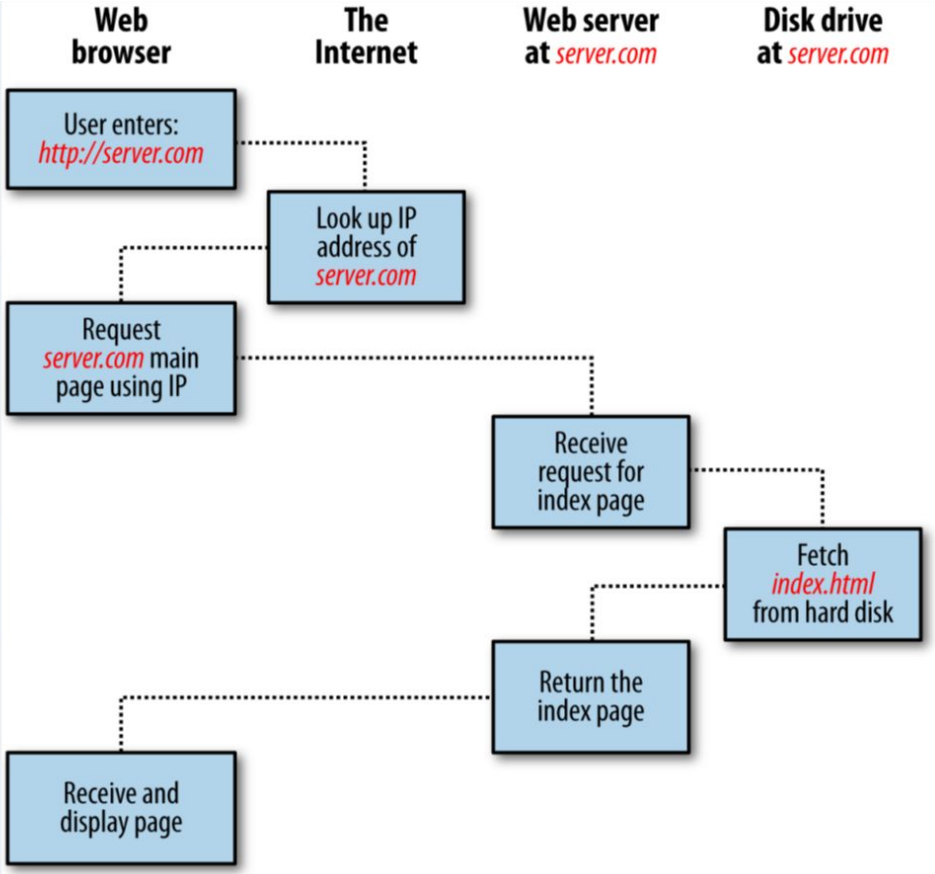
CSC 405
SSL & HTTPS

Adam Gaweda
agaweda@ncsu.edu

Alexandros Kapravelos
akaprav@ncsu.edu

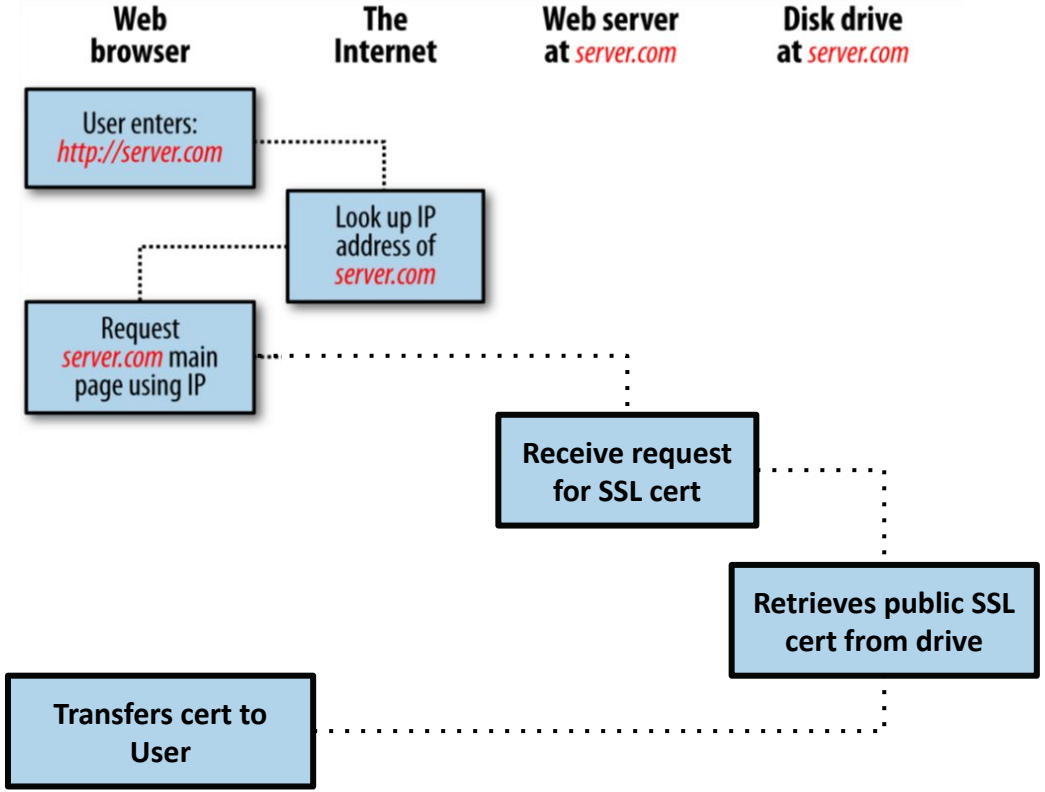
HTTP Workflow

1	User enters web address in address bar
2	Browser looks up the IP address of the URL via DNS
3	Browser submits a request for content from IP
4	Server receives the request
5	Server looks for the content on its hard disk
6	Binary Digits are Transferred via the Internet
7	Browser processes the binary into HTML, which it then interprets



