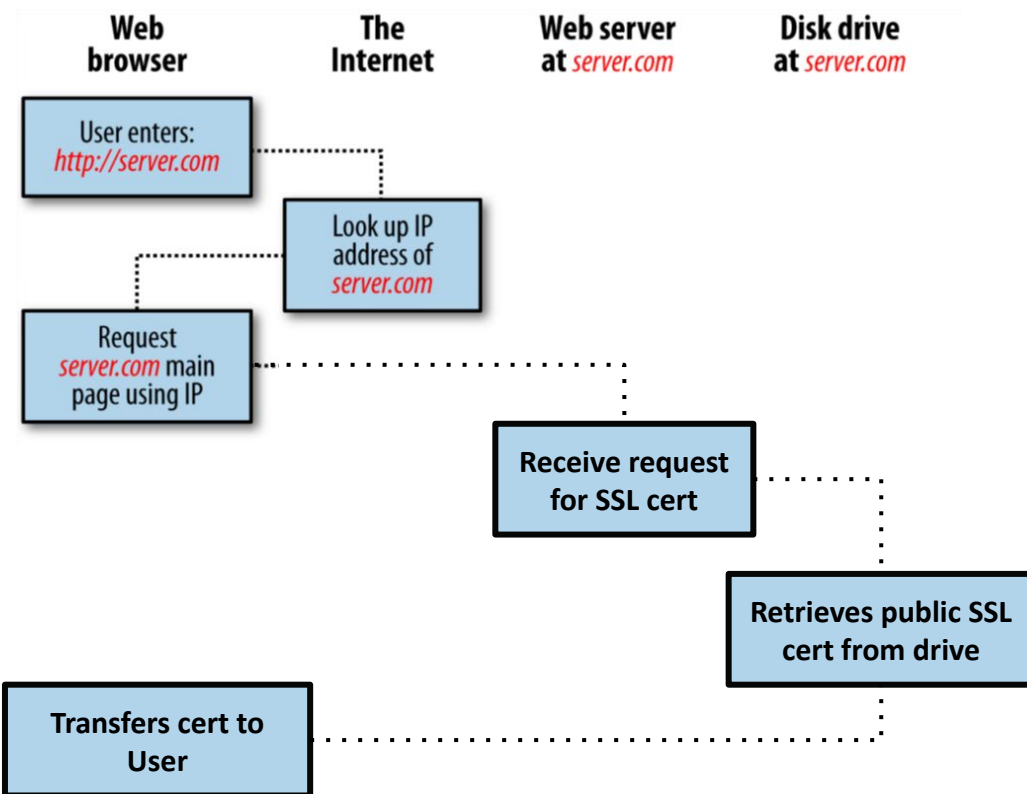
Alexandros Kapravelos
akaprav@ncsu.edu

HTTP Workflow



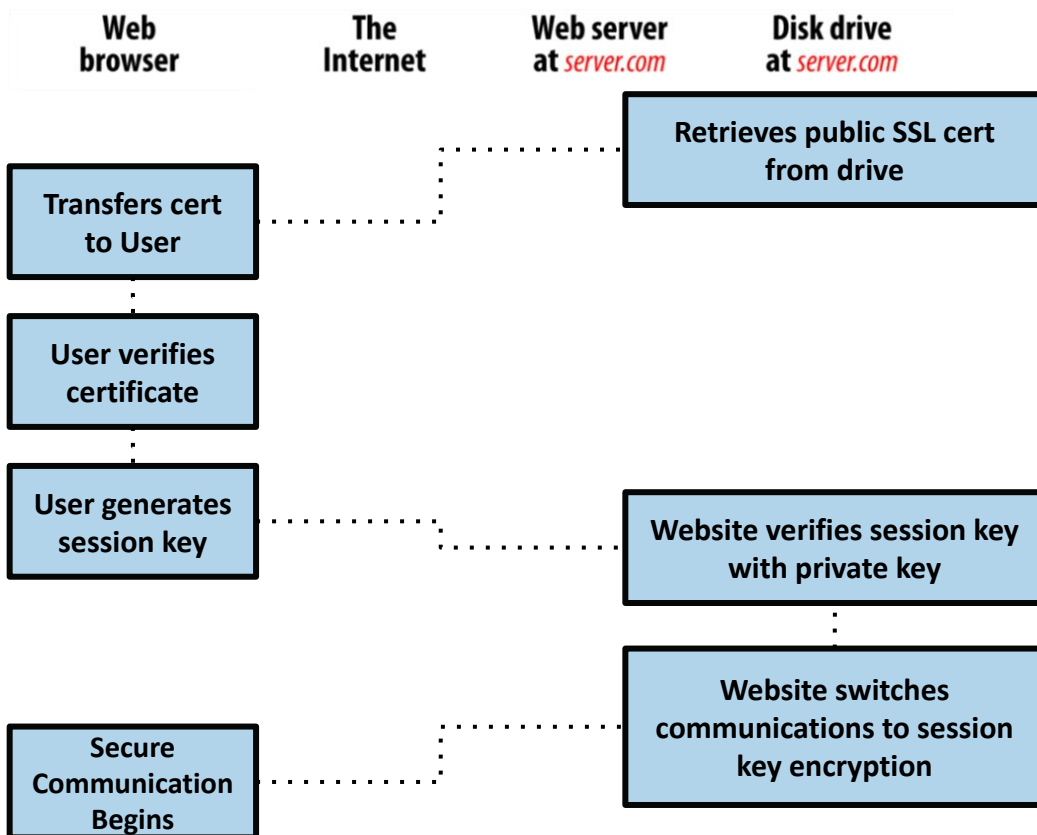
HTTPS Workflow

1	User enters web address in address bar
2	Browser looks up IP address of the URL via DNS
3	Browser submits request for SSL Connect from Website
4	Website responds with an SSL certificate



