COMP3052.SEC Computer Security

Session 09: Internet Security

Acknowledgements

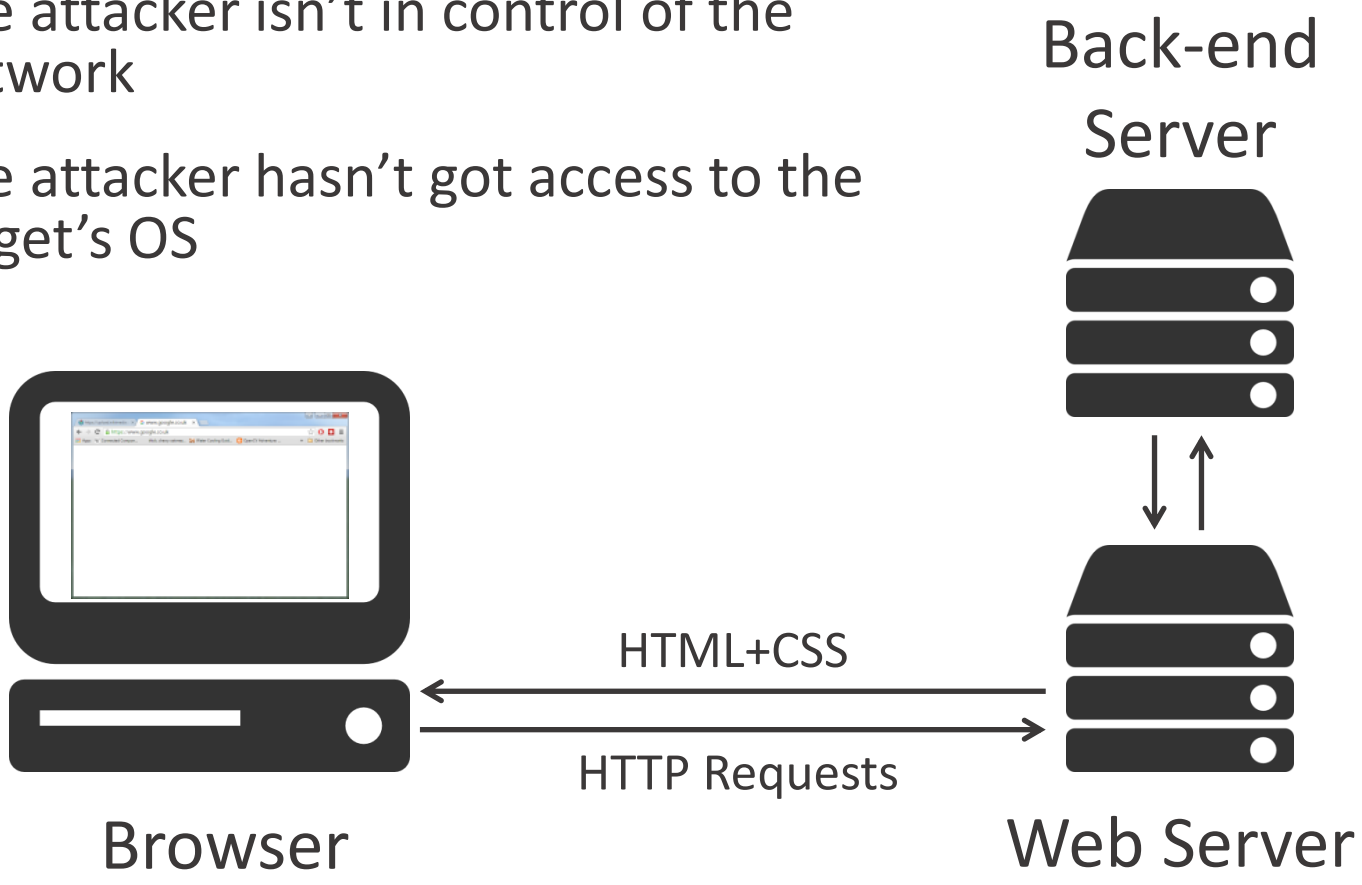
- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...
- Thank you to (amongst others):
  - Michel Valstar, Milena Radenkovic, Mike Pound, Dave Towey, …

This Session

- Internet Security
- Cookies – Stealing & Tracking
- Cross-site Scripting
- Cross-site Request Forgery
- SSL / TLS
  - Vulnerabilities

Browser Server Model

- Different from other threat models:
  - The attacker isn't in control of the network
  - The attacker hasn't got access to the target's OS



