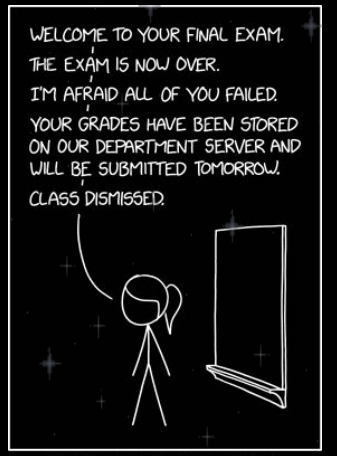
CMSC388U

Web I & II

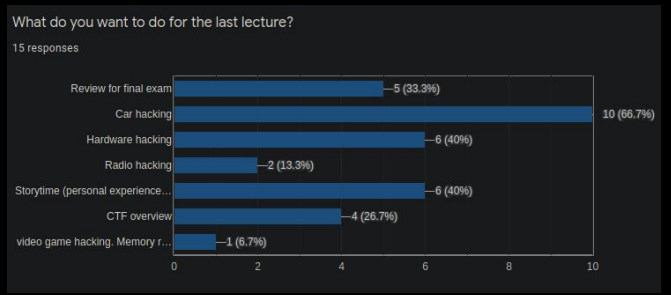
COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Announcements

- Web I and Web II are combined - today!
- HW 10 (this week's) will be due by next lecture
  - This is the final write-up!
- Topics for 4/30 and 5/7 are up for discussion
  - Keep and eye for for new poll on Piazza/ELMS
- Final Exam!
  - Will be CTF-like, with an overview of the concepts covered in the class
    - Specifics will be announced soon
  - Will be open for a week (will not take that long!)
    - 5/7 @ 3:30pm ET to 5/14 @ 3:30pm ET



CYBERSECURITY FINAL EXAMS



Homework #9 Review

- **Question 1A**
  - Y21zYzM4OHV7ZW5jb2RpbmchPWVuY3J5cHRpb24tcmlnaHQ/fQ==
  - Base64 decode
  - `echo Y21zYzM4OHV7ZW5jb2RpbmchPWVuY3J5cHRpb24tcmlnaHQ/fQ== | base64 -d cmsc388u{encoding!=encryption-right?}`
- **Question 1B**
  - pzpf388h{prnfre_jnfa'g_gung_fzneg}
  - ROT13 decode!
  - cmcs388u{ceaser_wasn't_that_smart}

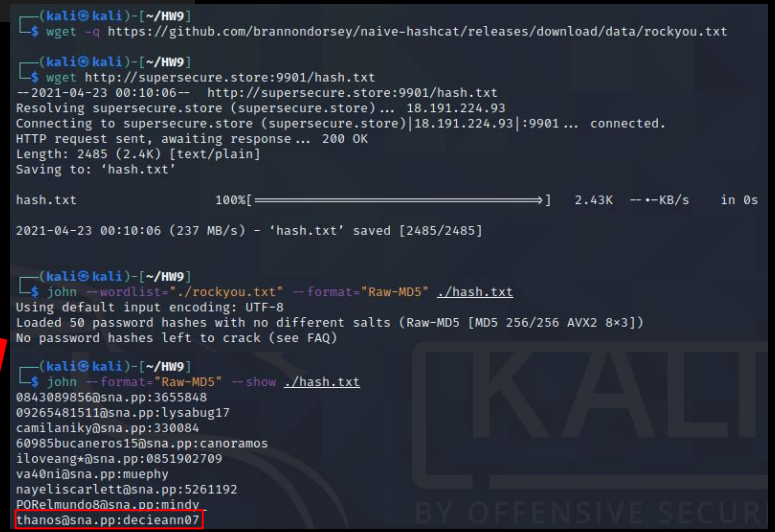
A

Homework #9 cont.