HTTPS Workflow

5	User verifies SSL certificate is issued to website and not expired
6	User generates a random number
7	User encrypts session key with public key
8	Website decrypts the session key with their private key
9	"Secure" communication can now occur between the two



SSL vs TLS

- **SSL (Secure Sockets Layer):** Developed by Netscape in the mid-1990s. SSL versions 1.0 (never publicly released), 2.0, and 3.0 were created.
- **TLS (Transport Layer Security):** When SSL 3.0 was found to have security weaknesses, the Internet Engineering Task Force (IETF) took over development, improved upon it, and released it under the new name TLS.
 - TLS 1.0 was released in 1999, essentially an upgrade of SSL 3.0.
 - TLS 1.1, 1.2, and the current strongest version, **TLS 1.3** (released in 2018), followed, each bringing significant security and performance improvements.

Creating the Certificate

Step One: Generate a Certificate Signing Request (CSR)

```
openssl req -nodes -newkey rsa:2048 -keyout myserver.key -out server.csr
```

You are about to be asked to enter information that will be incorporated into your certificate request.

...

Country Name (2 letter code) [AU]:**US**

State or Province Name (full name) [Some-State]:**NC**

Locality Name (eg, city) []:**Raleigh**

Organization Name (eg, company) [Internet Widgits Pty Ltd]:**NC State University**

Organizational Unit Name (eg, section) []:**HackPack**

Common Name (e.g. server FQDN or YOUR name) []:**Hack T. Pack**

Email Address []:**hackpackclub@ncsu.edu**

Please enter the following 'extra' attributes
to be sent with your certificate request

A challenge password []:

An optional company name []:

Creating the Certificate

Step One: Generate a Certificate Signing Request (CSR)

```
openssl req -nodes -newkey rsa:2048 -keyout myserver.key -out server.csr
```

This will generate two files with RSA-2048 encryption



myserver.key

server.csr

Creating the CSR

Step One: Generate a Certificate Signing Request (CSR)

```
openssl req -nodes -newkey rsa:2048 -keyout myserver.key -out server.csr
```

This will generate two files with RSA-2048 encryption



myserver.key

server.csr

The .key file becomes your internal private key for encryption

Creating the CSR

Step One: Generate a Certificate Signing Request (CSR)

```
openssl req -nodes -newkey rsa:2048 -keyout myserver.key -out server.csr
```

This will generate two files with RSA-2048 encryption



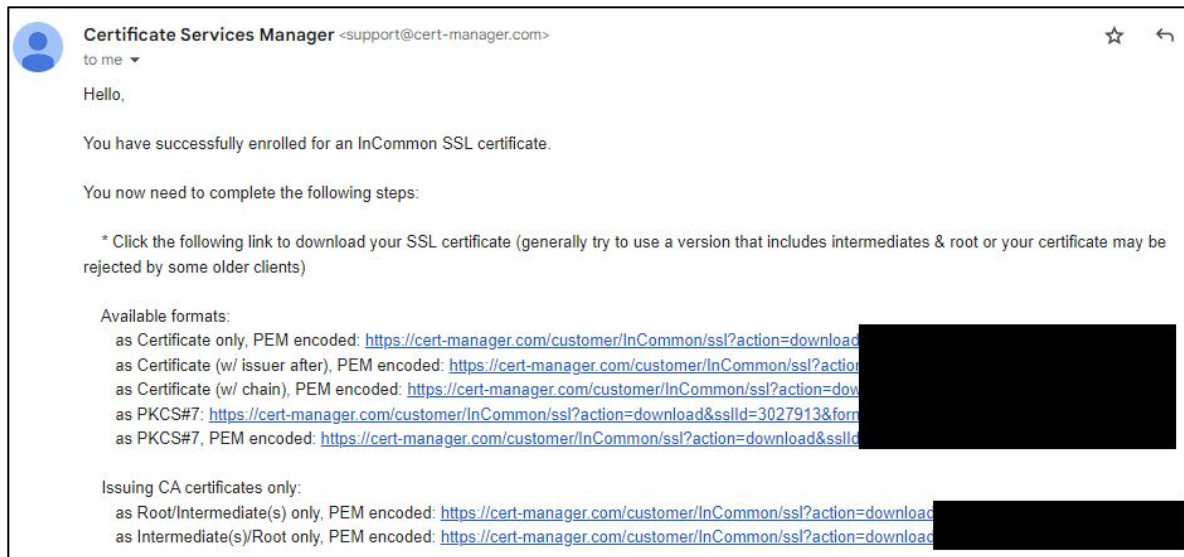
myserver.key

server.csr

The .csr file contains the information entered during creation

Submitting Your CSR

- The CSR file is then submitted to a **Certificate Authority**
 - These entities in turn verify the certificate for users and the server
 - Once verified by the CA, the registering party will receive a signed version of the certificate



Submitting Your CSR

- The CSR file is then submitted to a **Certificate Authority**
 - These entities in turn verify the certificate for users and the server
 - Once verified by the CA, the registering party will receive a signed version of the certificate



The Foundation of Trust: Certificate Authorities (CAs)

- **Who are they?** CAs are trusted third-party organizations (e.g., Let's Encrypt, DigiCert, GlobalSign).
- **What do they do?** They issue digital certificates (like SSL/TLS certificates).
- **Core Function:** To verify the identity of an entity (like a website owner) and bind that identity to a cryptographic public key.