Threat Model

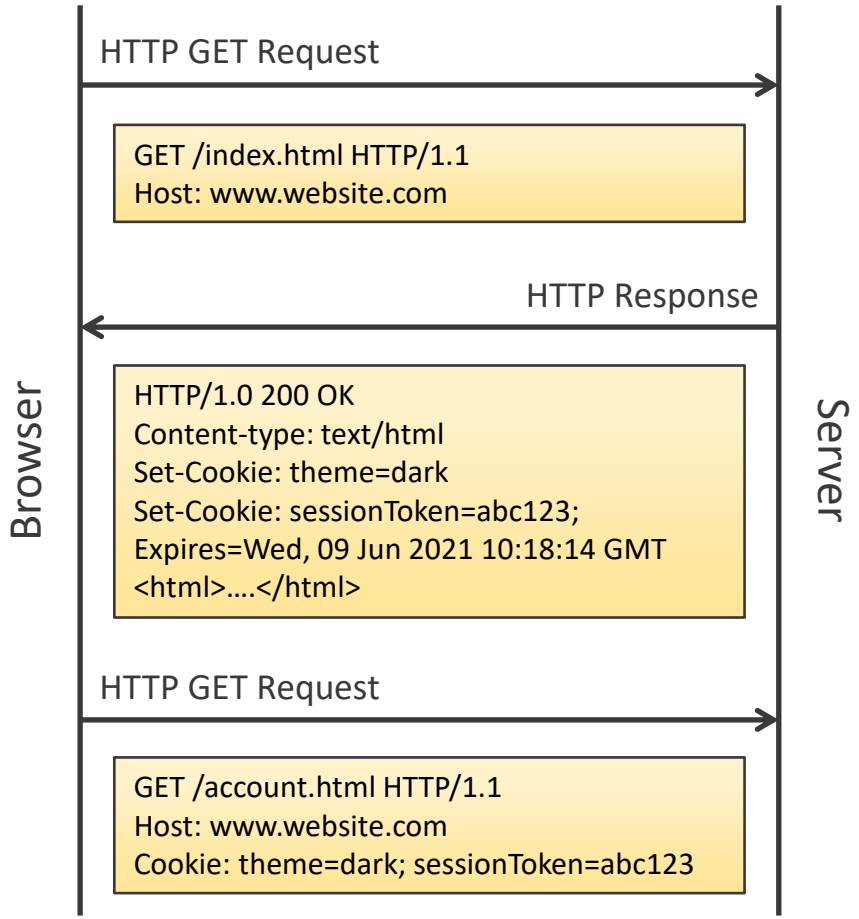
- Different from other threat models:
  - The attacker isn't in control of the network
  - The attacker hasn't got access to the target's OS
- Instead, the attacker
  - Sees messages addressed to themselves or others
  - Sees data obtained from security compromises
  - Can make educated guesses

Cookies

- HTTP is a stateless protocol
- Most of what we do online is stateful
- Cookies are small text files used to provide persistence

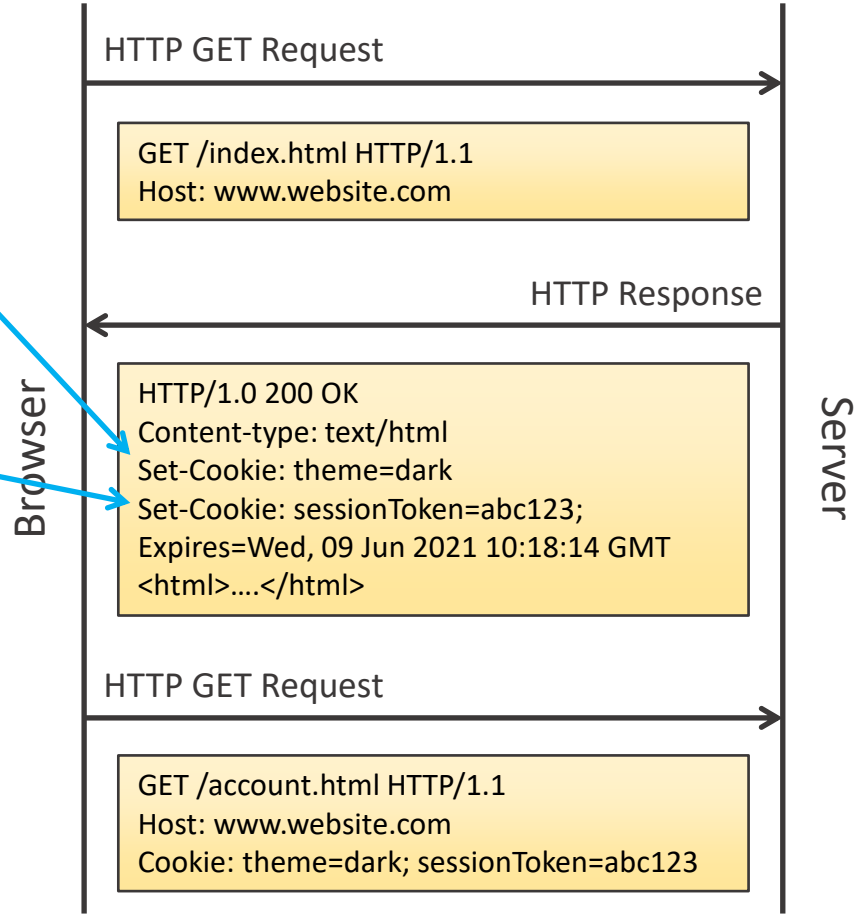
Cookie

- Servers can provide cookies during HTTP responses, using Set-Cookie
- Browsers will return any cookies for a given domain in GET and POST requests