- Question 2
  - Issue with how original fake breach was generated

```
for line in $(shuf -n 50 rockyou.txt) do
    pw_hash=$(cat rockyou.txt | shuf -n 1 | tr -d "\n" | md5sum | awk '{print $1}')
    echo "$line@sna.pp:$pw_hash"
done
```

- "Raw-MD5" should have taken seconds to crack
  - Unsalted MD5 is not a strong algorithm against brute forcing



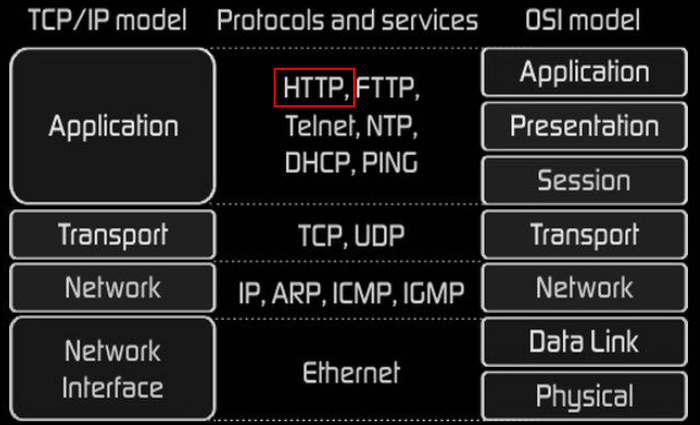
This would have a different output if it wasn't run before

HTTP

- HyperText Transfer Protocol

- What is it?
  - Protocol that is used to interact with web applications
  - Interfaces between web browser and internet

- OSI Model?
  - Open Systems interconnection model
  - Demonstrates modern internet/network abstraction
  - Modern CS mostly stays from Network up
    - Lots of fun to be had lower!



https://medium.com/@rrugamba/hypertext-transfer-protocol-http-f7c0665f7695

HTTP Response Codes

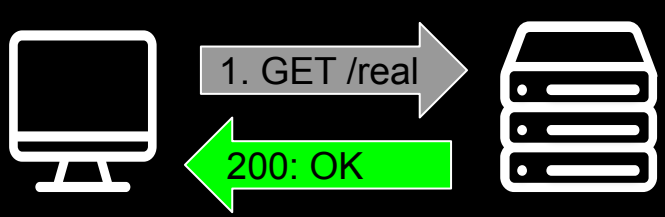
- When you send a request to a web server you'll get a response
- Depending on the status of the request the corresponding resp code will be sent back to you
- Example: 404 not found pages!

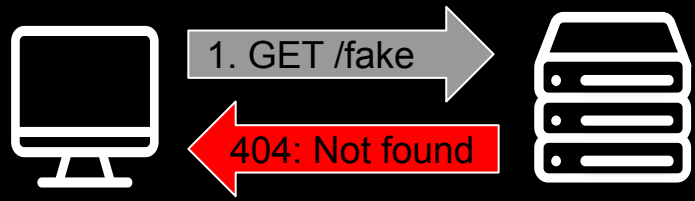
- **100:** Informational Responses
- **200:** Successful
  - Success/Ok
- **300:** Redirecting
  - 301: Permanent Redirect
  - 302: Temporary Redirect
  - 304: Not Modified
- **400:** Client Errors (your fault)
  - 401: Unauthorized
  - 403: Forbidden
  - 404: Not found
- **500:** Server Errors (not your fault)
  - 502: Bad gateway
  - 503: Service unavailable

A

HTTP Request Methods

- Within HTTP there are multiple types of requests you can make to webservers
- When you want to receive content - browser sends a GET request
- When you want to send content - browser uses a POST request
- There is also - `PUT`, `OPTION`, `HEAD`, `DELETE`, `CONNECT`, `TRACE`, `PATCH`