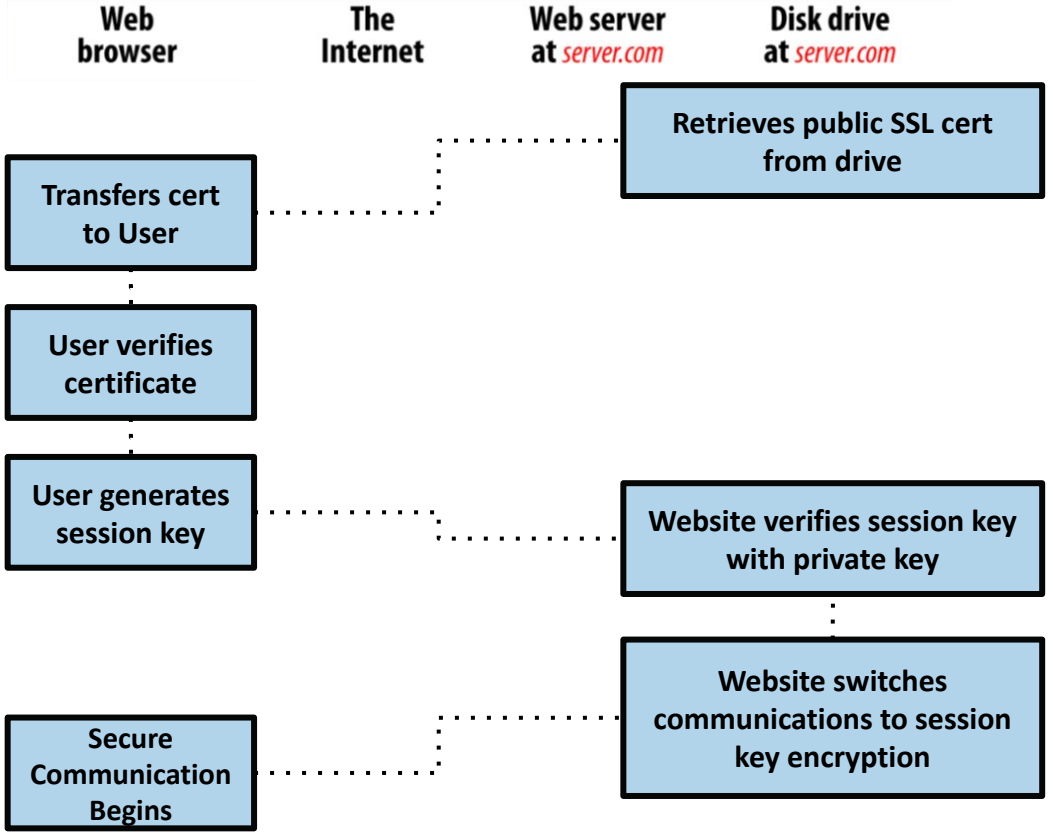
HTTPS Workflow

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



Creating the Certificate

Step One: Generate a Certificate Signing Request (CSR)

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

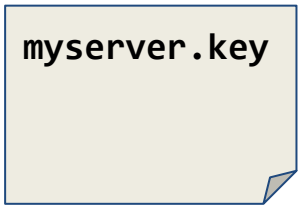
```
You are about to be asked to enter information that will be incorporated into your
certificate request.
...
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:NC
Locality Name (eg, city) []:Raleigh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NC 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 []:
```