How CAs Enable Secure Connections (TLS/SSL)

1. **Verification:** A website owner proves their identity and control over a domain to a CA.
2. **Issuance:** The CA issues a certificate containing the website's domain name, public key, and other details. This certificate is digitally signed by the CA using its private key.
3. **Browser Trust:** Your web browser and operating system come pre-loaded with a list of trusted Root CAs and their public keys (the "Trust Store").
4. **Connection:** When you visit an HTTPS website:
 5. The website presents its certificate.
 6. Your browser checks if the certificate was signed by a CA in its Trust Store.
 7. It verifies the signature using the CA's public key.
 8. It checks if the certificate is valid (not expired, revoked) and matches the domain name.
9. **Result:** If everything checks out, the browser trusts the server's identity and establishes an encrypted connection. This system allows millions of websites to be trusted without prior direct relationships.

Upload Your Key and Cert to the Server

- Transfer the `myserver.key` file to your server
 - Typically stored somewhere like `/etc/ssl/`

```
[user@server ~] ls /etc/ssl/key  
myserver.key
```

Upload Your Key and Cert to the Server

- Transfer the signed certificate files to your server
 - Typically `domainName.crt` and `domainName.ca-bundle`

```
[user@server ~] ls /etc/ssl/cert
domainName.crt
domainName.ca-bundle
```

Configure Your Server

- This will range from the application running your server (Apache, nginx, etc.)

```
[user@server ~] cat /etc/httpd/conf/httpd.conf
...
# Load config files in "/etc/httpd/conf.d" directory, if any.
IncludeOptional conf.d/*.conf
<VirtualHost *:80>
    ServerName domainname.tld
    Redirect "/" "https://domainname.tld/"
</VirtualHost>
```

Apache Configuration

Configure Your Server

- This will range from the application running your server (Apache, nginx, etc.)

```
[user@server ~] cat /etc/httpd/conf/httpd.conf
...
# Load config files in "/etc/httpd/conf.d" directory, if any.
IncludeOptional conf.d/*.conf
<VirtualHost *:80>
    ServerName domainname.tld
    Redirect "/" "https://domainname.tld/"
</VirtualHost>
```

Establish that requests occurring from Port 80 should be redirected to the HTTPS address (Port 443)

Configure Your Server

- This will range from the application running your server (Apache, nginx, etc.)

```
[user@server ~] cat /etc/httpd/conf.d/ssl.conf
<VirtualHost _default_:443>
  ServerName domainName:65432
  SSLEngine on
  SSLCertificateFile /etc/ssl/cert/domainName.crt
  SSLCertificateKeyFile /etc/ssl/key/myserver.key
  ...
  ProxyPass / uwsgi://localhost:65432/
  ProxyPassReverse / uwsgi://localhost:65432/
</VirtualHost>
```

Configure Your Server

- This will range from the application running your server (Apache, nginx, etc.)

```
[user@server ~] cat /etc/httpd/conf.d/ssl.conf
```

```
<VirtualHost _default_:443>
```

```
ServerName domainName:65432
```

```
SSLEngine on
```

```
SSLCertificate
```

```
SSLCertificate
```

```
...
```

```
ProxyPass / uwsgi://localhost:65432/
```

```
ProxyPassReverse / uwsgi://localhost:65432/
```

```
</VirtualHost>
```

Now, communications occur via the 443 port, which can in turn redirect traffic to internal applications or /var/www/html

Configure Your Server

- This will range from the application running your server (Apache, nginx, etc.)

```
[user@server ~] cat /etc/httpd/conf.d/ssl.conf
<VirtualHost _default_:443>
  ServerName domainName:65432
  SSLEngine on
  Congratulations! You're website is HTTPS!
  SSLCertificateFile /etc/ssl/cert/domainName.crt
  SSLCertificateKeyFile /etc/ssl/key/myserver.key
  ...
  ProxyPass / uwsgi://localhost:65432/
  ProxyPassReverse / uwsgi://localhost:65432/
</VirtualHost>
```

Configure Your Server

- This will range from the application running your server (Apache, nginx, etc.)

```
[user@server ~] systemctl restart httpd
```

Now restart Apache...

The Life of a Computer Scientist...

- This will range from the application running your server (Apache, nginx, etc.)



This site can't provide a secure connection

domainName.tld sent an invalid response.