HTTP Cookies

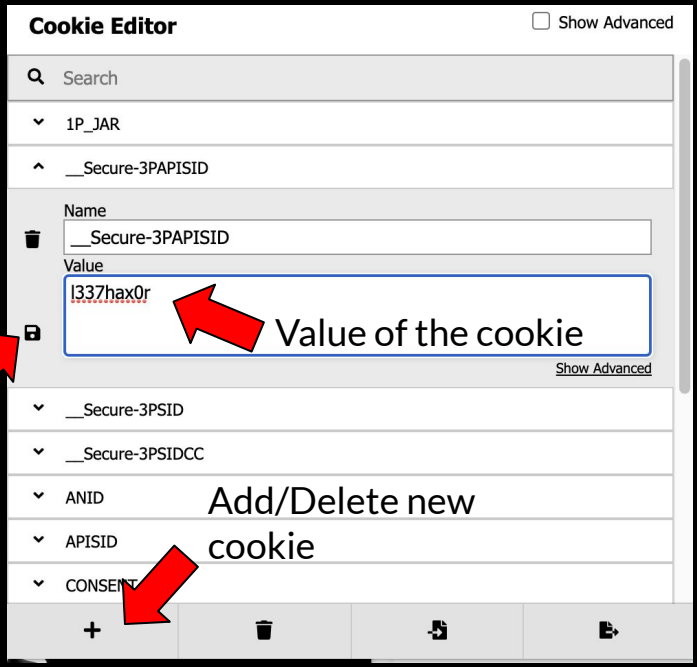
- “Small piece of data stored in the web browser that is used to remember stateful information (settings, username, password, logged in, etc)
- Stored client side
  - Can be modified by user!
- Persistent, server can set expiry time
- Used to track users across websites
- Secure - specifies that they can be transferred with only HTTPS

