COMP64 eek 3)

Aut agic

A NOTE ON ETHICS / LEGALITY

- UNSW hosting this course is an extremely important step forward.
- We expect a hig ism from you, meaning:
  - Respect the property of othe niversity
  - Always abide by the law and egulations
  - Be considerate of others to ensure everyone has an equal learning experience
  - Always check that you have written permission before performing a security test on a system

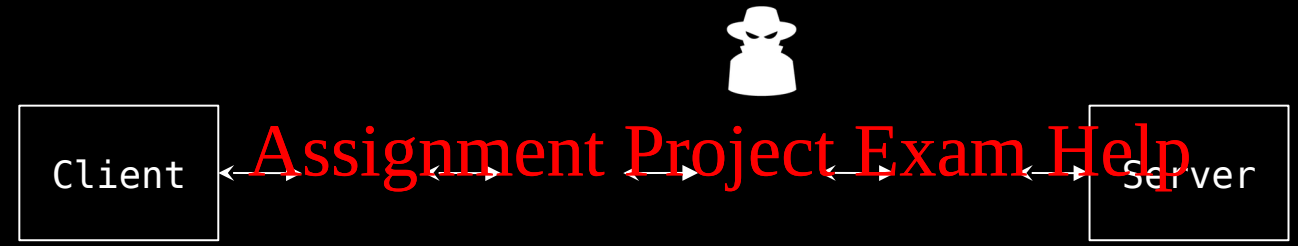
Always err on the side of caution. If you are unsure about

“NOT-A-HOMEWORK”

HTTP

- Client opens erver
- Client and server now have a al communication stream
- Requests and responses sent over this channel

The Problem with HTTP



- The requests cleartext
- A malicious party (Man in th the path of the requests/responses can read ify them
- This could be:
  - A router routing the packet
  - An attacker on the client's local network

HTTPS

- Transport Lay nted on top of TCP connection
- Client and server now have a communication stream
- TLS was previously known as Secure Sockets Layer (SSL)

HTTPS

- But how do we know the server is the one we intend to connect to?
- During the TLS handshake, the server sends a certificate indicating th intended domain
- The browser (ificate, showing a privacy warni

How do certificates work?

- A certificate a server has control of a domain
- The CA issues the server a p rtificate and corresponding private key
- CAs themselves may be signed by another CA, resulting in a “certificate chain” with the root certificate authority at the top
- Operating Systems come loaded with a set of trusted root CAs

Can HTTPS be MiTM’d?

- A malicious attacker can sit between client/server as a proxy
- However, it will not be able to present a certificate signed by a t
- Best it can d ed” certificate, which will re

How to be secure?

- Don’t serve any content over HTTP, redirect to HTTPs
- HOWEVER, this is still problematic as the initial HTTP request is still susceptible to MiTM

OVERVIEW

Authentication → Session Management → Access Control		
Is the user who they claim to be?	that us	Is the user llowed to access this thing?

SESSION MANAGEMENT in 1999

CLIENT

```
GET /bestcms.php?pid=123 HTTP/1.0
Host: www.lol.com
Cookie: username=uid0123456
Cookie: usertype=admin
```

SESSION MANAGEMENT in 2021

CLIENT

GET /1 HTTP/1.1

Co 890

SERVER

I recognize this SESSIONID. You are user ABCD
You have 1 item in your cart, item XYZ
You are not currently logged in.

ANATOMY OF A COOKIE

The main way we store session information is in a cookie.

Server → Client


```
Set-Cookie: SSID=abcdef;                                    an 2020 20:20:20 GMT;
Secure; HttpOnly
```



```
          name=value                the data t
          Domain                    specifies                  that the cookie belongs
     to
               Expires              date when                  d be deleted
               Secure               only send the cookie over secure connections
     (i.e. HTTPS)
               HttpOnly             disable access to the cookie from JavaScript
```


Client → Server

```
Cookie: country=aus; SSID=abcdef
```

ATTACKING SESSIONS

- Session Creation
  - How are sessions created? Can I fake my own session?
  - Can I attack the PRNG, and generate my own cookie?
  - Can I “fixate” a session?
- Session Handling / Tra
  - Can I steal the c g?)
  - Can I steal the cookie through redirect
  - What information does the site trust th ?
- Session Cleanup
  - What happens when I click “log out”?
  - Under what conditions is a session actually destroyed? What happens then?
  - Do sessions time out correctly?