ERR_SSL_PROTOCOL_ERROR

...and debug whatever
you broke 😏

Let's Encrypt

Literally no reason to **not** have SSL encryption on your site



Let's Encrypt

<https://letsencrypt.org/>

My HTTP website is running

Software

Software

Apache

Nginx

HAProxy

Plesk

Other

Web Hosting Product

on

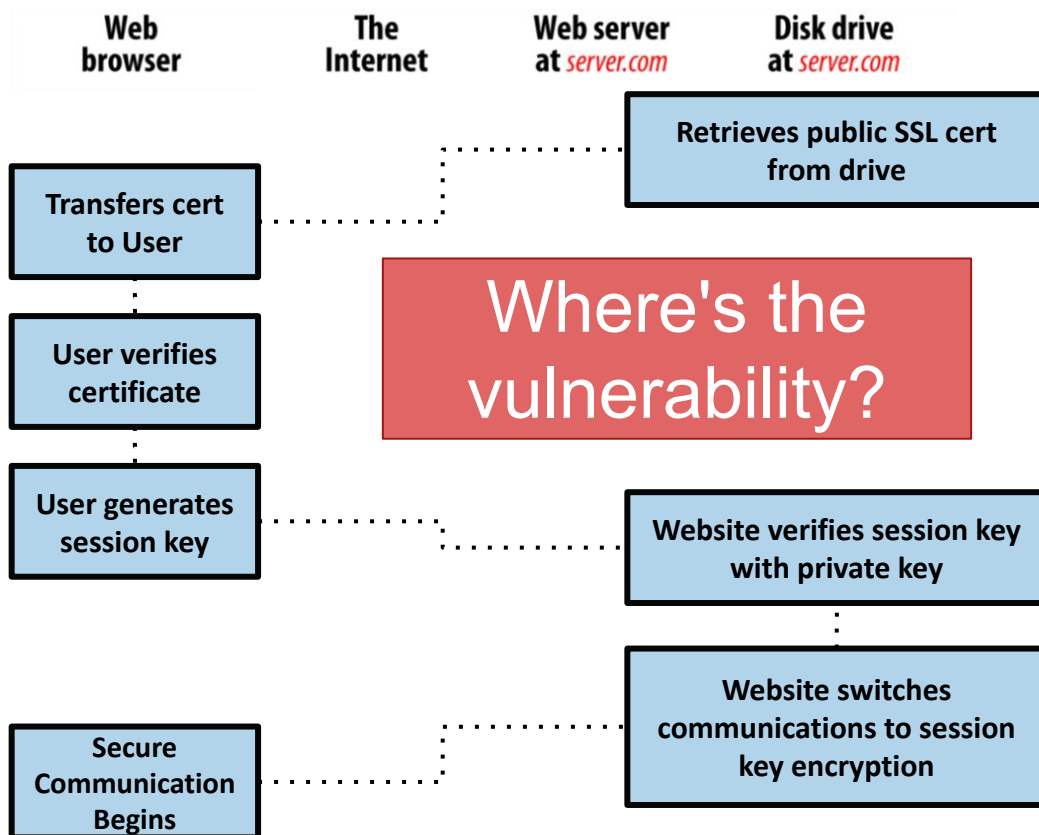
System

[Help, I'm not sure!](#)

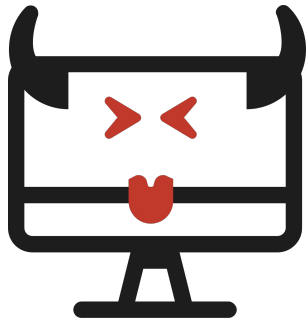
```
[user@server ~] snap install core
[user@server ~] snap refresh core
[user@server ~] snap install --classic certbot
[user@server ~] ln -s /snap/bin/certbot /usr/bin/certbot
[user@server ~] certbot --apache
```

HTTPS Workflow

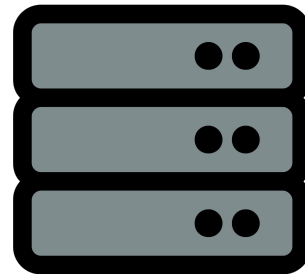
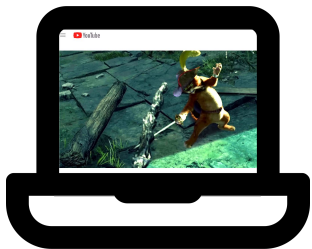
5	User verifies SSL certificate is issued to website and not expired
6	User generates a random number
7	User encrypts session key with public key
8	Website decrypts the session key with their private key
9	"Secure" communication can now occur between the two



SSL Hijacking

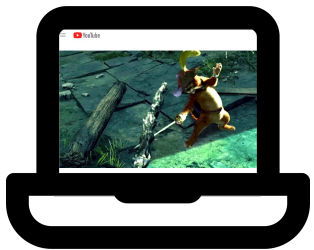
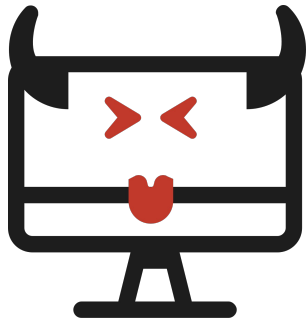


Attacker sends a phishing attack utilizing JavaScript to install a bogus CA certificate



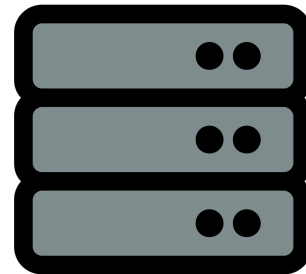
business.com

SSL Hijacking



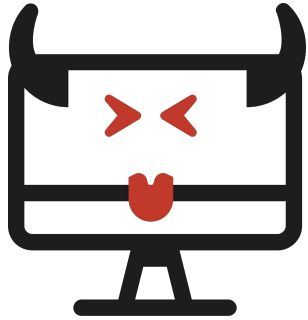
Known as
"DNS Poisoning"

The user's DNS caches are poisoned to make the user's browser route traffic to business.com to attacker's IP Address