- Extra reading: https://samy.pl/evercookie/

A

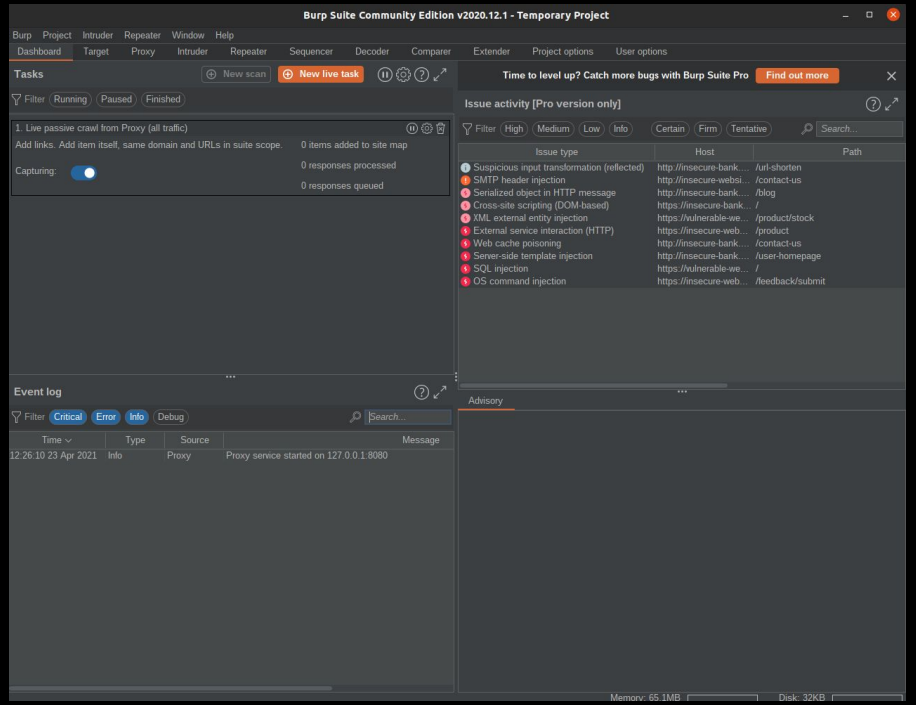
Modifying Cookies

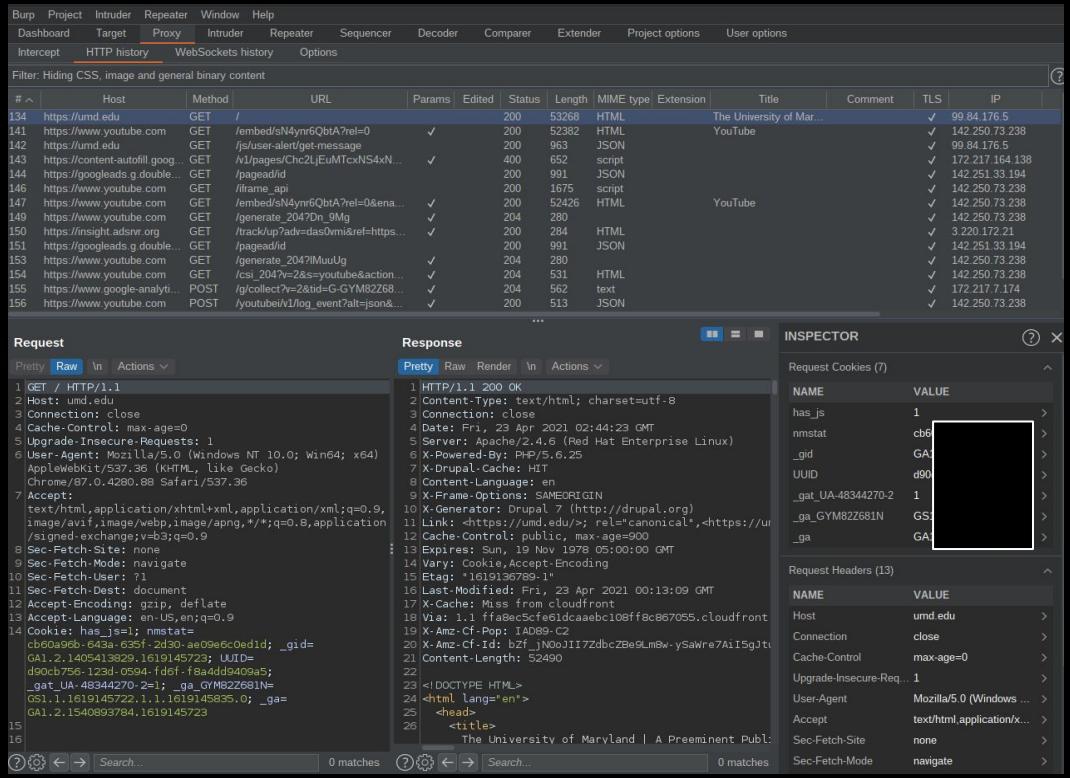
- Tons of browser plugins for this
  - EditThisCookie
  - Cookie-Editor
  - Natively in Chrome/Firefox Dev tools
- Important values:
  - Name (what js looks for)
  - Value
  - State
  - Expiry
  - Secure