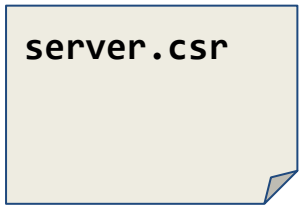
Creating the Certificate

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



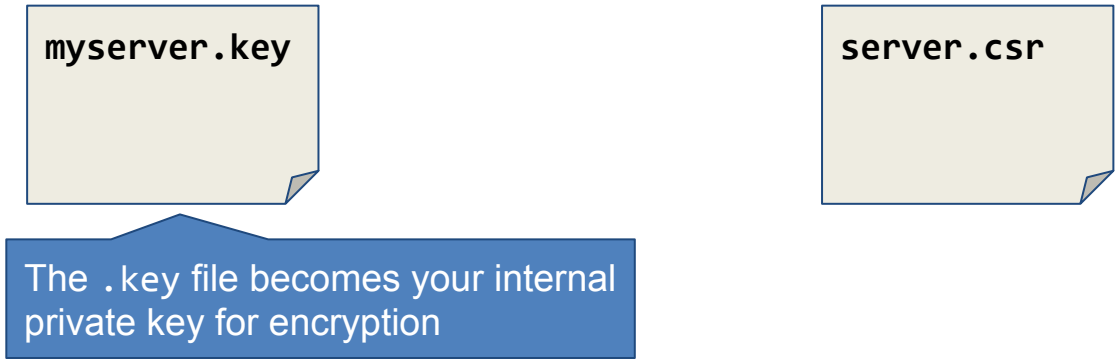


Creating the CSR

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

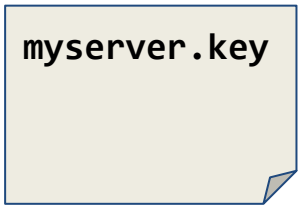


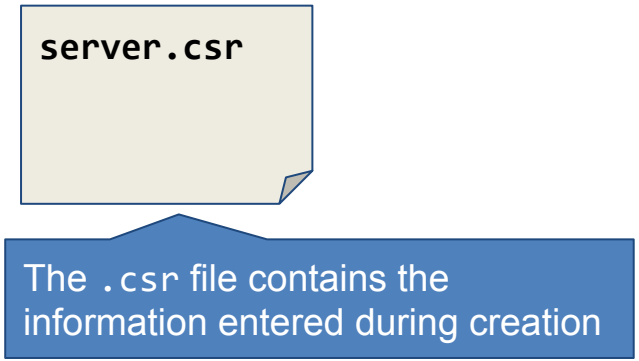
Creating the CSR

Step One: Generate a Certificate Signing Request (CSR)

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

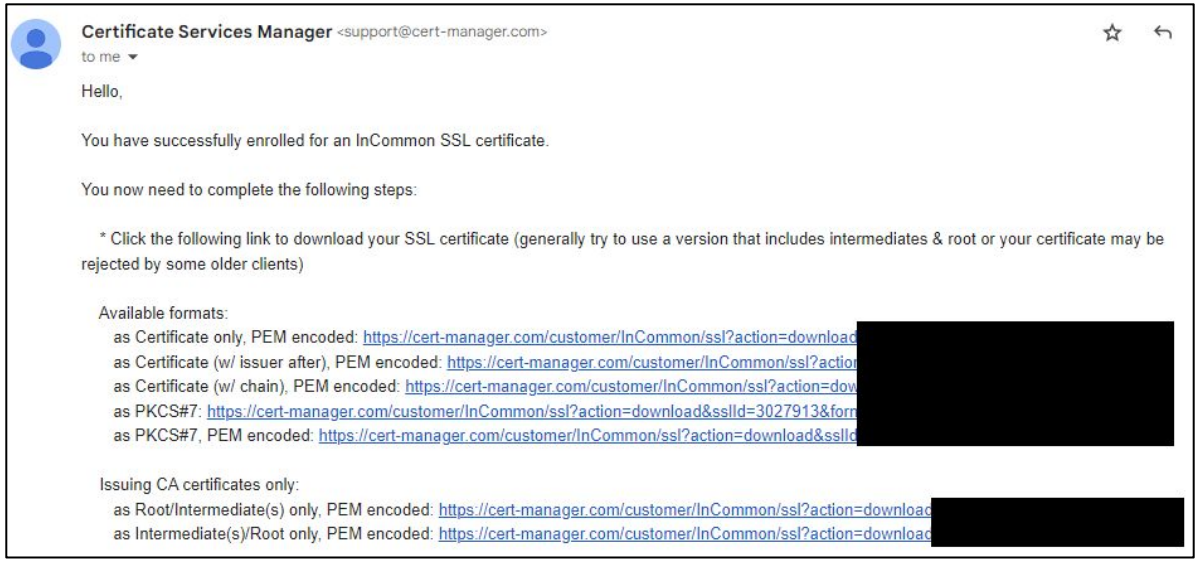
This will generate two files with RSA-2048 encryption





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



Certificate Services Manager <support@cert-manager.com>
to me

Hello,

You have successfully enrolled for an InCommon SSL certificate.

You now need to complete the following steps:

* Click the following link to download your SSL certificate (generally try to use a version that includes intermediates & root or your certificate may be rejected by some older clients)

Available formats:
as Certificate only, PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=download
as Certificate (w/ issuer after), PEM encoded: https://cert-manager.com/customer/InCommon/ssl?actio
as Certificate (w/ chain), PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=dow
as PKCS#7: https://cert-manager.com/customer/InCommon/ssl?action=download&sslId=3027913&forn
as PKCS#7, PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=download&sslId

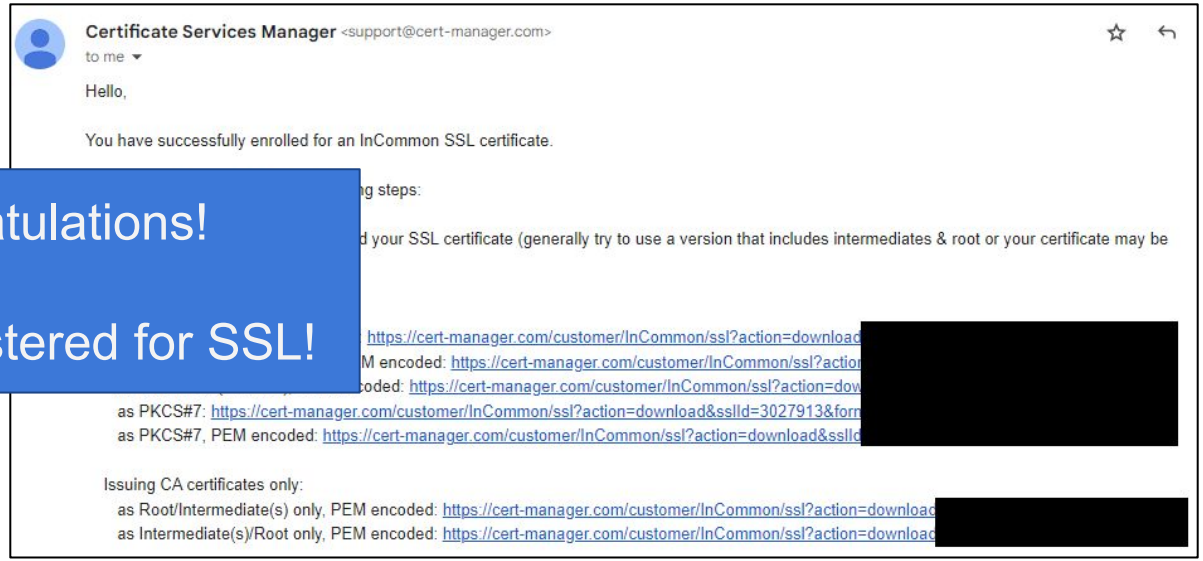
Issuing CA certificates only:
as Root/Intermediate(s) only, PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=downloac
as Intermediate(s)/Root only, PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=downloac