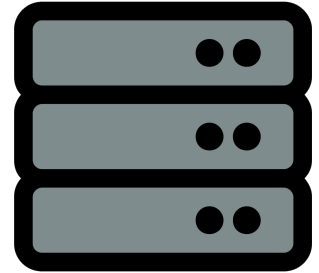
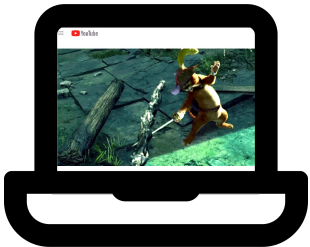
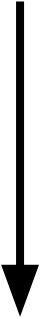


business.com

SSL Hijacking

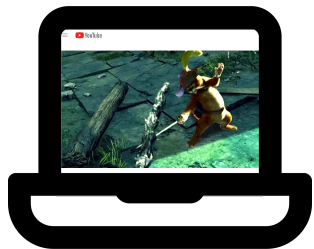
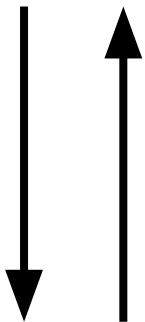
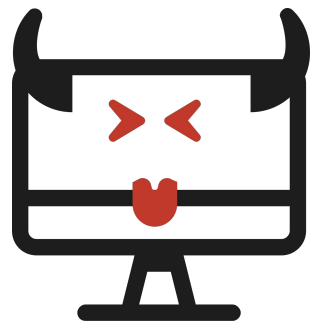


Attacker also configures their server to act as a proxy to business.com

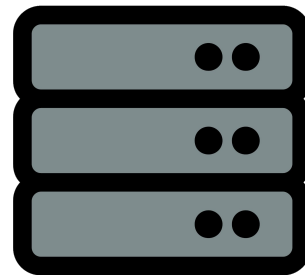


business.com

SSL Hijacking

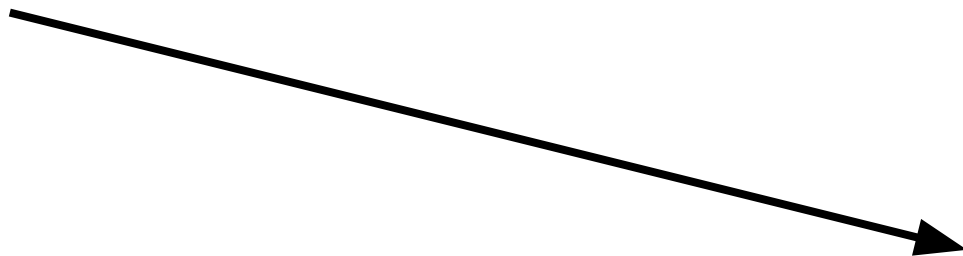
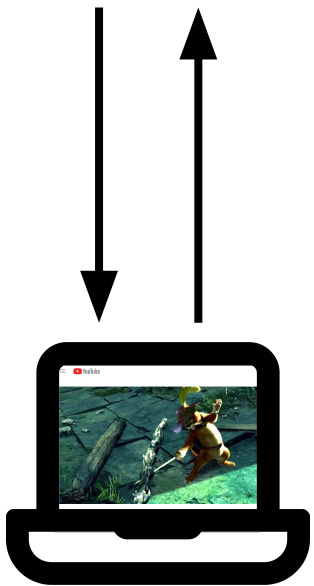
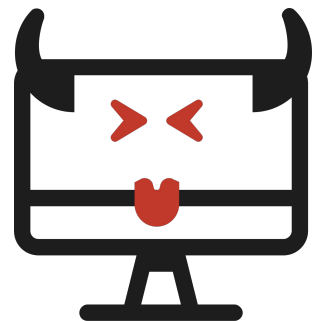


When the user attempts to connect to `business.com`, their DNS cache points to the attacker's server and accepts the fake SSL certificate