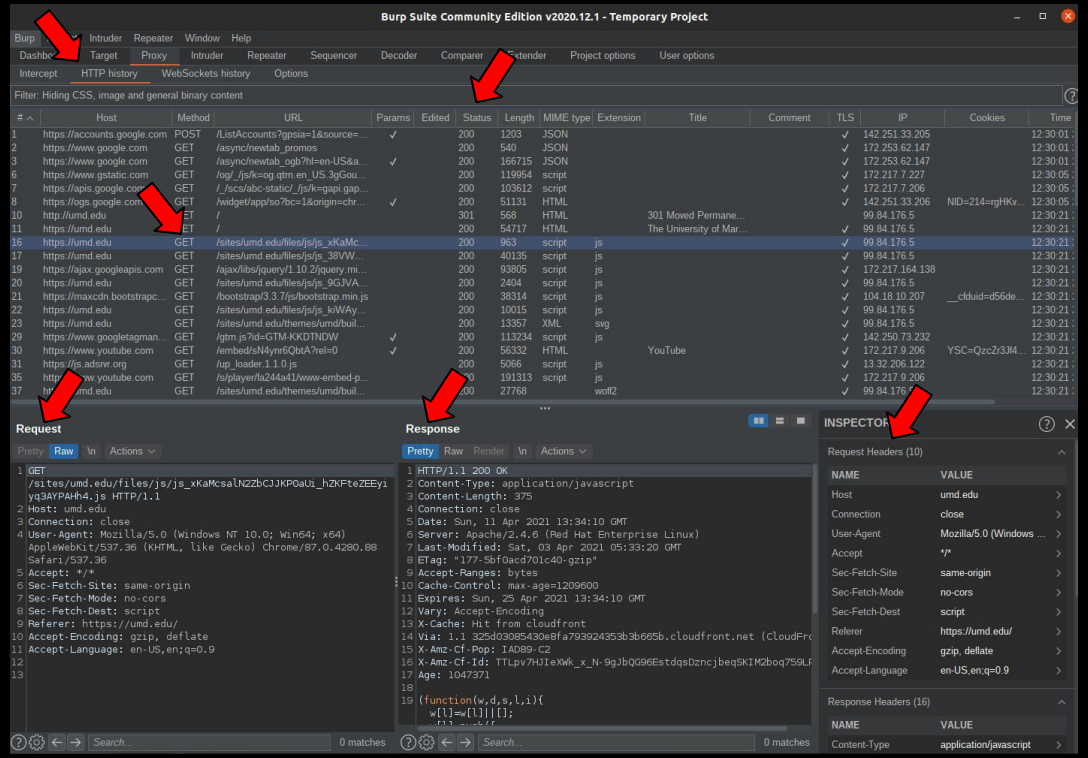
BURP

- Great multi tool for web security analysis
- General workflow:
  - Proxy traffic through burp
    - Either make your browser/device proxy through or use burp's embedded browser
  - Intercept/examine/modify/etc the traffic burp gets




Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSockets history Options
Filter: Hiding CSS, image and general binary content
Host Method URL Params Edited Status Length MIME type Extension Title Comment TLS IP
134 https://umd.edu GET / 200 53268 HTML The University of Mar... 99.84.176.5
141 https://www.youtube.com GET /embed/sN4ynr6QbtA?rel=0 ✓ 200 52382 HTML YouTube 142.250.73.238
142 https://umd.edu GET /js/user-alert/get-message 200 963 JSON 99.84.176.5
143 https://content-autofill.goog... GET /v1/pages/Chc2LjEuMTcxNS4xN... ✓ 400 652 script 172.217.164.138
144 https://googleads.g.double... GET /pagead/id 200 991 JSON 142.251.33.194
146 https://www.youtube.com GET /iframe_api 200 1675 script 142.250.73.238
147 https://www.youtube.com GET /embed/sN4ynr6QbtA?rel=0&ena... ✓ 200 52426 HTML YouTube 142.250.73.238
149 https://www.youtube.com GET /generate_204?Dn_9Mg ✓ 204 280 142.250.73.238
150 https://insight.adsrvr.org GET /track/up?adv=das0vmi&ref=https... ✓ 200 284 HTML 3.220.172.21
151 https://googleads.g.double... GET /pagead/id 200 991 JSON 142.251.33.194
153 https://www.youtube.com GET /generate_204?lMuuUg ✓ 204 280 142.250.73.238
154 https://www.youtube.com GET /csi_204?v=2&s=youtube&action... ✓ 204 531 HTML 142.250.73.238
155 https://www.google-analyti... POST /g/collect?v=2&tid=G-GYM82Z68... ✓ 204 562 text 172.217.7.174
156 https://www.youtube.com POST /youtubei/v1/log_event?alt=json&... ✓ 200 513 JSON 142.250.73.238
Request
Pretty Raw \n Actions
1 GET / HTTP/1.1
2 Host: umd.edu
3 Connection: close
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Sec-Fetch-Site: none
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: has_js=1; nmstat=cb60a96b-643a-635f-2d30-ae09e6c0ed1d; _gid=GA1.2.1405413829.1619145723; UUID=d90cb756-123d-0594-fd6f-f8a4dd9409a5; _gat_UA-48344270-2=1; _ga_GYM82Z681N=GS1.1.1619145722.1.1.1619145835.0; _ga=GA1.2.1540893784.1619145723
15
16
Search... 0 matches
Response
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Date: Fri, 23 Apr 2021 02:44:23 GMT
5 Server: Apache/2.4.6 (Red Hat Enterprise Linux)
6 X-Powered-By: PHP/5.6.25
7 X-Drupal-Cache: HIT
8 Content-Language: en
9 X-Frame-Options: SAMEORIGIN
10 X-Generator: Drupal 7 (http://drupal.org)
11 Link: <https://umd.edu/>; rel="canonical",<https://u
12 Cache-Control: public, max-age=900
13 Expires: Sun, 19 Nov 1978 05:00:00 GMT
14 Vary: Cookie,Accept-Encoding
15 Etag: "1619136789-1"
16 Last-Modified: Fri, 23 Apr 2021 00:13:09 GMT
17 X-Cache: Miss from cloudfront
18 Via: 1.1 ffa8ec5cfe61dcaaebc108ff8c867055.cloudfront
19 X-Amz-Cf-Pop: IAD89-C2
20 X-Amz-Cf-Id: bZf_jNOoJII7ZdbcZBe9Lm8w-ySaWre7AiI5gJtu
21 Content-Length: 52490
22
23 <!DOCTYPE HTML>
24 <html lang="en">
25 <head>
26 <title>
The University of Maryland | A Preeminent Publi
Search... 0 matches
INSPECTOR
Request Cookies (7)
NAME VALUE
has_js 1
nmstat cb6
_gid GA1
UUID d90
_gat_UA-48344270-2 1
_ga_GYM82Z681N GS1
_ga GA1
Request Headers (13)
NAME VALUE
Host umd.edu
Connection close
Cache-Control max-age=0
Upgrade-Insecure-Req... 1
User-Agent Mozilla/5.0 (Windows ...
Accept text/html,application/x...
Sec-Fetch-Site none
Sec-Fetch-Mode navigate