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

Congratulations!

You've registered for SSL!



Certificate Services Manager <support@cert-manager.com>
to me

Hello,

You have successfully enrolled for an InCommon SSL certificate.

g steps:

d your SSL certificate (generally try to use a version that includes intermediates & root or your certificate may be

https://cert-manager.com/customer/InCommon/ssl?action=download
M encoded: https://cert-manager.com/customer/InCommon/ssl?actio
coded: https://cert-manager.com/customer/InCommon/ssl?action=dow
as PKCS#7: https://cert-manager.com/customer/InCommon/ssl?action=download&sslId=3027913&for
as PKCS#7, PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=download&sslI

Issuing CA certificates only:
as Root/Intermediate(s) only, PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=downloa
as Intermediate(s)/Root only, PEM encoded: https://cert-manager.com/customer/InCommon/ssl?action=downloa

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
  SSLCertificateFile /etc/ssl/cert/domainName.crt
  SSLCertificateKeyFile /etc/ssl/key/myserver.key
  ...
  ProxyPass / uwsgi://localhost:65432/
  ProxyPassReverse / uwsgi://localhost:65432/
</VirtualHost>
```

Congratulations! You're website is HTTPS!

Configure Your Server

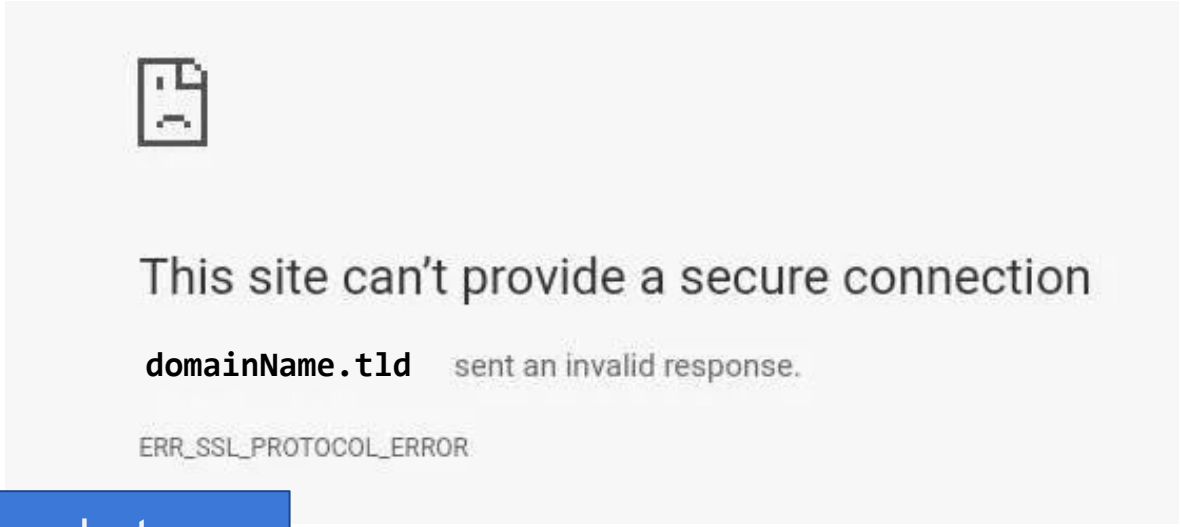
- This will range from the application running your server (Apache, nginx, etc.)

```
[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.)