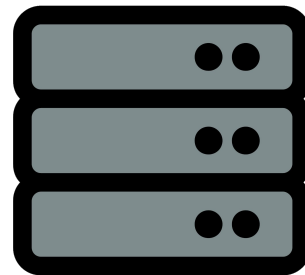


`business.com`

SSL Hijacking

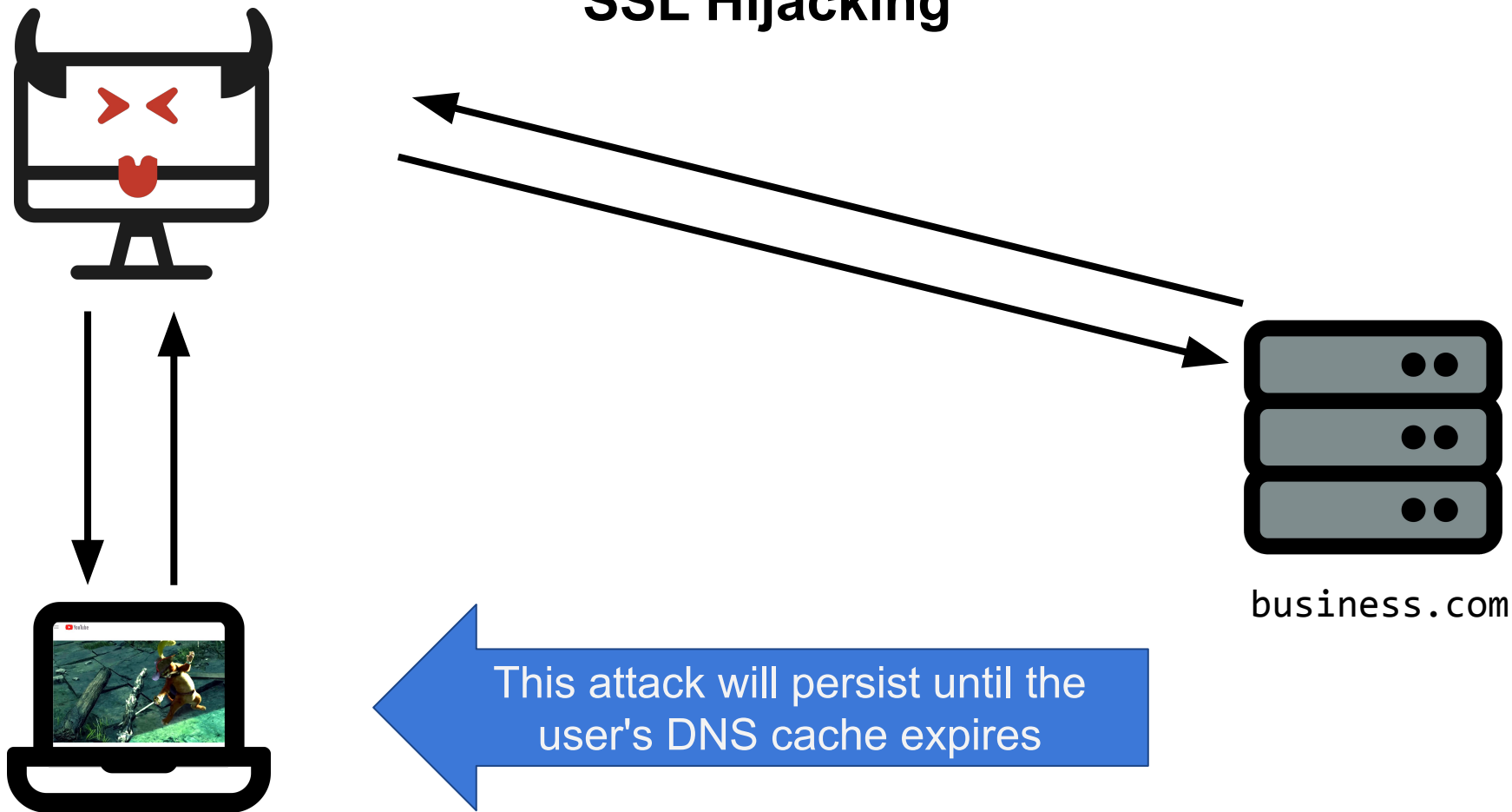


User's request is decrypted by
attacker, logged, and then relayed to
business.com's server

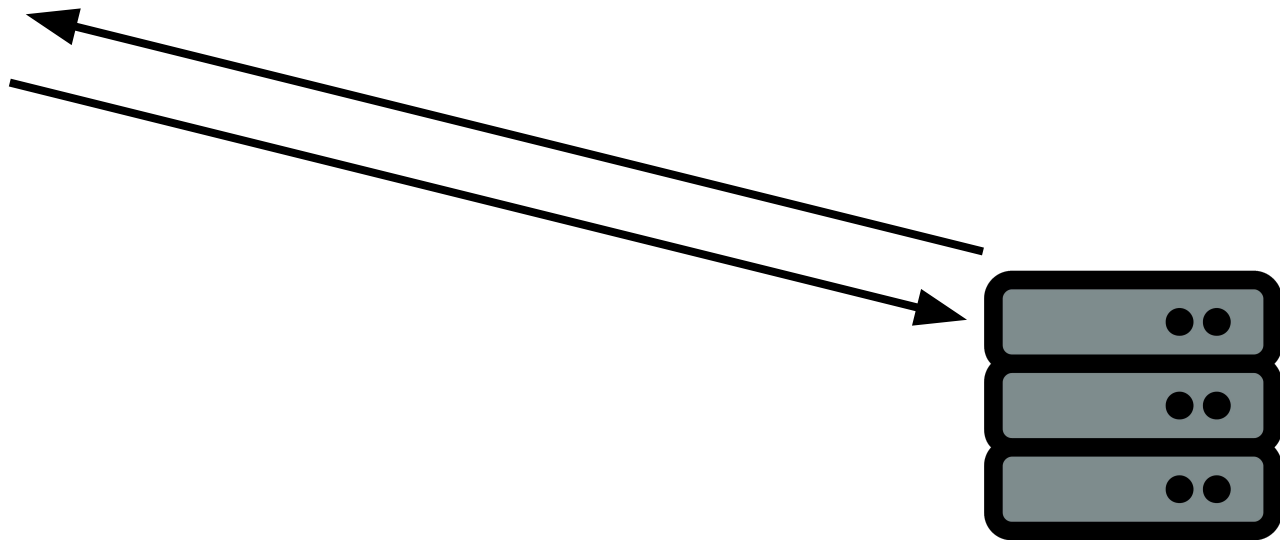
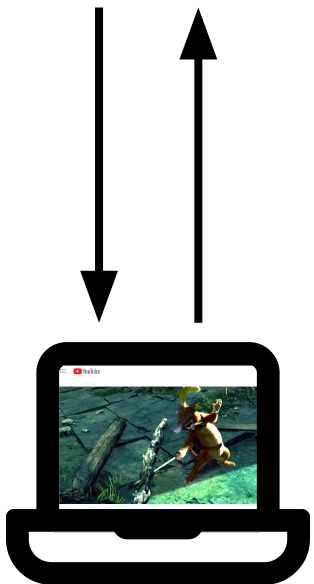
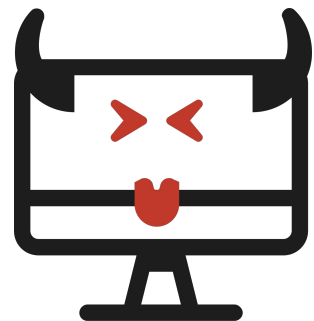