The University of Maryland
Not secure | umd.edu
4 MARYLAND
WE CAN PREVENT THE SPREAD OF CORONAVIRUS 4 EVERYONE AT UMD.
PODEMOS PREVENIR LA PROPAGACIÓN DEL CORONAVIRUS PARA TODOS EN UMD.
UNIVERSITY OF MARYLAND
Search
Inside UMD
News & Events
Admissions
Support UMD
4 Maryland
Find Semester Updates and Learn About Critical Health Behaviors to Prevent the Spread of Coronavirus 4 Everyone at UMD
Read More
Wash your hands often and sanitize frequently used surfaces
Lávese las manos con frecuencia y desinfecte las superficies de uso frecuente

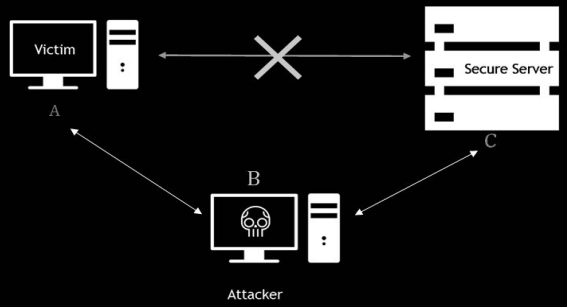

Burp Suite Community Edition v2020.12.1 - Temporary Project
Burp
Intruder
Repeater
Window
Help
Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
User options
Intercept
HTTP history
WebSockets history
Options
Filter: Hiding CSS, image and general binary content
Request
Response
INSPECTOR
Request Headers (10)
Response Headers (16)

V

SSLstrip & HSTS, a story

- Example of a downgrade attack
- Evil access point that redirects HTTPS queries to HTTP, which can be intercepted
- Fixed with “HTTP Strict Transport Security” (HSTS)

- Timeframe:
  - SSLstrip (10 years ago)
  - HSTS
  - SSLstrip+/SSLstrip2 (7 years ago)
  - HSTS updated



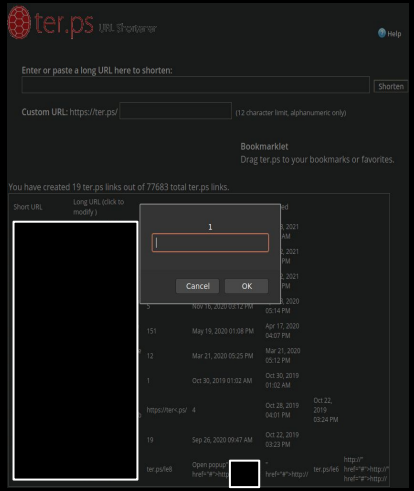
https://arnavtripathy98.medium.com/sslstrip-tool-for-fighting-https-e917d76b0d45

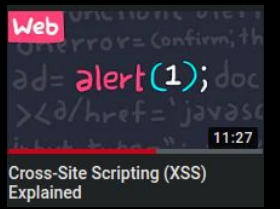
XSS

- “Cross Site Scripting”
  - Occurs when unsanitized user input is stored/printed back out
    - User input can contain valid HTML/JS/etc.
  - <script><alert>(“XSS”)</alert></script>

- Different types:
  - Reflected (aka “Self-XSS”)
    - Only shown against user who inputs the XSS
  - Stored
    - Becomes stored on the site, affects all users
  - DOM
    - https://example.com/page.html?variable=<script>alert(“XSS”)</script>
  - Blind
    - Hope that the input will be appropriately rendered somewhere