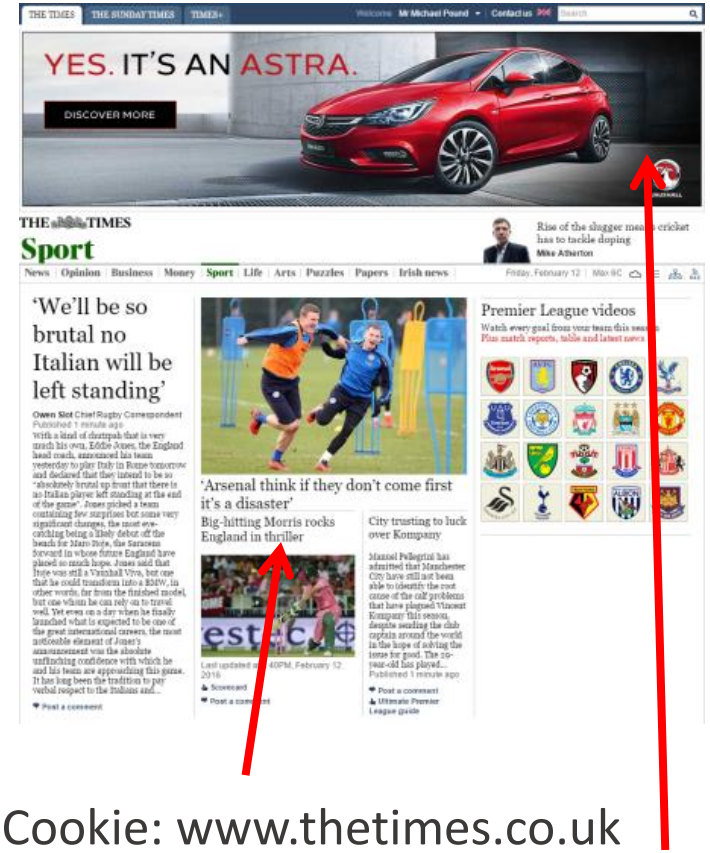
Types of Cookie

- Session – Deleted when the browser exits, contain no expiration date
- Persistent – Expire at a given time
- Secure – Can only be used over HTTPS
- HTTPOnly – Inaccessible from JS



Third Party Cookies

- Cookies are associated with the domains that produced them
  - Amazon.com cookies don't go to google.co.uk
- Some websites include requests to other domains, such as 3rd party advertisers
  - These serve cookies – *a lot*



Cookie: www.thetimes.co.uk

Cookie: happybanners.net

Tracking Users

- Third party cookies are returned to the advertiser every time any website includes one of their ads
- Over time, very detailed information on users can be constructed

The Complete Hacking Course: Learn ethical hacking with Kali Linux!

Cookie Vulnerabilities

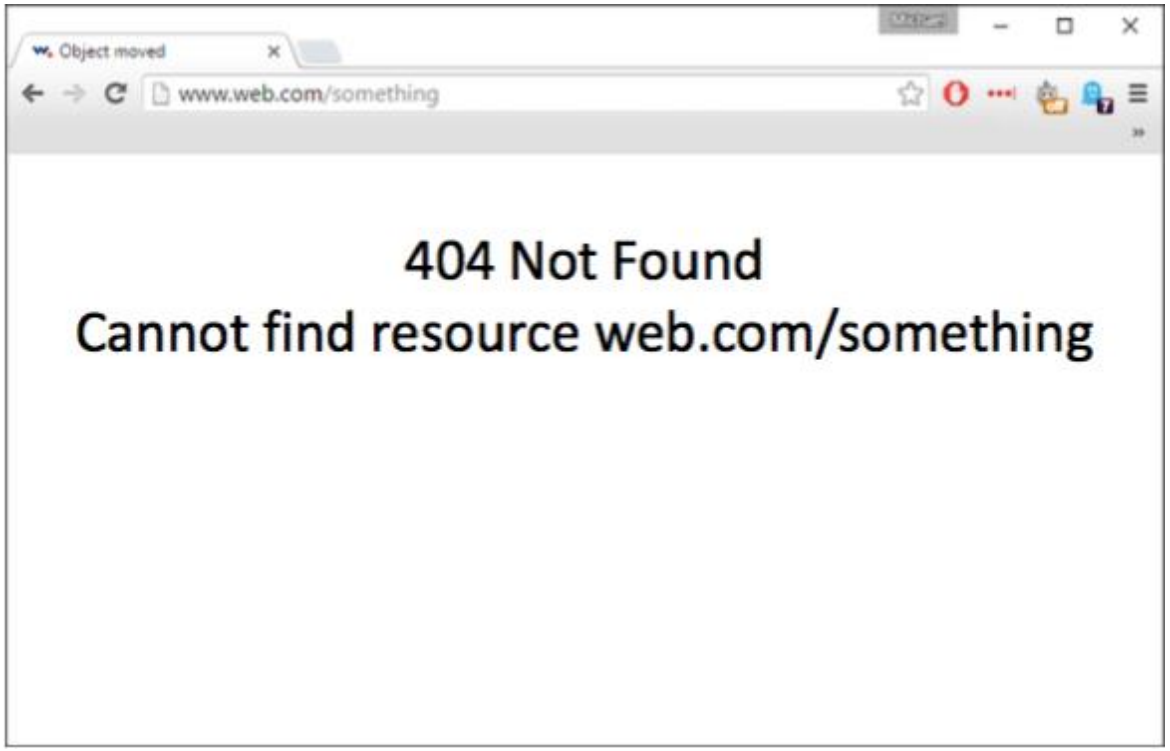
- How a website uses a cookie is up to the server
- Many create an SID to authenticate users, for example to “keep me logged on”
- Obtaining this cookie – Cookie Stealing – lets you hijack their session
  - HTTP Cookies can be stolen simply by monitoring
  - HTTPS will require Cross-site scripting attacks or DNS poisoning