business.com

SSL Hijacking



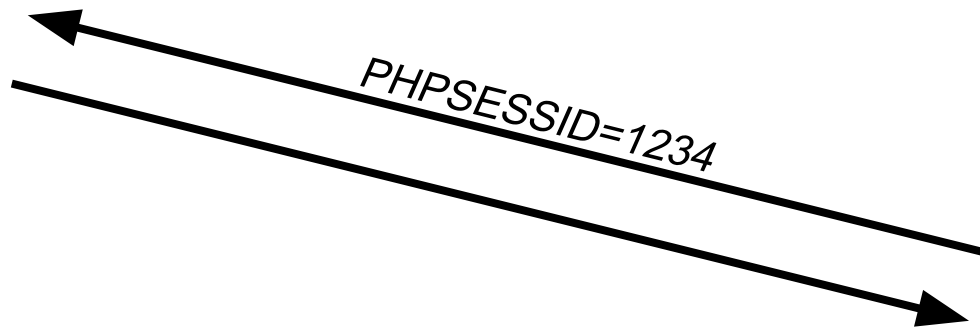
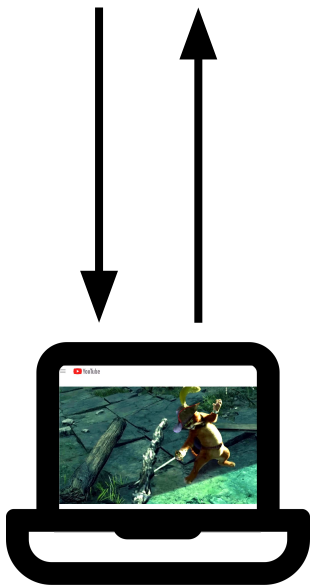
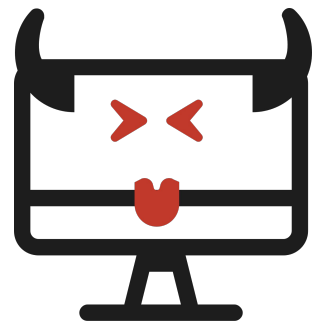
SSL Hijacking



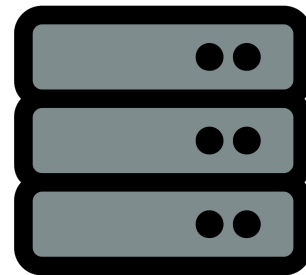
business.com

If the victim installs the fake CA certificate onto the system, detecting SSL hijacking becomes **nearly impossible**

SSL Hijacking



If the server also relies on Session IDs, the attacker can also store those for future attacks



business.com

The Problem: Limitations of Standard TLS Trust

- **Standard TLS/SSL:** Relies on a chain of trust rooted in Certificate Authorities (CAs).
- **Trust Model:** Your device/browser trusts hundreds of CAs globally.
- **The Weak Link:** What if a trusted CA is compromised or tricked into issuing a fraudulent certificate for a legitimate domain?
- **The Threat:** A Man-in-the-Middle (MitM) attacker could present this fraudulent (but technically valid) certificate.
- **Result:** Standard validation would succeed, allowing interception of sensitive data, even over HTTPS.

What is Certificate Pinning?

- **Definition:** A security technique where an application associates a specific host directly with its expected X.509 certificate or public key.
- **Mechanism:** Instead of trusting *any* certificate signed by a trusted CA, the application *only* trusts the specific certificate(s) or public key(s) it has "pinned".
- **Implementation:** Usually done within the client application (e.g., mobile apps, specific software).
- **What's Pinned?:**
 - The hash of the entire certificate.
 - The hash of the certificate's Subject Public Key Info (SPKI) - *often preferred for flexibility.*
 -

How Pinning Works & Benefits

1. **Connection Attempt:** Application connects to a host (e.g., `secure.service.com`).
2. **Server Responds:** Server presents its TLS certificate chain.
3. **Pin Check:** Application extracts the certificate/public key from the server's response.
4. **Comparison:** It compares the extracted info against its stored ("pinned") values for that specific host.
5. **Decision:**
 - **Match:** Connection proceeds securely.
 - **Mismatch:** Connection is **ABORTED**, even if the certificate chain validates against the device's trusted CAs.


Primary Benefit: Drastically reduces the attack surface for MitM attacks using compromised or fraudulent CA certificates.

Drawbacks & Considerations

- **Maintenance Burden:**
 - Pinned certificates expire! The application *must* be updated *before* the server certificate changes.
 - Requires careful coordination between server administration and app development/release cycles.
- **Risk of "Bricking":** If the server certificate changes unexpectedly (e.g., emergency rotation, mistake) *before* the app is updated with the new pin, the app will refuse to connect, locking users out.
- **Inflexibility:** Can break connections when users are behind corporate web proxies that intercept TLS traffic using their own certificates (a legitimate form of MitM in that context).
- **Alternatives/Complements:** Certificate Transparency (CT) logs help detect mis-issued certificates publicly.
- **When to Use:** Best suited for applications with high-security requirements (e.g., banking, finance) where the extra operational complexity is justified. Pinning public keys offers more flexibility than pinning full certificates.