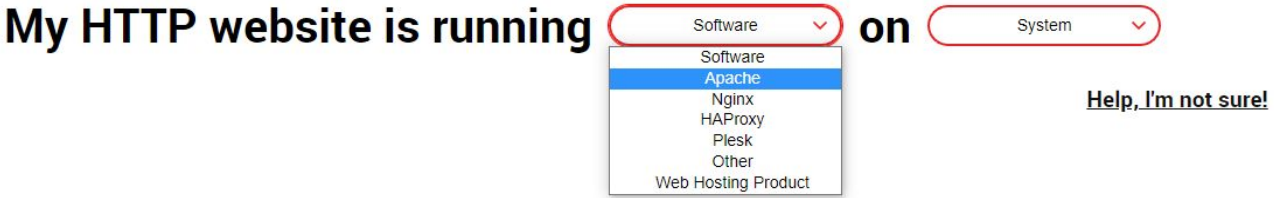


…and debug whatever you broke 😆

Let's Encrypt

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

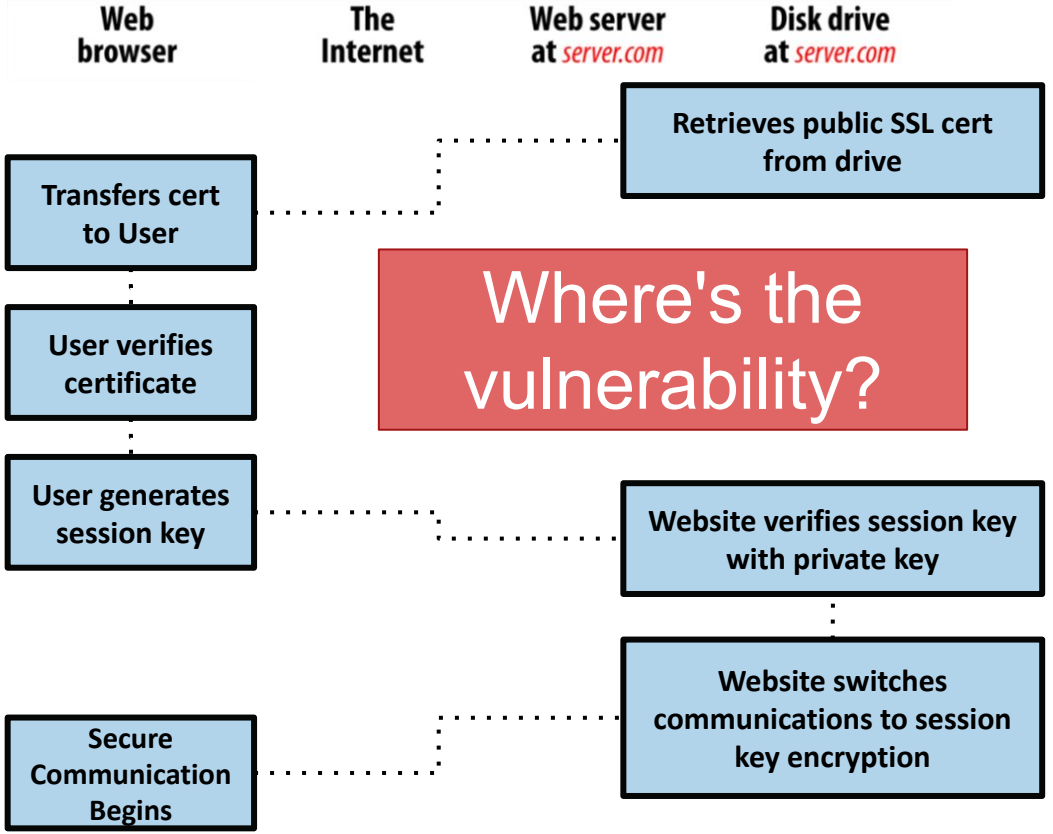
https://letsencrypt.org/



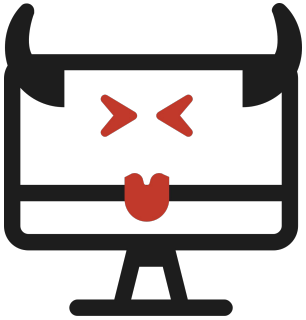

```
[user@server ~] snap install core
[user@server ~] snap refresh core
[user@server ~] snap install --classic certbot
[user@server ~] ln -s /snap/bin/certbot /usr/bin/certbot
[user@server ~] certbot --apache
```

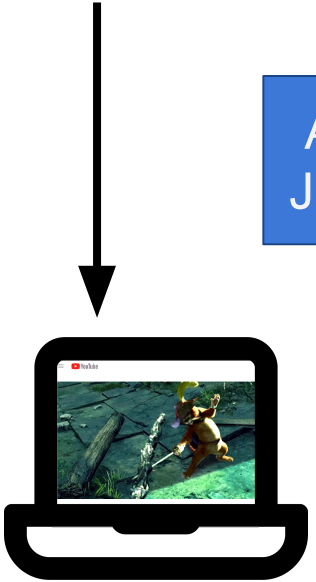
HTTPS Workflow

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

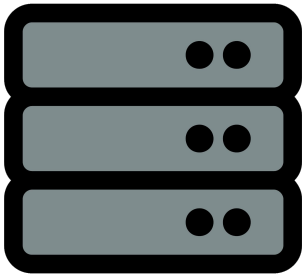


SSL Hijacking



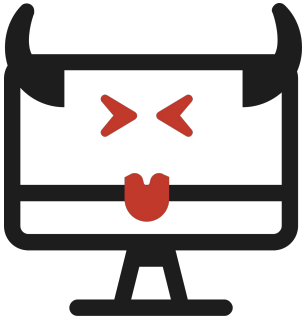


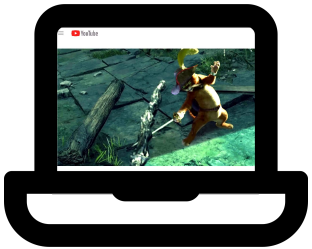
Attacker sends a phishing attack utilizing JavaScript to install a bogus CA certificate



business.com

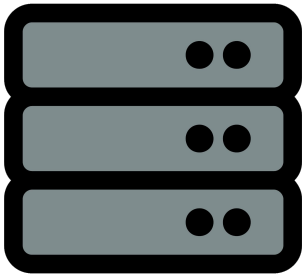
SSL Hijacking





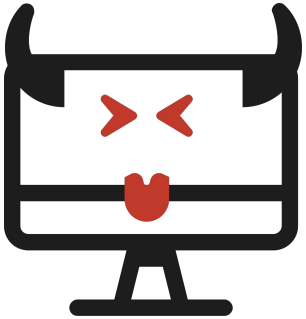
Known as
"DNS Poisoning"