Cross-site Scripting (XSS)

- A type of injection attack, similar in many ways to an SQL Injection
- HTML is read by a browser, and is a combination of content (text) and structure (html tags)
- If we can inject html structures into the content of a website, the browser will simply execute these – e.g. <script> tags

Cross-site Scripting (XSS)

- A malicious URL that inserts an exploit directly into the page returned by a server
- Consider a 404 page at www.web.com/something:



Cross-site Scripting (XSS)

- Now consider this URL:

Preventing XSS

- Websites must aggressively escape HTML characters from *any* user input / output

- When you consider all of the things people input on interactive websites, this can be a real problem

Cross-site Request Forgery (XSRF)

- When a user puts in an HTTP request, they will also send any relevant session cookies
  - E.g. an SID from having logged in
- If the user has already authenticated, a malicious URL can then perform some action on their account

```
http://shop.com/account.php?act=editemail&e=attacker@mail.com
```

XSRF in POST

- Most websites use POST, this is little defence
- The phishing email just points to a convincing website with a malicious form on it

```
<form action="http://bank.com/transfer.do" method="POST">
<input type="hidden" name="acct" value="attacker"/>
<input type="hidden" name="amount" value="10000"/>
<input type="submit" value="View my pics!"/>
</form>
```

```
<body onload="document.forms[0].submit()">
<form...
```

You don't even need to have them click..

Preventing XSRF

- XSS vulnerabilities make XSRF a lot easier! Fix these!
- Use synchronizer tokens
  - Each website form has a one-time token that the server validates when the form is submitted

```
<form action="http://bank.com/transfer.do" method="POST">
<input type="hidden" name="sToken" value="OWY4NgmQdnw"
<input type="hidden" name="acct" value="attacker"/>
<input type="submit" value="Transfer Money"/>
</form>
```

SSL/TLS

- There are dangers associated with sending plain text cookies, passwords etc.
- SSL, and the newer TLS provide authenticated and encrypted sessions
- Secure Socket Layer (SSL) came first, then after v3.0 it became Transport Layer Security (TLS), currently v1.3

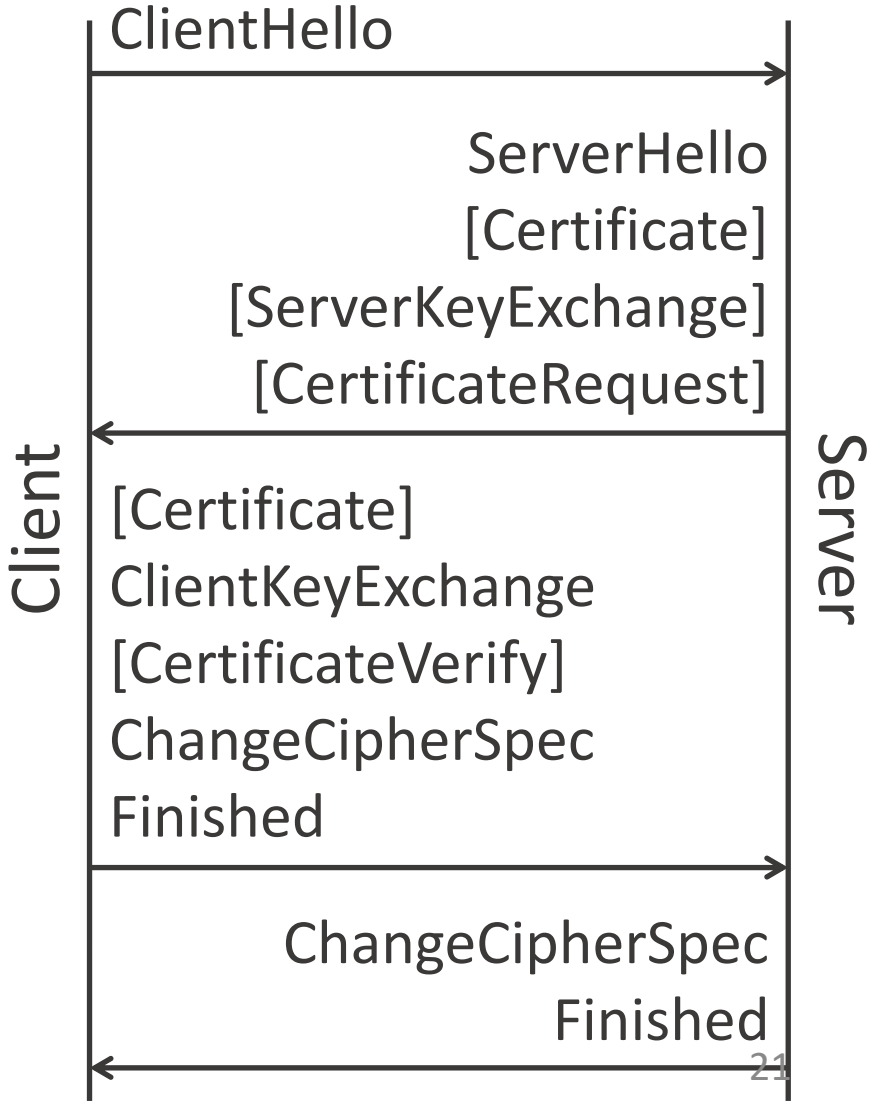
We will treat SSL and TLS here interchangeably

