A2: Broken Authentication and Session Management

But first…

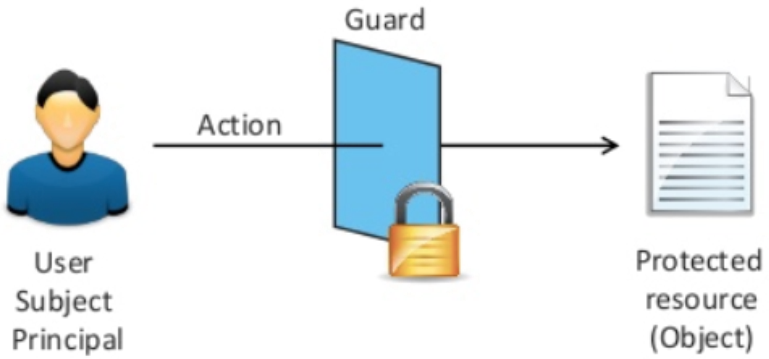
Authentication, Authorization, Sessions

Authentication

- Determining user identity
- Multiple ways
  - What you know (password)
  - What you have (phone, RSA SecureID, Yubikey)
  - Who you are (fingerprint, eye scan)
  - Where you are (GPS, IP address)

Authorization

- Ensure users only perform actions in their privilege level or role (A4/A7)
- **Policy** to set which **users** are allowed which **actions** on which **objects**



  - Users
    - User, external web application, internal web application, database
  - Actions
    - Read, Write, Execute, Append, Create, Delete
  - Objects
    - Resource (network, operating system, files, web application, database, etc.)
  - Policy
    - Discretionary Access Control (object owner decides)
    - Mandatory Access Control (system/administrator decides)
      - Stronger limits on activity
    - Role-Based Access Control (system decides based on user role)

Session management

- Embodiment of user's authentication and authorization for duration of the user's interaction with service
- Sessions used to maintain authentication and authorization state over stateless HTTP
- Done via multiple mechanisms sent on each request
  - HTTP cookies
  - URL parameters (not recommended)
  - JavaScript web tokens
  - HTML5 session storage
  - Hidden Form fields

A2 – Broken Authentication and Session Management

Example: Guessable credentials

- Common passwords and weak passwords allowed

RANK	PASSWORD	CHANGE FROM 2014
1	123456	Unchanged
2	password	Unchanged
3	12345678	1 ↗
4	qwerty	1 ↗
5	12345	2 ↘
6	123456789	Unchanged
7	football	3 ↗
8	1234	1 ↘
9	1234567	2 ↗
10	baseball	2 ↘
11	welcome	NEW
12	1234567890	NEW
13	abc123	1 ↗
14	111111	1 ↗
15	1qaz2wsx	NEW
16	dragon	7 ↘
17	master	2 ↗
18	monkey	6 ↘
19	letmein	6 ↘
20	login	NEW
21	princess	NEW
22	qwertyuiop	NEW
23	solo	NEW
24	password	NEW
25	starwars	NEW

Example: Common credentials

- Default passwords or security credentials left unchanged
- Allows easy, brute-force attacks by an adversary
- Example lists
  - http://www.defaultpassword.com/
  - https://wiki.skullsecurity.org/Passwords
  - Metasploit's Mirai lists, Dyn IoT attack (10/2016)
  - Repository of passwords dumped from vulnerable sites that stored them in the clear

```
apt-get install seclists
ls /usr/share/seclists
```

Example: Guessable resets

- Guessable password reset questions
  - Information publicly available or easily inferred
  - Anonymous hack of Sarah Palin's Yahoo account 2008
    - ZIP, birthdate, where she met spouse
- Guessable "Change My Password" links

Example: Vulnerable authentication processes

- No rate-limits on authentication attempts and failures
  - Via web front-end and web API
- Side-channel attacks
  - Username checked before password instead of simultaneously
  - Non-time-constant string comparison vulnerability (Program #2)

Example: Password storage problems

- Passwords stored in the clear instead of hashed
  - Single security compromise gives up all user credentials
  - Credential reuse across sites makes problem worse

```
alpha:Alfred Phangiso:Alfie99
bravo:David Bravo:aprilVII2004
charlie:Charles Windsor:password
duck:Philip Ducklin:password
echo:Eric Cleese:norwegianBlue
```