- Lots of possibilities!
  - https://github.com/payloadbox/xss-payload-list



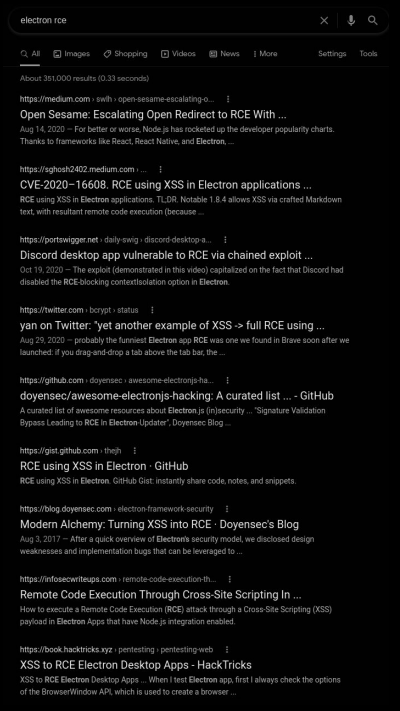


^ Great video on the subject!

XSS: Electron

- ElectronJS is effectively a chromium based browser
  - Popular electron apps include:
    - Discord
    - Slack
    - Atom
    - VSCode
    - And many more!

- Note: With ElectronJS XSS can be a big issue!
  - If “nodeIntegration” is set to true (used to be default in electron):
    - XSS becomes RCE!
      - <script> require('child_process').exec('whoami'); </script>



Ess-queue-ehl / See-quill

- Databases, in web?
  - Just a place to store data! Usernames, passwords, settings, etc
  - **Main types:**
    - MySQL
    - Postgres
    - MongoDB
    - SQLite

- **S**tructured **Q**uery **L**anguage (SQL)
  - Used to communicate with SQL databases
  - Websites will build queries from user input
  - If done wrong can allow for leaking of information from said database

A

SQLi

- SQL Injection
- When user input isn't "sanitized"
  - Dangerous characters such as ' " --)
- You can make queries you aren't supposed to!
- Ex: Information supplied by user - such as a username - is passed into a query that will do something with that info (login, sign up, etc)
- Example:
  - `SELECT * FROM users WHERE username='admin' ... <- Vulnerable!`
  - `SELECT * FROM users WHERE username='admin'--' ...`

SQL Comment

A

SQL Map

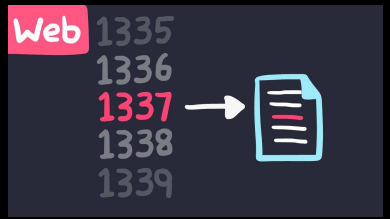
- Tool that automates the discovery of SQLi
- Powerful but you gotta know how to use it
- Usage: `python sqlmap.py -u "http://testsite/get_thing.php?id=1"`

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.3.4.44#dev}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

IDORs

- "InDirect Object Response"
  - If you login to example.com and see:
    - https://example.com/?userID=155
  - What would happen if you change the URL to say:
    - https://example.com/?userID=156?

- If no checks were placed (other forms of authentication), incrementing/decrementing the userID could cause you to be logged in as another user!
  - This example is simplified, but not far off!



^ Great video on the subject!

LFI

- Local File Inclusion:
  - Occur when the web application allows users to read files from victim machine
  - `http://192.168.80.134/dvwa/vulnerabilities/fi/?page=../../../../../../etc/passwd`
  - PHP is usually culprit

Directory traversal to root, read etc/passwd

- Exploiting LFI
  - If a user is able to upload file - upload some malicious `script.php`
  - Then you can access the file using LFI
  - File may be executed

OWASP - Top X

- There are many categories of web based vulnerabilities, such as
  - XXE, CSRF, SSRF, etc.

- **O**pen **W**eb **A**pplication **S**ecurity **P**roject
  - List the top threats to web applications
    - https://owasp.org/www-project-top-ten/
  - Juice Shop: great place to practice web hacking stuff
    - https://owasp.org/www-project-juice-shop/

A

Burp Demo?

(Time permitting)

For next time

- Final HW writeup due by next lecture :)
- Keep an eye out for the ELMS+Piazza poll to vote on the topics for the next two lectures
- Don’t be afraid to ask on Piazza or schedule office hours!