TLS

- Transport Layer Security has two layers:
- The Record Layer
  - Using established symmetric keys and other session info, will encrypt application packets, very like IPSec
- The Handshake Layer
  - Used to establish session keys, as well as authenticate either party – usually the server using a Public-Key Certificate

TLS Handshake

- The TLS handshake allows us to:
  - Establish the master secret
  - Resume sessions
  - Authenticate the identity of the server or client
- This example is for ECDHE_RSA



TLS Handshake

ClientHello

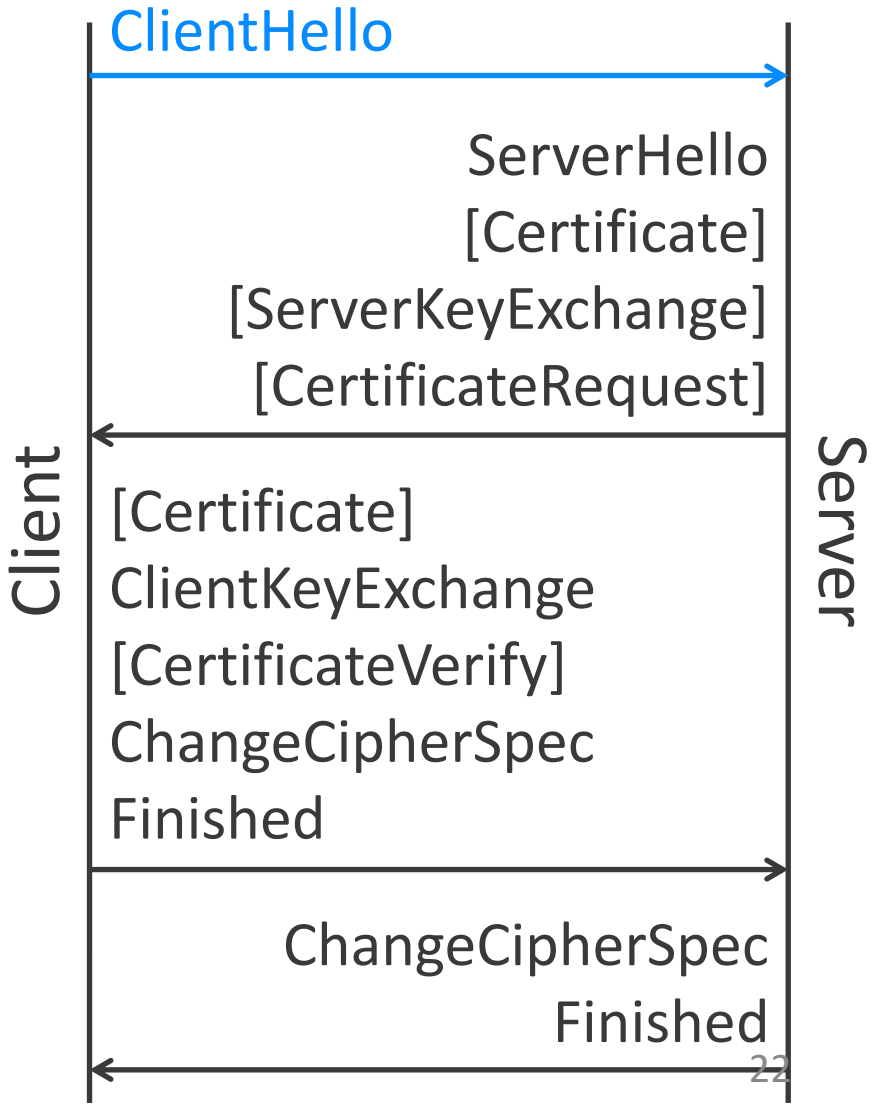
Supported TLS version: 1.2

Random number: f3bc12ad..

Supported Ciphers
{
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
}

[Extensions]

[Session ID]