- Password hashes used, but stored without a “salt”
  - Salt is random data hashed with password
  - Attacker can employ precomputed dictionary attack via rainbow tables
  - Rainbow table lookup https://crackstation.net

```
alpha:Alfred Phangiso:D5D459FFDFCE..7DCF3651919B
bravo:David Bravo:4620F0E4F362..9C88A6B3BD09
charlie:Charles Windsor:5E884898DA28..EF721D1542D8
duck:Philip Ducklin:5E884898DA28..EF721D1542D8
echo:Eric Cleese:89E1D86C63B8..6D0CC7424EDC
```

Example: Password storage problems

- Salt added to prevent rainbow table lookup

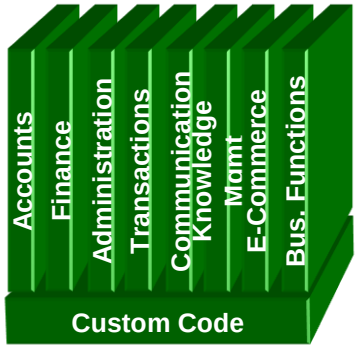
```
#username:realname:salt:hash

alpha:Alfred Phangiso:0050B9..D970C4:1DC87318B512..A338DC5543EB
bravo:David Bravo:B5916E..325460:B954EF627298..3D1B21FC9DD0
charlie:Charles Windsor:49C20B..78418B:9A0A75EAB9B5..30A0253B6137
duck:Philip Ducklin:71E831..166D6A:D721A297603F..723B175381E4
echo:Eric Cleese:864E2A..A346B7:BF19240CE02E..D45DEFDB952B
```

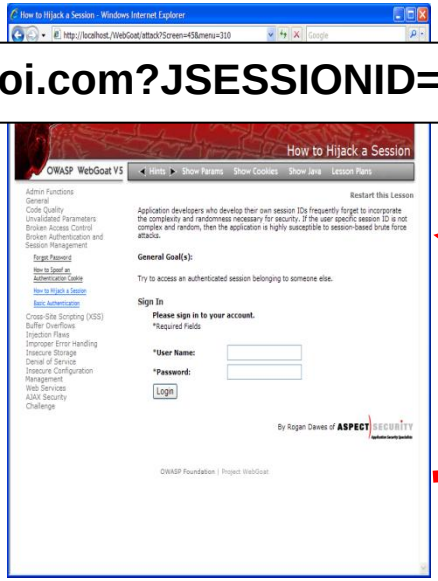
- But, cryptographic hashes used instead of password hashes
  - Cryptographic hashes intended to be *fast*
  - But, if one has salt and hash, a brute-force dictionary attack is *still* fast against weak passwords

Example: Session IDs carried in URLs

1 User sends credentials



www.boi.com?JSESSIONID=9FA1DB9EA...



Site uses URL rewriting
(i.e., put session in URL)