Certificate Search

- Certificates are public and *searchable*
- Based on Certificate Transparency (CT) Logs
- <https://crt.sh/?q=hackpack.club>
- <https://crt.sh/?q=tiktok>

Certificates	crt.sh ID	Logged At 	Not Before	Not After	Common Name	Matching Identities	Issuer Name
	17721946043	2025-04-07	2025-04-07	2025-07-06	ctf2025.hackpack.club	ctf2025.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	17699899850	2025-04-07	2025-04-07	2025-07-06	ctf2025.hackpack.club	ctf2025.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	17704975693	2025-04-06	2025-04-06	2025-07-05	acsac24-hotcrp.hackpack.club	acsac24-hotcrp.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	17678265848	2025-04-06	2025-04-06	2025-07-05	acsac24-hotcrp.hackpack.club	acsac24-hotcrp.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	16889605665	2025-02-27	2025-02-27	2025-05-28	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	16946546654	2025-02-27	2025-02-27	2025-05-28	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	16533113853	2025-02-05	2025-02-05	2025-05-06	acsac24-hotcrp.hackpack.club	acsac24-hotcrp.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	16567379158	2025-02-05	2025-02-05	2025-05-06	acsac24-hotcrp.hackpack.club	acsac24-hotcrp.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	16279192702	2024-12-29	2024-12-29	2025-03-29	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R10
	15922743004	2024-12-29	2024-12-29	2025-03-29	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R10
	15989430383	2024-12-06	2024-12-06	2025-03-06	acsac24-hotcrp.hackpack.club	acsac24-hotcrp.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	15642145351	2024-12-06	2024-12-06	2025-03-06	acsac24-hotcrp.hackpack.club	acsac24-hotcrp.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	15144259955	2024-10-30	2024-10-29	2025-01-27	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	15135140773	2024-10-30	2024-10-29	2025-01-27	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	14368161238	2024-08-31	2024-08-30	2024-11-28	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	14368157635	2024-08-31	2024-08-30	2024-11-28	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13596295233	2024-07-02	2024-07-02	2024-09-30	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13590664890	2024-07-02	2024-07-02	2024-09-30	hackpack.club	hackpack.club www.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13372838489	2024-06-13	2024-06-13	2024-09-11	ask-me-a-question.cha.hackpack.club	ask-me-a-question.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13372839465	2024-06-13	2024-06-13	2024-09-11	ask-me-a-question.cha.hackpack.club	ask-me-a-question.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13372838466	2024-06-13	2024-06-13	2024-09-11	murdermystery.cha.hackpack.club	murdermystery.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13372834535	2024-06-13	2024-06-13	2024-09-11	murdermystery.cha.hackpack.club	murdermystery.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13372839889	2024-06-13	2024-06-13	2024-09-11	yellowdog2.cha.hackpack.club	yellowdog2.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R10
	13372835855	2024-06-13	2024-06-13	2024-09-11	yellowdog2.cha.hackpack.club	yellowdog2.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R10
	13372838587	2024-06-13	2024-06-13	2024-09-11	codesanitize.cha.hackpack.club	codesanitize.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13372839322	2024-06-13	2024-06-13	2024-09-11	codesanitize.cha.hackpack.club	codesanitize.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R11
	13372838332	2024-06-13	2024-06-13	2024-09-11	llmrunner.cha.hackpack.club	llmrunner.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R10
	13372838146	2024-06-13	2024-06-13	2024-09-11	longhorn2.cha.hackpack.club	longhorn2.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R10
	13372835636	2024-06-13	2024-06-13	2024-09-11	longhorn2.cha.hackpack.club	longhorn2.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R10
	13372838570	2024-06-13	2024-06-13	2024-09-11	llpm.cha.hackpack.club	llpm.cha.hackpack.club	C=US, O=Let's Encrypt, CN=R11

The State of https Adoption on the Web

Workshop on Measurements, Attacks, and Defenses for the Web
(MADWeb) 2025 [link](#)

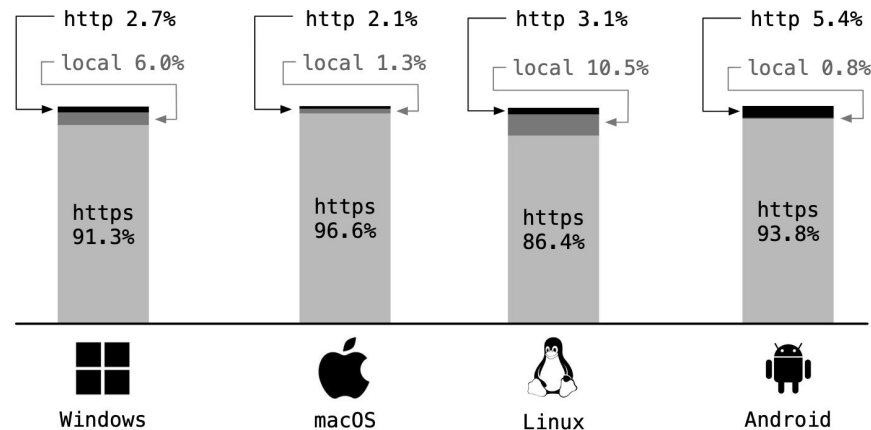


Fig. 3: Adoption of https on the different Operating Systems: Windows, macOS, Linux and Android.

Let's Encrypt

- Let's Encrypt issued its first certificate in 2015
- It **democratized web security**
- Let's Encrypt is funded entirely through charitable contributions, primarily from corporate sponsorships and individual donations
- Grew from serving a few thousand domains to nearly 600 million between 2015-2025
- Currently issues more than 6 million TLS certificates daily
- Serves more than 550 million websites worldwide
- Has become the world's largest certificate authority, providing more HTTPS certificates than all other certificate authorities combined
- It's operating budget is ~\$3m/year (!)

