Mason Competitive Cyber

Meeting 3: Web Exploitation

Upcoming Events

- Boston Key Party
  - Online CTF
  - Today 8PM - Sunday 2/26 8PM
  - Other CTFs this weekend: VolgaCTF (russian),
- Cryptoparty
  - In-person cryptography workshop at GMU
  - Tomorrow 9:30am-3:30pm in SUB I 3B
  - go.gmu.edu/cryptoparty
- iCTF
  - Friday March 3rd, 12PM-8PM for academic league
  - 24 hours for public league (can be in both)

News

- Cloudflare data leak bug
  - Cloudflare is supposed to help secure websites
  - “we make the Internet work the way it should”
  - Bug was caused by HTML parser (email obfuscation, Automatic HTTP rewrites, Server-side Excludes)
  - Leaked cookies, POST requests, HTTP data
- SHA1 Collision
  - SHA1 = hashing algorithm (used for integrity verification)
  - Generates unique string for file
- Verizon buying Yahoo for 4.48 billion
  - 350 million less because of August 2016 Yahoo breach

News

- Financial industry cybersecurity regulations in NY take effect March 1
  - Retain Chief Information Security Officer (can use 3rd party companies to fulfill)
  - report (nonpublic info) breaches to Superintendent of Financial Services within 72 hours
  - use MFA

Web Exploitation

- 52.91.34.172
  - Exploitable clone of Patriotweb
- If any links take you to a .gmu.edu address… don’t fuck with it!

Web Exploitation things to look at

- Right click, view page source
  - Could also use right click, Inspect element
- Login boxes
- Cookies
  - Chrome - EditThisCookie
  - Firefox - Cookies Manager+
- URL

Command Injection

- User inputs command that runs on server
  - Input in URL, in Login box, in a form that the user fills out, etc.
  - Probably linux commands
- Code injection
  - Like command injection but with code (PHP, JavaScript)

Cross-site Scripting (XSS)

- Kind of like Code Injection except malicious code (usually JavaScript) is executed client-side instead of server-side

XSS Types

- Reflective XSS
  - Usually means JavaScript in URL
  - Chrome and FF try to stop this
  - Not persistent because it's all in one HTTP request and response
- Stored XSS
  - Persistent because stored in text on page (like in a comment)
  -
- Self XSS
  - U R dumb
  - Someone social engineered you into opening browser dev tools and running it on yourself

Preventing XSS

- Filter HTML, JavaScript, and PHP from user input
- How?

< > ( ) { } [ ] " ' ; / \

- Escape the above characters

Cookies

- Store information about user's session
- In real life, attackers trying to get them
- In CTFs, may have to modify value of your cookie

SQL Injection

- SQL = Structured Query Language
  - Used for accessing databases
- Normal user login:
  - User enters username and password
  - Generates a query
- SQL Injection login:
  - User generates own query
  - Might view, change, delete, or steal data from database
  - Might log in
- go.gmu.edu/sqli

SQL Injection

- SELECT * FROM Users WHERE Username='$username' AND Password='$password'
- admin = 1' or '1' = '1
- password = 1' or '1' = '1

- Most CTF SQL injection involves logins
- In real world, could also be done with URL

Preventing SQL Injection

- Sanitize database inputs by using escape strings
- Limit length of login input
- Don’t store credentials in plaintext
- Hash them and SALT THE HASHES
  - Hashing is theoretically one-way aka impossible to crack
  - Rainbow tables associate words with hashes then search to match hash
  - Salting = adding random string to password before hashing
  - Salt usually stored with hash

If you want more...

- go.gmu.edu/hackthissite
  - Hackthissite.org has easy webpages to exploit
  - Tells you which technique to use
  - Document is a walkthrough of all of the problems
- OWASP NoVA or OWASP DC
  - Open Web Application Security Project
  - Next OWASP NoVA meeting- March 16, 6PM-9PM
- OWASP Top 10
- go.gmu.edu/owasp
- 52.91.34.172