2

3 User clicks on a link to http://www.hacker.com in a forum

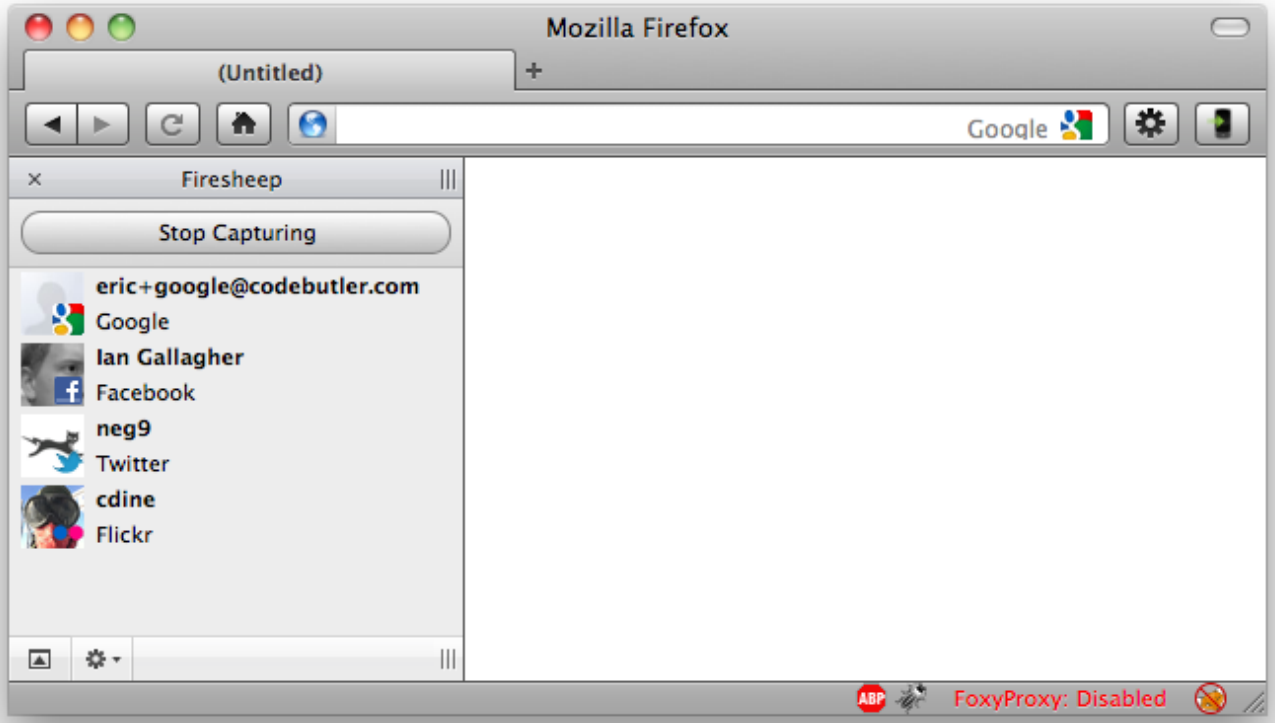
Hacker checks referrer logs on www.hacker.com and finds user's JSESSIONID

4

5 Hacker uses JSESSIONID and takes over victim's account

Example: Exposed tokens and cookies

- Cookies sent over HTTP
  - Dump via Burp, Wireshark, or browser tools
  - Session hijacking, request forgery



Example: Vulnerable tokens and cookies

- Repeated or unchanging session tokens
  - Persistent access if captured
- Predictable generation of session tokens
  - Blind hijacking of authorized sessions
- Unsigned session tokens
  - Forging authorized sessions (JavaScript Web Token)

Example: Vulnerable tokens and cookies

- Insecure management of session information at server
  - User sessions stored in server insecurely
  - PHP active sessions directory

```
# cat /var/lib/php5/sess_o8d7lr4p16d9gec7ofkdbnhm93
pentesterlab|s:12:"pentesterlab";
```

  - Global session management in web frameworks
    - Vulnerable to side-channel attacks for co-located apps (`natas`)

Example: Case sensitivity mismatch

- Database and web application handle case differently
- Creating a user with an existing username
- Allow access to “admin” account via “Admin” or “ADMIN”

A2 – Prevention

Authentication Cheat Sheet
https://www.owasp.org/index.php/Authentication_Cheat_Sheet

Password Storage Cheat Sheet
https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet

Forgot Password Cheat Sheet
https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet

Session Management Cheat Sheet
https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

Verify authentication architecture

- Use session management provided by your framework
- Ensure SSL/TLS protects all credential transmission
  - Form data, HTTP Auth, etc.
- Limit exposure of credentials (e.g. do not send repeatedly)
- Ensure secure storage of credentials at end points

Password management best practices

- Remove unnecessary accounts and default credentials
- Enforce strong password policies
  - Ensure capital+lower-case letters, numbers, symbols used
  - No username in password
  - No dictionary words in password
  - Enforce minimum length > 8 characters
  - No obvious substitutions (e.g. zero for o)
  - No common passwords
- Ban credentials that have been compromised and dumped
  - https://haveibeenpwned.com/
- Recommend password managers to users