TLS Handshake

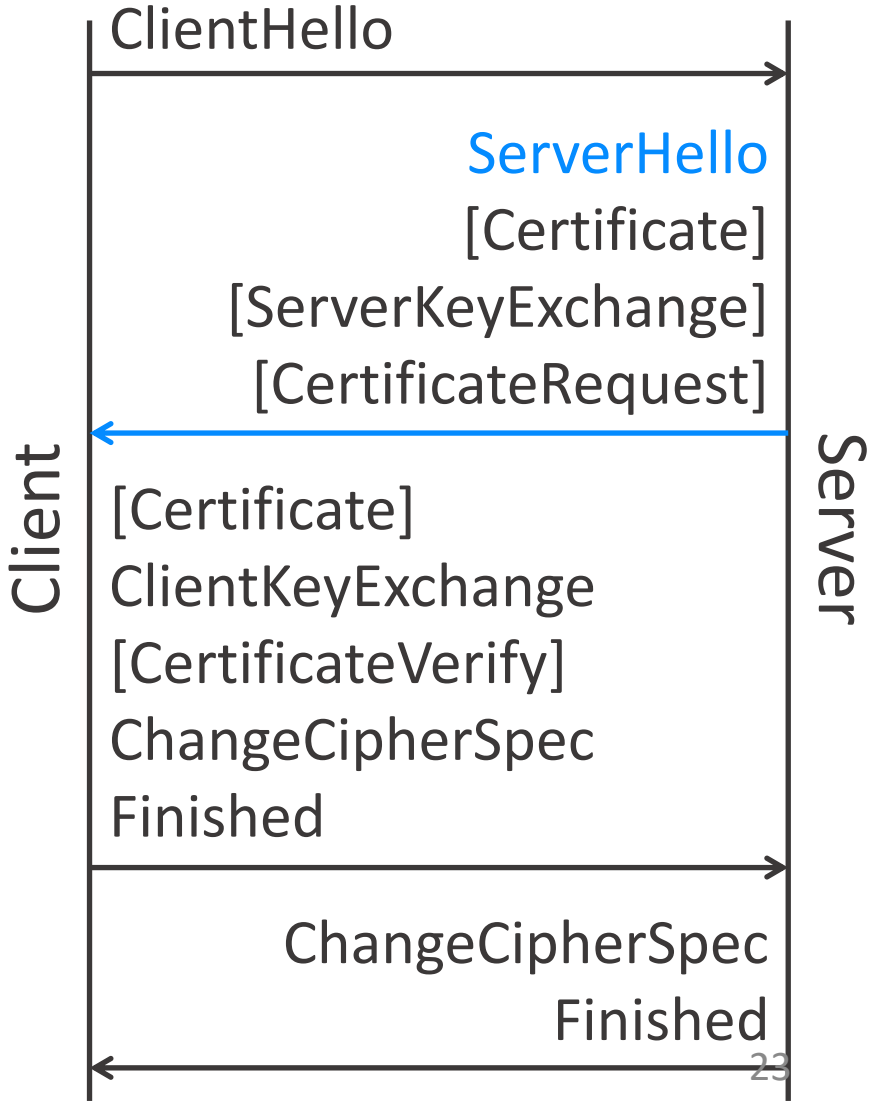
ServerHello

Version: 1.2

Random number: 16cf90ed..

Suite :
TLS_ECDHE_RSA_WITH_
AES_128_GCM_SHA256

[Session ID]



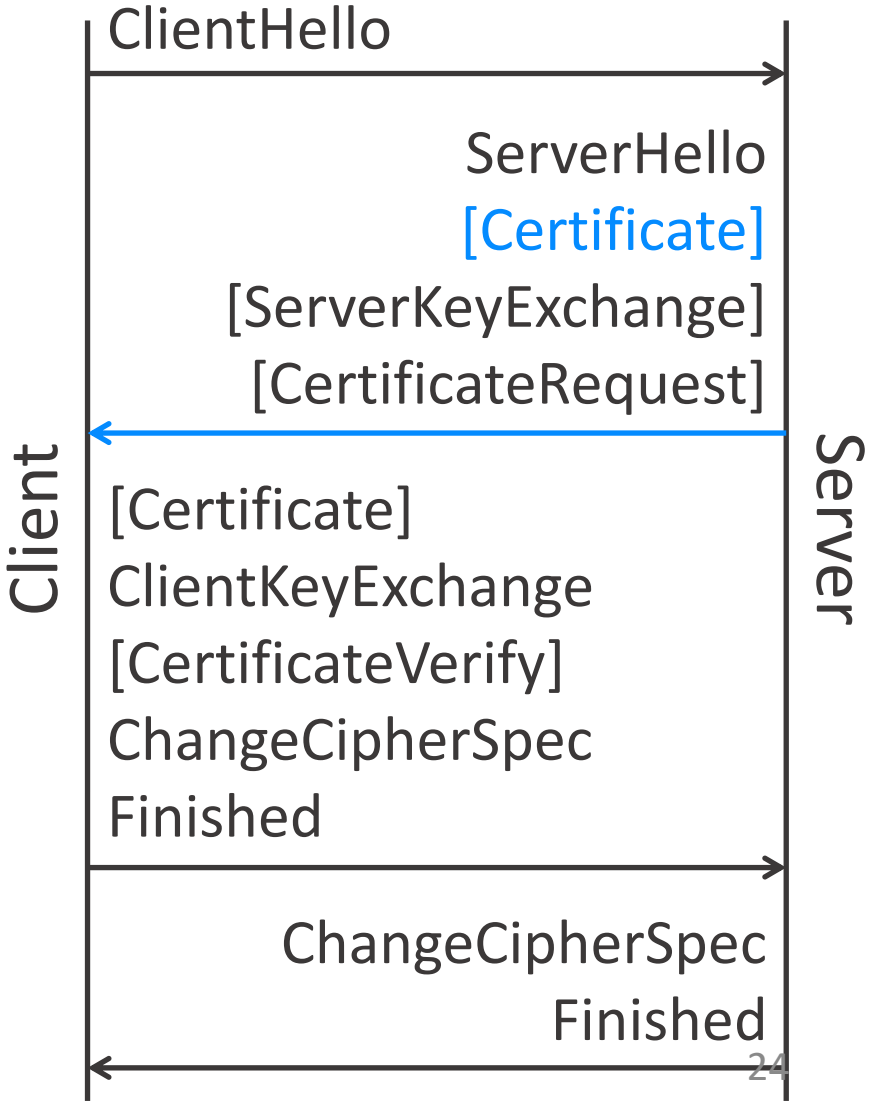
TLS Handshake

Certificate

The server sends its public-key certificate to the client

The client checks it using its browser root certificates, or via a CA cert

Digital signature using server private key, client uses this to confirm server identity