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

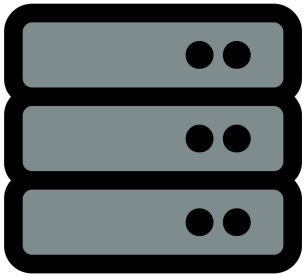


`business.com`

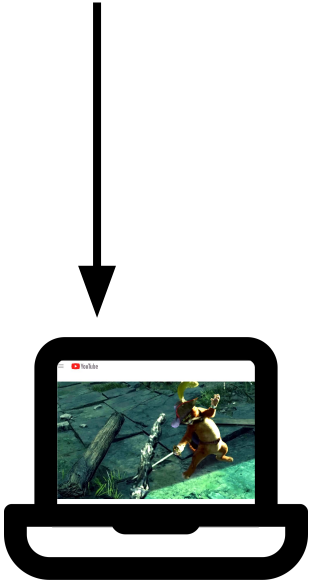
SSL Hijacking



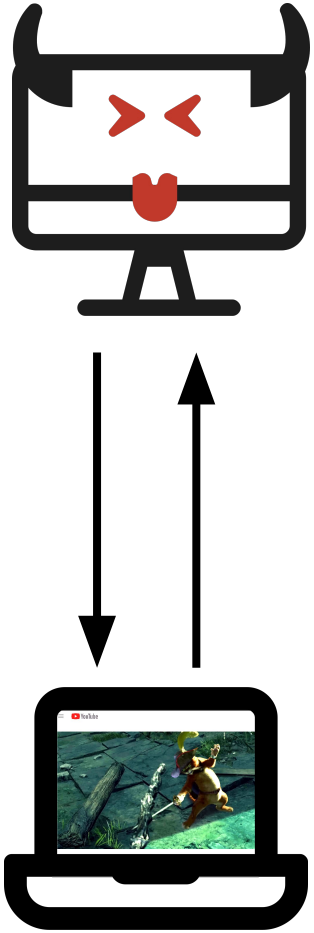
Attacker also configures their server to act as a proxy to `business.com`



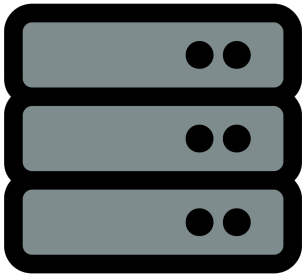
`business.com`



SSL Hijacking

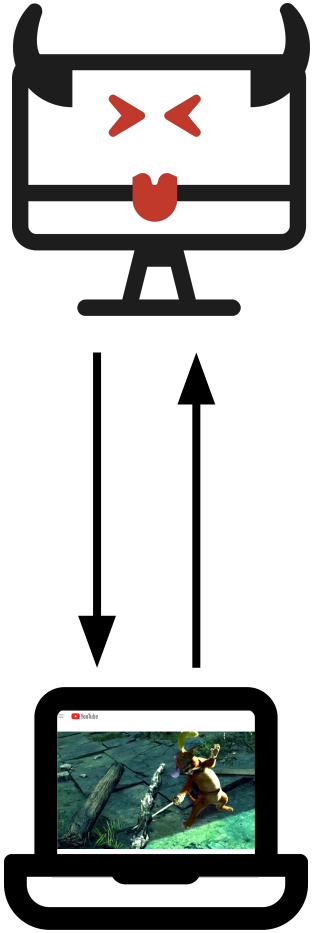


When the user attempts to connect to `business.com`, their DNS cache points to the attacker's server and accepts the fake SSL certificate

