
Web Attacks I

John E. Savage
Brown University
Overview

• Web security centered on session management
  – Must maintain user through sessions
• Security can be broken via client-side controls
  – Can’t rely on obfuscation, e.g. hidden info
• Loss of security token may lead to session hijack
  – Many ways to break security of tokens
  – Some forms of encrypted data can leak information
• Design of secure session management software
  is for experts, not novices.

Fundamentals of Web Security

• Fundamental truth: User (client) input is untrusted

• Goal: Provide security to authenticated users by
  – Preventing users from obtaining unauthorized access to data and functionality
  – Preventing malformed inputs from causing undesirable behavior
  – Implementing measures to frustrate attackers
  – Enabling administrators to monitor activities and configure functionality.
Handling Client Access

• To control user access to data and functionality correctly implement the following mechanisms:
  – Authentication
  – Session management
  – Access control
Authentication

• Strength of authentication must correlate with the level of security needed.
• Conventional login/password authentication may not be enough
• Once initial authentication complete, remaining accesses must also be authenticated.
  – This is the role of session management.
Session Management Overview

• Goal: manage authenticated user session
• Users make many HTTP requests via browsers
• Server handles requests from multiple users
• Methods needed to isolate user activities
• This is done via session management

• Each session must have an identifying token
• Users send tokens to servers as authentication
  – Either in URLs or via hidden form fields
• Servers create data structures for each session
Session Management Overview (Cont.)

• Cookies often used as tokens
  – Small piece of data that is stored on a user’s site
  – Needed because the HTTP protocol is stateless

• Cookies can encode sensitive information
  – Private data, e.g. passwords, purchasing history, and credit card no., can be encoded in cookies by servers

• Cookies cannot carry viruses or install malware but they can be used to invade privacy.

• They play a central role in session security
Session Security Overview

• Security of sessions is highly dependent on the security of tokens

• Majority of web attacks compromise token use

• Defects often occur in the following:
  – Token generation
  – Handling of tokens

• They help attackers find or guess others’ tokens

• Knowledge of a token facilitates session capture!
Access Control Overview

- Access control implement fine-grained logic
- For example, roles may determine access and a user may play different roles over time
- The complex nature of access control is a frequent source of security vulnerabilities
  - E.g. Web designers often assume that users will progress sequentially through pages and authenticate them only on the first page
Handling User Input

• Unfiltered user input can be dangerous
  – Too many characters can cause buffer overflow
  – HTML commands can change application behavior
• Filtering input
  – Can whitelist, black or sanitize input
  – E.g. try removing `<script>` from `<scr<script>ipt>`
• Servers may send data to users that is expected to be returned unaltered.
  – But an attacker may intercept and change the data
Handling Attackers

• Responding to errors
  – Small number may be normal, many => an attack

• Maintaining audit logs recording times and events
  – E.g. failed logins, password change, key transactions, blocked access attempts, known attack strings

• Alerting administrators
  – Usage or business anomalies, mods to hidden data

• Reacting to attacks
  – Slow down or terminate putative attacker
Bypassing Client-Side Controls

• Web applications can be difficult to write
  – Sometimes shortcuts used that jeopardize security
• Designers may assume that clients won't change data sent to them for retransmission
• But the user can be trusted!
• They can use and intercepting proxy to edit data
  – E.g. ZAP, Burp Proxy
Transmitting Data Via the Client

• Why would server send data through user?
  – Makes handling session data by server easier
  – If multiple services used on server, it may be easier to send data to the services via the client
  – Tracking a new piece of data on server may require a major rewrite. Simpler to pass it to client
Hidden Information

• Hidden form fields
  – `<input type="hidden" name="price" value = "12">`

• URL parameters
  – `http://new.net/shop/?prod=3&price=5`

• In both cases price is visible and can be changed by a hacker.
Transmission of Credentials

• If via HTTP, obviously not secure
• If via HTTPS, may still not be secure, e.g.
  – If sent as query string parameters, might be found in browser history, reverse proxy logs, etc.
  – If login request handled via redirect to new URL, could be embedded here as query string parameter
  – Credentials could be poorly encoded in cookies.
Changing Passwords

• Errors common in handling password change
  – Is user name valid?
  – Allow unrestricted number of password attempts
  – Not confirm that two copies of password match
• Password change requires complex logic
• Forgotten password functionality also fragile
  – Answers to secret questions have low entropy
  – Hints may reveal too much
  – May drop user into authenticated session directly
Attacking Session Management

• Token problem: often easy to predict
• **Attacker goal:** hijack a session!
• Many sites roll their own session management
  – Sites may need more fine-grained control
  – Want to avoid cookie-based vulnerabilities
• Common session management vulnerabilities:
  – Weaknesses in session token generation
  – Weaknesses in handling tokens through life cycle
Token Generation Weaknesses

• Tokens may encode sensitive info, e.g
  757365723d6461663b6170703d61646d696e3b646174653d30312f31322f3131
  is a hex encoding of
  user=daf;app=admin;date=10/09/11

• Polling a server will generate tokens

• Apply decoders (Base64 to Hex).

