CO 1

Web Application

WELCOME TO
COMP64{4,8}3

- 10 weeks, 5 topics across web security hybrid teaching.
- 6443 introduc es on securing code
- 6483 deep div lications
- Assessment:
  - 0% Week 1 Self-Assessment
  - 50% Coursework
  - 10% Mid-Semester
  - 40% Final Exam
- Course contact: cs6443@cse.unsw.edu.au

A NOTE ON ETHICS / LEGALITY

- UNSW hosting this course is an extremely important step forward.
- We expect a high standard of professionalism from you, meaning:
  - Respect th he university
  - Always abide by the law an regulations
  - Be considerate of others t yone has an equal learning experience
- Always check that you have written permission before performing a security test on a system

PLEASE BE SUPER CAREFUL WHENEVER YOU’RE GENERATING NETWORK TRAFFIC

WHAT IS WEB SECURITY?

In the age of Electron, mobile WebViews and embedded Chromium, what does “web application” even mean?

EXAMPLE: SERVER-SIDE ISSUES

EXAMPLE: JS KEYLOGGER

How much do you trust your browser to keep you safe?

WHAT IS WEB SECURITY?

CORPORATE CITIZENSHIP REMIX

1. Code exec on web server

2. AWS secret keys leaked, XL GPU instances mining crypto.

3. Website down for 30 mins

4. SQLi in web store, allowing purchases for $0.01

BRIEF [illegible] ML/JS

This is core skill for this co [illegible] you get stuck.

HTML

- Each web page is an XML-like tree.
- HTML is made of elements
  - Attributes
  - Script
  - Styles
- HTML5 contains new elements allowing rich media
- Older elements (e.g. iframes) steadily seeing less use.

BROWSER

- Client-side application to render web content: HTML, JS, dynamic content
- Behaviour not 100% identical
- Extremely complex

- Somewhat out-of-scope:
  - Embedded browsers
  - Mobile browsers
  - MSHTA

Q: How many requests does it take to get /r/netsec? A: 102.

HTTP

- Web traffic is (mostly) done via HTTP.
  - Resources referred to by URI
  - Not necessar
  - Not necessar
- HTML/JS can allow users to send various HTTP requests
- Headers indicate various options to the web server
  - Cookies indicate session state
  - Logout vs real logout
- GET, POST, OPTIONS, HEAD, etc

WEB SERVER

- Code exists on the web server to process user input
  - Not visible
  - Before: PHP
  - Now: .NET,
- Proliferation of frameworks + framework complexity
  - Security often built-in
  - Vastly improved from good old days of PHP/MySQL
- Web Pages vs API’s
  - API Firewalls

JAVASCRIPT

- Allows web pages to run dynamic c user's browse
- Large ecosyst third-party libraries
- Secured with (mostly) sandboxing in browsers
- NodeJS*

SAME ORIGIN POLICY

- An Origin is a resource identified by a URI
- A “page” (what you see) can have multiple origins
  - Resources loaded from elsewhere
  - Frames
- Restrictions
  - Script from Origin 1 can send data to Origin 2
  - ... but cannot see the response
  - Script from Origin 1 cannot access data from Origin 2
- XMLHTTPRequest

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

HOW DO [illegible] APPS?

This is core skill for this co[illegible] you get stuck.

THE “ATTACKER MINDSET”

- Some other good names:
  - Null byte
  - A PNG fil
  - Newline
  - XML/JSON/SQL
  - OS Commands
  - Backticks (Unix)
  - Large/small names
  - Strange character sets

A THOUGHT EXPERIMENT

Q: How would you attack an image upload?

BASIC REQUEST AUTOMATION

You need to be able to automate requests for this course.

INPUT SANITISATION

Assume your users are doing wh st done.

Understand wha visible to your user.

UNDERSTANDING YOUR ENVIRONMENT

USENIX Enigma 2016 - NSA TAO Chief on Disrupting Nation State Hackers
https://www.youtube.com/watch?v=bDJb8WOJYdA

THE SAGA OF LEFT-PAD

MODERN WEB COMPLEXITY AND YOU

https://www.theregister.com/2016/03/23/npm_left_pad_chaos/

Q: What prevents anyone from uploading malware to package repositories?

WHY IS WEBSEC ~~HARD~~ NEAR-IMPOSSIBLE

- Near-infinite complexity (us [illegible] ogy, inputs)
- Generally untrusted environment
- Critical always-on functionality
- Increasingly modular programming / framework proliferation
- Rapidly evolving techniques

WEEK 1 (NON-ASSESSABLE) SELF-TEST

- Are you able to set up a web server, and successfully have it serve simple dynamic content?
- Are you able [illegible] traffic from a browser, and [illegible]?
- Do you know the difference b[illegible]r-side and client-side content?

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

REFERENCE: TOOLS OF THE TRADE

- Python (www.python.org)
  - requests
  - http.serve
  - Not mandat ke.
- Burpsuite, or .
  - OWASP Zap
  - Fiddler
- Your browser’s developer tools
- Basic JavaScript (javascript.com)

THAN TO US

questions? email ing