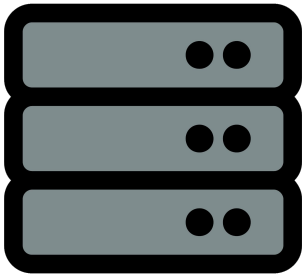


`business.com`

SSL Hijacking

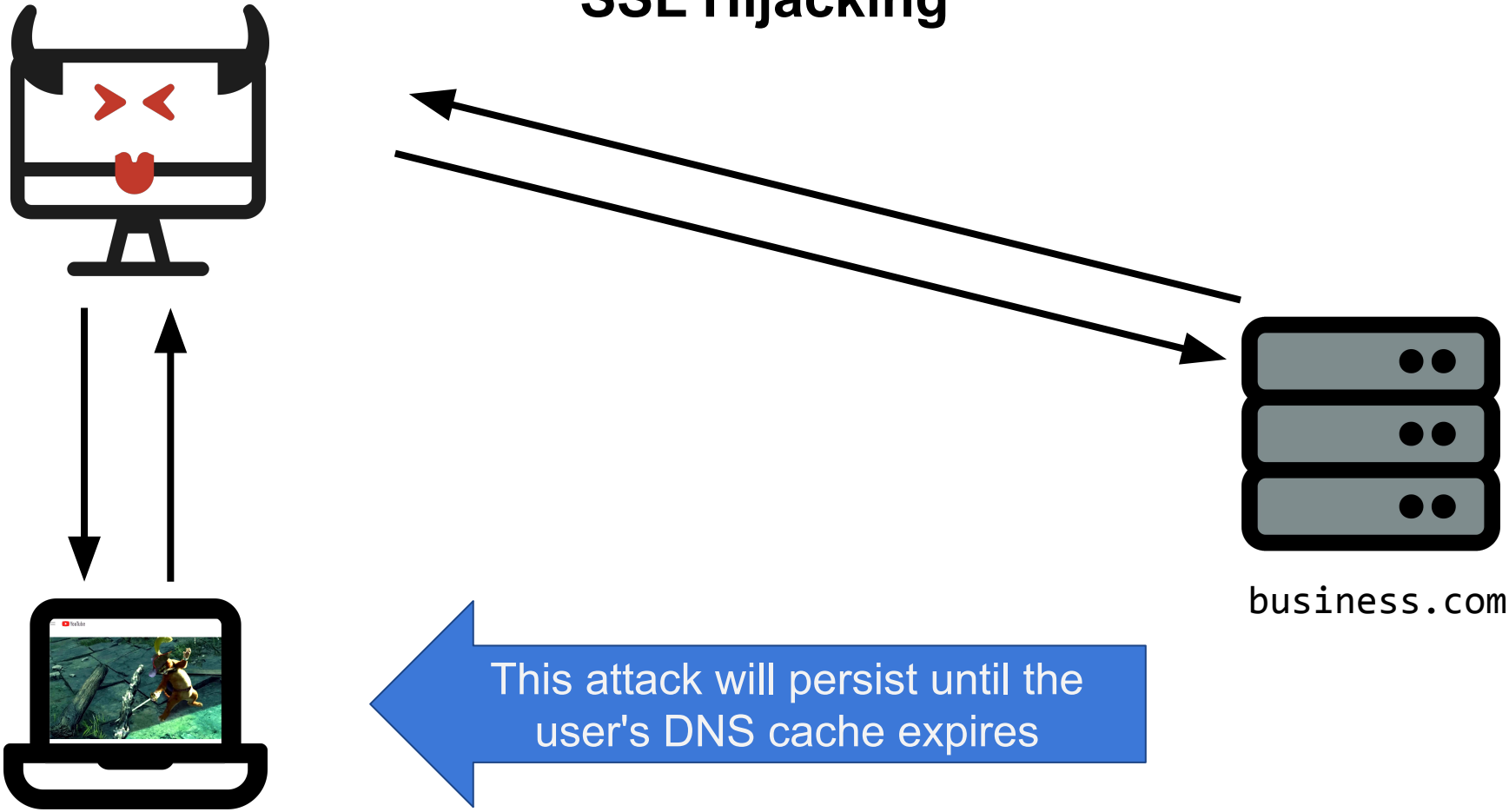


User's request is decrypted by attacker, logged, and then relayed to `business.com`'s server