• Look for time dependence. Increments constant?

• Look for weakness in random number generator
Encrypted Tokens

- Even encrypted tokens can be employed without first decrypting them.
- Applications employ symmetric encryption.
- Some use electronic codebook ciphers (ECBs)
  - Plaintext divided into equal-sized blocks
  - Each block encrypted with the same secret key
- Patterns in plaintext recur in encrypted text
Electronic Codebook Encryption

• Message bits grouped into blocks $B_1, B_2, B_3 \ldots$ of fixed size & encrypted

• Electronic Codebook mode (ECB)
  – Encode $C_j = E_K(B_j)$, Decode $B_j = D_K(C_j)$
  – Simple, resilient, but if coding deterministic, can reveal patterns.
ECB Cipher Miused

- Consider encoding following token with UID
  \[ \text{rnd}=2458992; \text{app}=\text{iTradeEUR}_1; \text{uid}=218; \text{username}=\text{dafydd}; \text{time}=634430423694715000; \]

- Here it is encrypted:
  \[ 68BAC980742B9EF80A27CBBBC0618E3876FF3D6C6E6A7B9CB8FCA486F9E11922776F0307329140AABD223F003A8309DDB6B970C47BA2E249A0670592D74BCD07D51A3E150EFC2E69885A5C8131E4210F \]

- Since it operates on 8-byte inputs, we layout it that way.
Where Is UID? Can We Change It?

• \( \text{rnd}=2458 \) 68BAC980742B9EF8
• \( 992;\text{app}= \) 0A27CBBBC0618E38
• \( \text{iTradeEU} \) 76FF3D6C6E6A7B9C
• \( \text{R}_1;\text{uid}=\) B8FCA486F9E11922
• \( 218;\text{user} \) 776F0307329140AA
• \( \text{name}=\text{daf} \) BD223F003A8309DD
• \( \text{ydd};\text{time} \) B6B970C47BA2E249
• \( =6344304 \) A0670592D74BCD07
• \( 23694715 \) D51A3E150EFC2E69
• \( 000; \) 885A5C8131E4210F
Put Second Line after Fourth

- **rnd=2458** 68BAC980742B9EF8
- **992;app=** 0A27CBBBC0618E38
- **iTradeEU** 76FF3D6C6E6A7B9C
- **R_1;uid=** B8FCA486F9E11922
- **992;app=** 0A27CBBBC0618E38
- **218;user** 776F0307329140AA
- **name=daf** BD223F003A8309DD
- **ydd;time** B6B970C47BA2E249
- **=6344304** A0670592D74BCD07
- **23694715** D51A3E150EFC2E69
- **000;** 885A5C8131E4210F
Exploiting the Weakness

• Now send the encrypted string to server
• If server accepts it, we have learned a bit about the nature of the original string.
• The weakness is in the cipher.
Cipher Block Chaining (CBC) Encryption

- \( C_0 \) is initialization block. \( B_1, B_2, B_3, \ldots \) are data blocks
- Encode \( C_1 = E_K(B_1 \oplus C_0), C_2 = E_K(B_2 \oplus C_1), \) etc.
- Decode \( B_j = D_K(C_j) \oplus C_{j-1} \)

- If \( C_j \) lost, can decrypt all but \( j \)th and \((j+1)\)st block.
- But weakness in EBC is overcome
Other Weaknesses in Session Tokens

• Tokens must be handled securely
  – Don’t start with HTTPS and then use HTTP while transmitting tokens
  – Using the back button may invoke HTTP

• Tokens can be disclosed in logs
  – Trick an operator in disclosing them to you

• More than one token assigned to a session
  – May reveal a security compromise

• Sessions should be short and timed-out
Review

• Web security centered on session management
  – Must maintain user through sessions
• Security can be broken via client-side controls
  – Can’t rely on obfuscation, e.g. hidden info
• Loss of security token may lead to session hijack
  – Many ways to break security of tokens
  – Some forms of encrypted data can leak information
• Design of secure session management software is for experts, not novices.