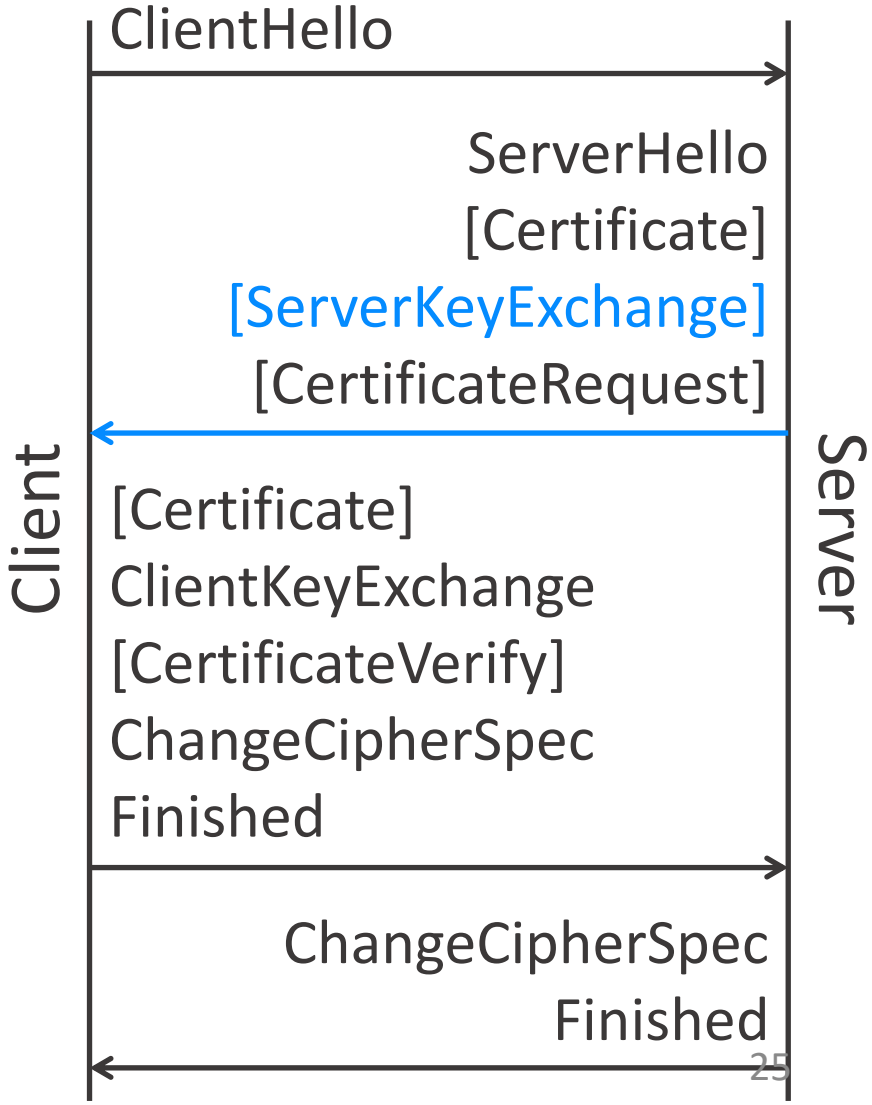
TLS Handshake

ServerKeyExchange

Elliptic Curve Diffie-Hellman Parameters:

Named Curve: secp256r1 (0x0017)