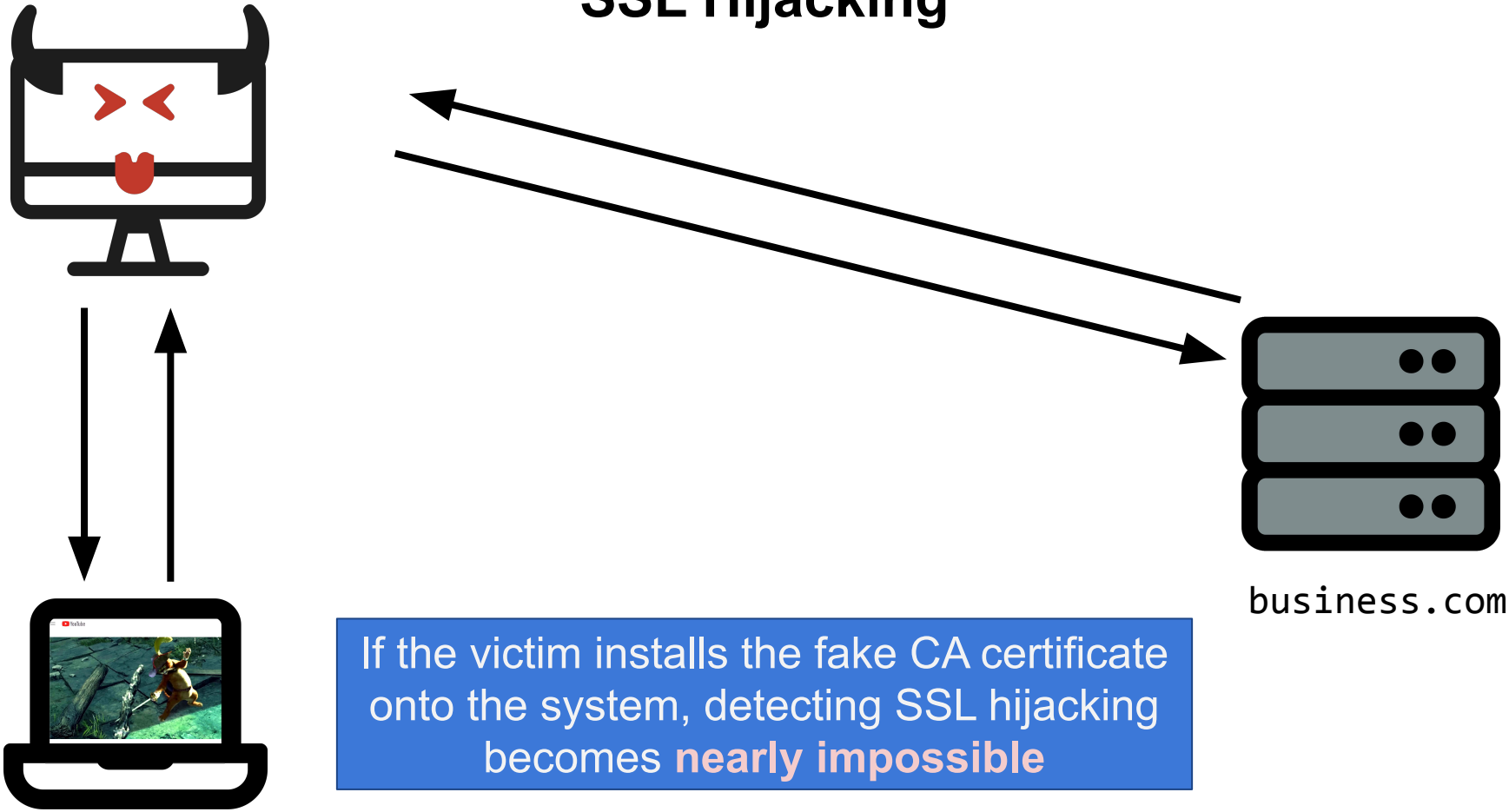


`business.com`

SSL Hijacking

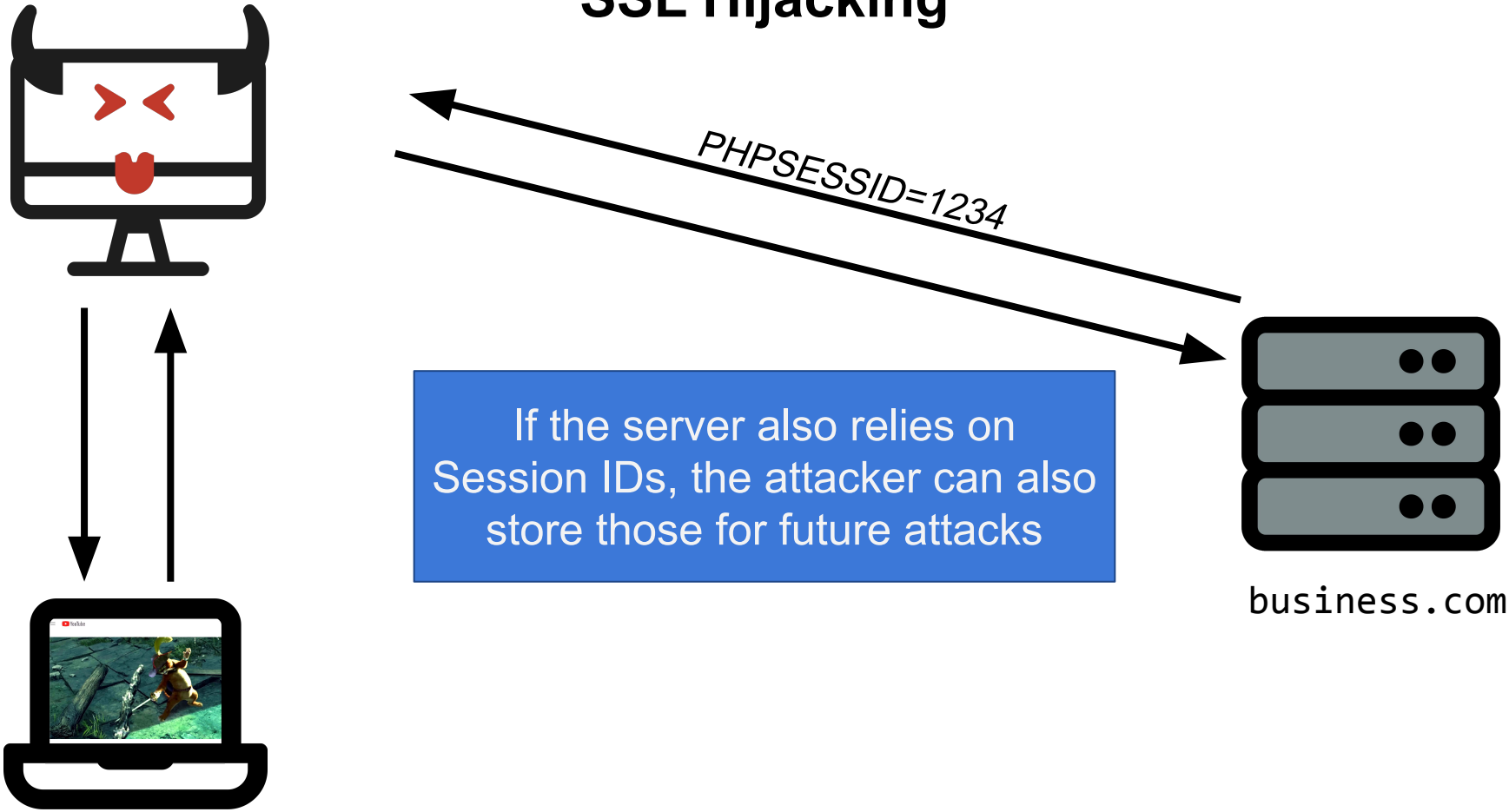


SSL Hijacking



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

SSL Hijacking



Attacking LLPM

llpm.cha.hackpack.club