Authentication → Session Management → Access Co)

Is the user who they claim to be?	Is it st that use	Is the user llowed to access this thing?

ACCESS CONTROL TYPES

- DAC (NTFS)
- RuleBAC and Role BAC (Attribute)
- Parameter-based

ON /admin AND OTHER THINGS...

Your user context is enough to get your ssh privkey.

TYPES OF (WEB) ACCESS CONTROL

Security through obscurity

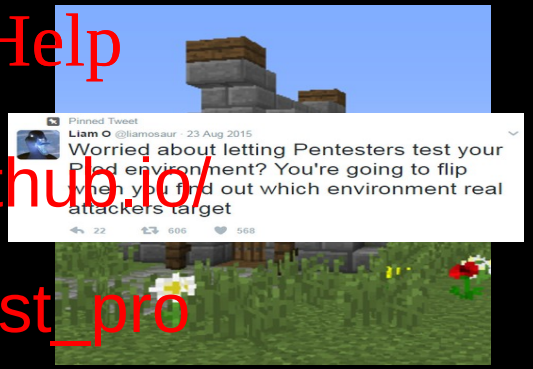
Rule-based acc

RBAC: HORIZONTAL vs VERTICAL

- Horizontal access control is making sure one user can't access a different user's data without permission
  - http://ban .id=12078
- Vertical access control is m nly administrative users can acc rative content
- Attacking vertical access control is commonly known as privilege escalation
  - http://bank.com/admin.php

ATTACKING ACCESS CONTROL

METHOD 1: BYPASS ENTIRELY



www.website.com

shop.website.com

blog.website.com

stage.website.com

db.website.com

api.website.com

dev.website.com

backup-syd.website.com

archive.website.com

s3 Buckets

github

pastebin

third party providers

mobile applications

analytics

etc etc…

ATTACKING ACCESS CONTROL

METHOD 1.5: ROBOTS.TXT

ATTACKING ACCESS CONTROL

METHOD 2: COPY LEGITIMATE USERS

ATTACKING ACCESS CONTROL

METHOD 3: ACTUAL TESTING

How does the application know what user role I am?
Are checks applied consistently throughout the application?
W ns?

What asp I control?
Can I impersonate anot ole?
What about content which ha control?

CYBER SUCCESS

TOWARDS BETTER ACCESS CONTROL

CLIENT

```
GET /statement.php?user_id=12078 HTTP/1.1
Host: www.bank.com
Cookie: SESSIONID=...
```


SERVER

id, authenticated

Does this type of user have to access the ‘view statement’ functionality?

NO

```
HTTP/1.1 302 Found
Location: /login.php
```

```
HTTP/1.1 403 Forbidden
```

Does this user have permission to view user 12078’s bank statement?

YES

```
HTTP/1.1 200 OK
...
```

CLIENT-SIDE ACCESS CONTROL

THE WEAKEST LINK

HOW TO PROTECT?

2FA

2FA

SMS
Mobile app
Key generator
Fingerprint

Channel-based
Location-based
Biometric

OAUTH

- Open id – Authentication Protocol
- Oauth – Authorization Framework(Oauth 1.0 & 2.0)
- OpenID Connect – built on top of Oauth 2.0

4 Types of Oauth grants:

- Authorization Grant
- Implicit Grant
- Resource Owner Credentials
- Client Credentials

OAUTH

https://www.oauth.com/playground/index.html

OAUTH

https://oauth.tools/

[DEFENSIVE] SECURING YOUR SESSIONS

- Minimize your attack surface
  - Your session should be managed on the server side where possible, using a single session token.
- Mostly mechani
  - Don’t let p (URLs, HTTP, etc)
  - Don’t let peo e to set them properly, log out)
  - Don’t let people generate tokens (secure PRNG, avoid rolling your own crypto, don’t allow users to supply tokens).
- Attention to detail is key.

READING MATERIAL (REFERENCE)

- OAuth2 Simplified
  https://aaronparecki.com/oauth-2-simplified/
- What the heck
  https://stormpath.com/blog/what-is-oauth
- How Anand hack
  https://medium.com/free-code-camp/how-i-hacked-tinder-accounts-using-facebook-accountkit-d5

WEEK 2-3 ASSESSMENT

- Hash collisions
- Don’t forget about automation!
- Don’t forget

Please call out if you get stuck.
Support one another, your tutors are here to help!

THAN TO US

questions? slack / email talk to us