DH Public Key: bG



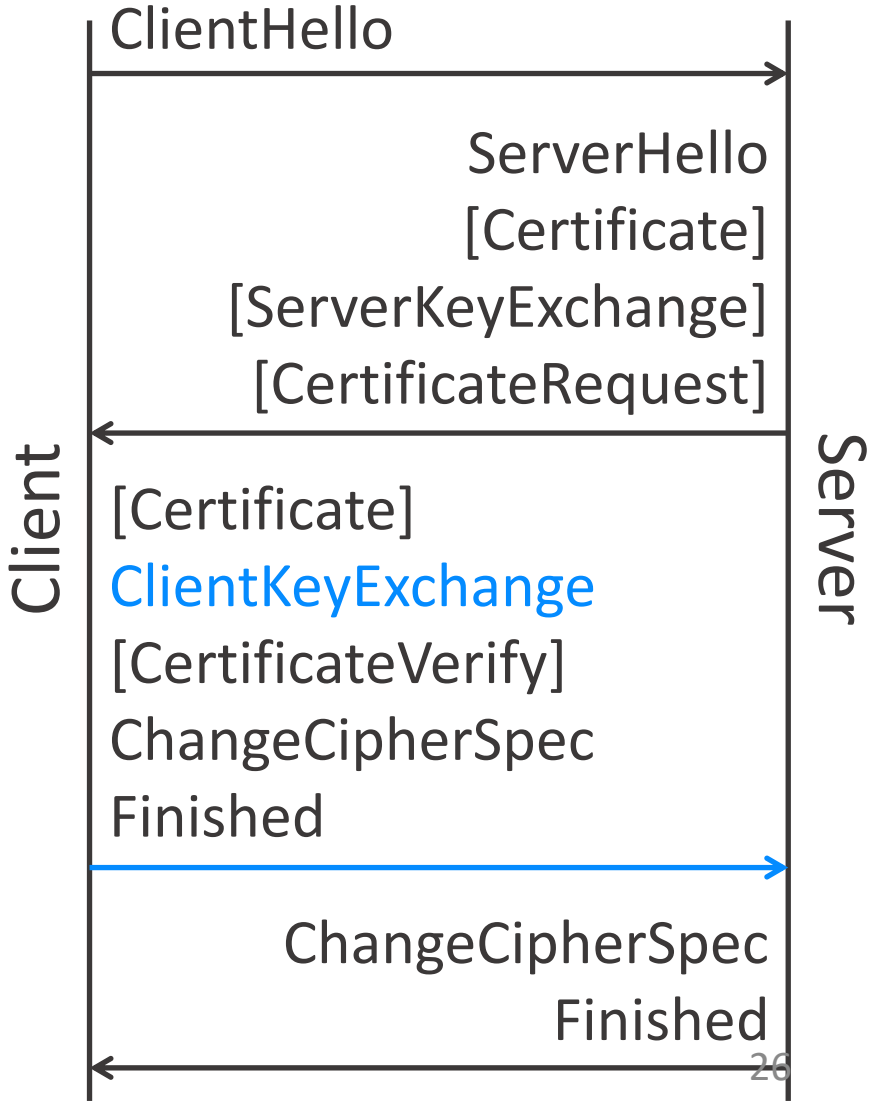
TLS Handshake

ClientKeyExchange

DH Public Key: aG

The server and client combine the DH parameters into the pre-master secret abG

The server and client combine the random parameters and pre-master into the session key



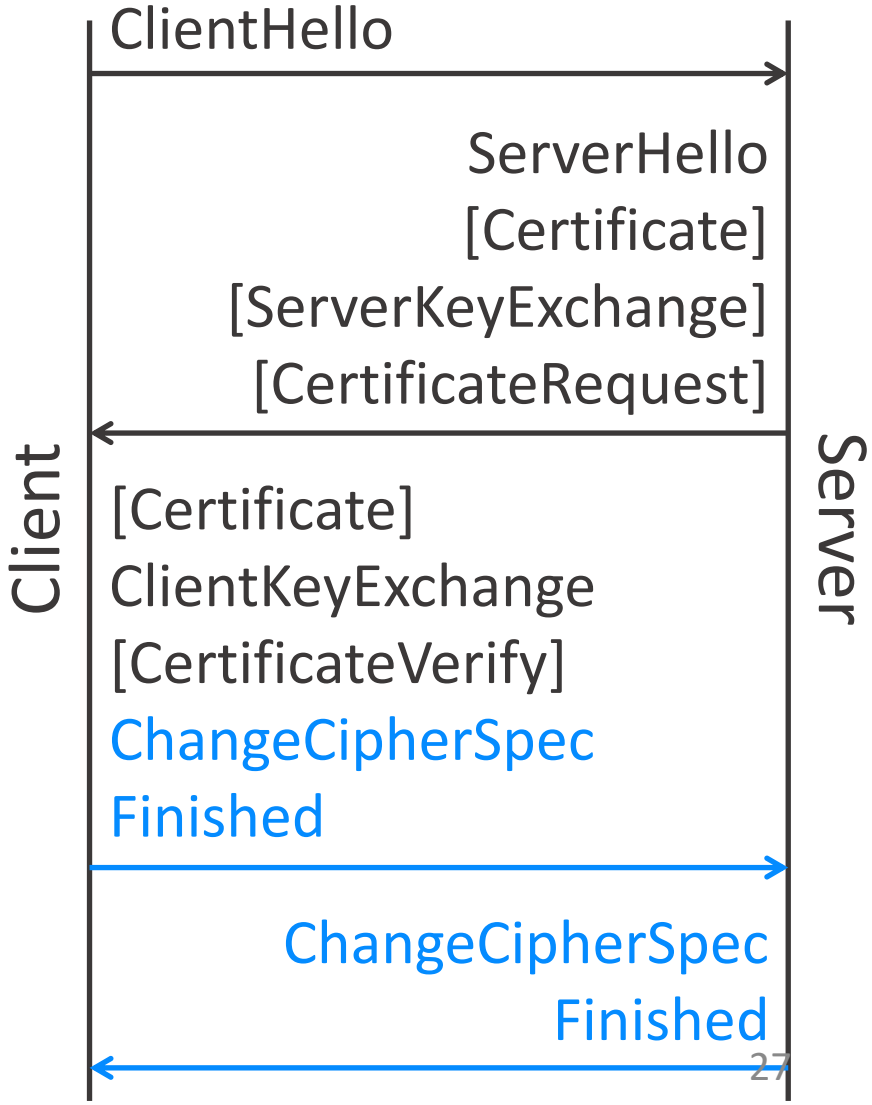
TLS Handshake

ChangeCipherSpec

Client then server send the ChangeCipherSpec message, which states that they are about to begin encryption

Finished

Client then server send Finish message, including a MAC of entire handshake for verification



TLS Vulnerabilities

- The man-in-the-middle vulnerabilities are usually countered using public-key authentication
- The majority of TLS problems are implementation
  - Heartbleed
- Protocol downgrade attacks are still a concern
  - many servers still allow weak cipher suites
  - FREAK and Logjam force the use of 512-bit keys

Summary

- Internet Security
- Cookies – Stealing & Tracking
- SSL / TLS
  - Vulnerabilities
- Cross-site Scripting
- Cross-site Request Forgery

Gollmann Chapter 16.5

Gollmann Chapter 18.2, 18.4, 18.5