Password storage

- Employ hash stretching and a password hashing algorithm when storing passwords
  - Use hashes that are extremely slow to compute
  - Attacker obtaining hashes can't perform an efficient, offline dictionary attack to obtain weak passwords of users
- One mechanism: use a salt and iterate through a password hash algorithm multiple times
  - scrypt or bcrypt (iteration = 100ms)
  - PBKDF2([salt] + [password], c=140,000);
  - https://krebsonsecurity.com/2012/06/how-companies-can-beef-up-password-security

```
#username:realname:iterations:salt:hash

alpha:Alfred Phangiso:10000:0050B9..D970C4:63E75CA4..3AF24935
bravo:David Bravo:10000:B5916E..325460:53149EAE..7545E677
charlie:Charles Windsor:10000:49C20B..78418B:86B2D4AD..CD917089
duck:Philip Ducklin:10000:71E831..166D6A:585B8490..3D68A8E5
echo:Eric Cleese:10000:864E2A..A346B7:F8908212..C0D84C6C
```

Multi-factor authentication

- Employ out-of-band token
  - Two-factor auth via Google Authenticator, Duo
  - Yubikey authentication
  - Mobile messaging app
  - But avoid SMS (SS7)

naked security by SOPHOS 05 MAY 2017

Bank accounts raided after crooks exploit huge flaw in mobile networks

The Signalling System No. 7 (SS7) telephony signaling protocol used to establish interoperability across some 800+ service providers worldwide, is deeply vulnerable to interception by hackers, criminals, and corrupt insiders. We've known this for years. Now, in Germany, someone's used that vulnerability to raid consumers' online bank accounts.

Meanwhile, per The Register, the attackers "purchased access to a rogue telecommunications provider and set up a redirect for the victim's mobile phone number to a handset controlled by the attackers". Now, they could wait until late at night, log into the victims' online accounts, and start money transfers. As part of their SMS-based two-factor authentication (2FA) systems, the banks would dutifully send one-time mobile transaction authentication number (mTAN) numbers to their customers. These would be hijacked by the criminals, who now had the second authentication factor they needed to complete the thefts.

Multi-factor authentication

- Biometric authentication
- Use IP address and geographic location information
- Multiple, good identity questions
  - https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet
- Enforce lockout policy on multiple failures
- Employ security seals in authentication
  - To train users to detect phishing attacks

Authorization best practices

- Centralize authorization mechanism
- Minimize custom authorization code
- Authorize every request at the server
- Fail closed
  - Unexpected input causes connection termination (see PHP issues)
- Operate with Least Privilege
  - Separate user and administrator accounts
  - Run web server as a regular user
- Keep accounts unique
  - No account sharing

Session management architecture

- Keep as much information on server as possible
  - Rely upon an opaque session ID that contains no semantic content
- Never trust client or network
  - Avoid insecure client-side authorization tokens
  - Encrypt and digitally sign anything sent to client for storage that needs to come back unmodified (while keeping key to yourself)
  - Remove session information from URL (and thus, browsing history)
- Timeout sessions
  - Ensure session ID expiration
  - Verify that logoff actually destroys the session (OWASP's WebScarab)
- Ensure all session information transmitted via SSL/TLS and only via HTTPS
  - e.g. secure flag and HTTP-only flag on cookies

Labs, homework, and program

- See handout
- Session #2
  - For AJAX responses, in Chrome
    - Developer Tools:Network:XHR:<req>:Response
- Session #4
  - In python3

```
username = "%04d" % i
```

- Session #6
  - Cookie value is set both by the server (via Set-cookie:), as well as by the JavaScript code the client executes.

```
// Answer Controller
document.cookie="ac=ZG9Ob3RSZXR1cm5BbnN3ZXJz";
```

  - When solving via a Python script, JavaScript is not executed. As a result, an additional cookie parameter must be added manually to avoid the “INVALID CONTROL SET” error.

```
cookies={'ac':'ZG9Ob3RSZXR1cm5BbnN3ZXJz'}
```

Questions

- https://sayat.me/wu4f

Extra

Example: Poor CAPTCHAs

- Logic flaws

```
if params[:captcha] and params[:captcha] !=
 session[:captcha]
   # ERROR: CAPTCHA is invalid redirect
 end
 # CAPTCHA is valid
 # If no captcha is provided, the script does not fail
 safely
```

- CAPTCHA answer easily guessable
  - Single-digit sum
- Repeated answers
- OCR tools
- Improper rate-limits for incorrect guesses