Cookiejar Kintsugi

Reviewing the state of web application session security

Julian Lobbes
TU Braunschweig

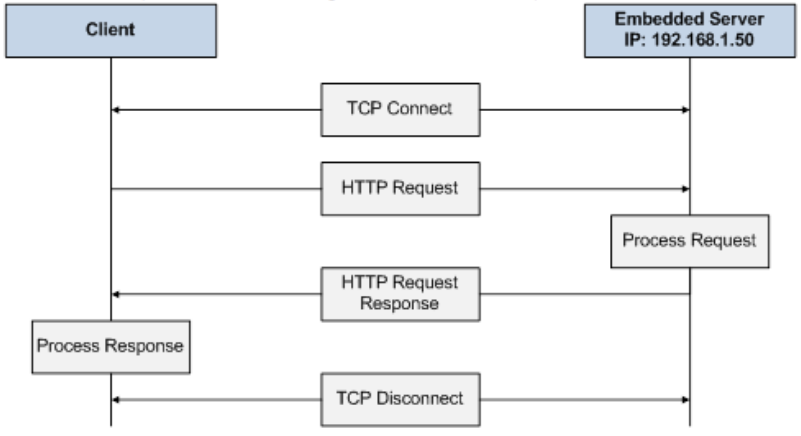
Introduction

- Web sessions
  - what are they and why are they important?
  - vulnerabilities and attacks
  - threat mitigation
  - the current state of the web

Background

HTTP

- *Hyper Text Transfer Protocol*
- client ⇄ server
  - client requests (GET, POST, …)
  - server responds
- **cleartext** messages
  - metadata in *Header*
  - content in *Body*
- **stateless** communication



HTTPS

- *Hypertext Transfer Protocol Secure*
  - **encrypts** messages
  - **authenticates** the **server**
  - public certificate infrastructure

HSTS

- *HTTP Strict Transport Security*
- server declares that **only** HTTPS connections are accepted
  - in response to client's initial HTTP request
- HSTS **preloading**:
  - list of HSTS-enabled domains hardcoded in the web browser
  - initial request is encrypted as well

Web sessions

- web applications need stateful communication
- **session:**
  - identify/authenticate a client across requests
  - using a *session identifier* (**SID**)
  - SID assigned to client by the server

Session Management

Session Management

- different methods used to transmit/store SIDs:
  - hidden form fields
  - URL rewriting
  - **cookies**

Hidden form fields

- SID is embedded inside HTML source code (*<form>* element)
  - client submits form with every request (*POST*)
  - server responds with an HTML document containing the SID
- problems:
  - usage of back-button
  - performance costs (parsing, chaching)
  - vulnerable to XSS

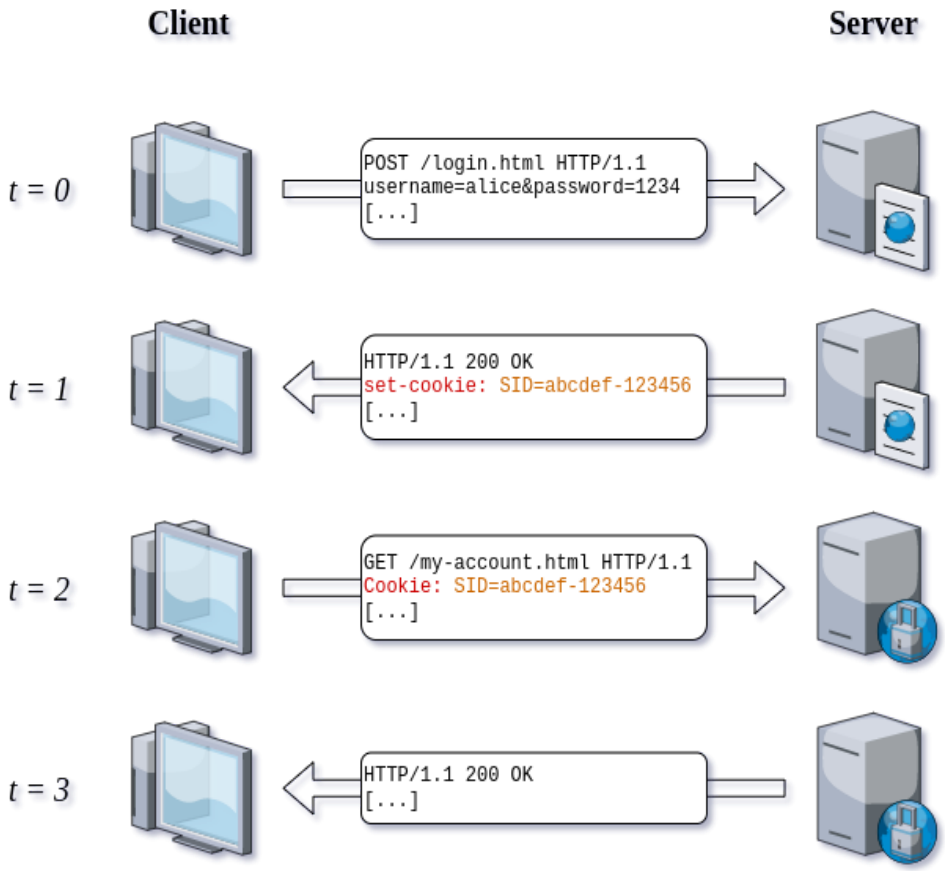
URL rewriting

- SID is stored as a URL parameter:
  - *https://example.com/profile.html?**sid=a92nl52***
  - client appends SID to URL with each request
  - server redirects to URL containing SID with each response
- problems:
  - usage of back-button
  - performance costs (caching)
  - SID leakage to third parties
  - vulnerable to session fixation attacks

Session Cookies

- Cookie:
  - key/value-pairs in HTTP Header
  - server sends a cookie using *Set-Cookie* directive
  - client stores cookies locally per domain
    - appends all stored cookies with each request to domain
  - special attributes:
    - ***Secure*** prevents transmission in plain text
    - ***HttpOnly*** prevents access by client-side scripts
- problems:

Session Cookies



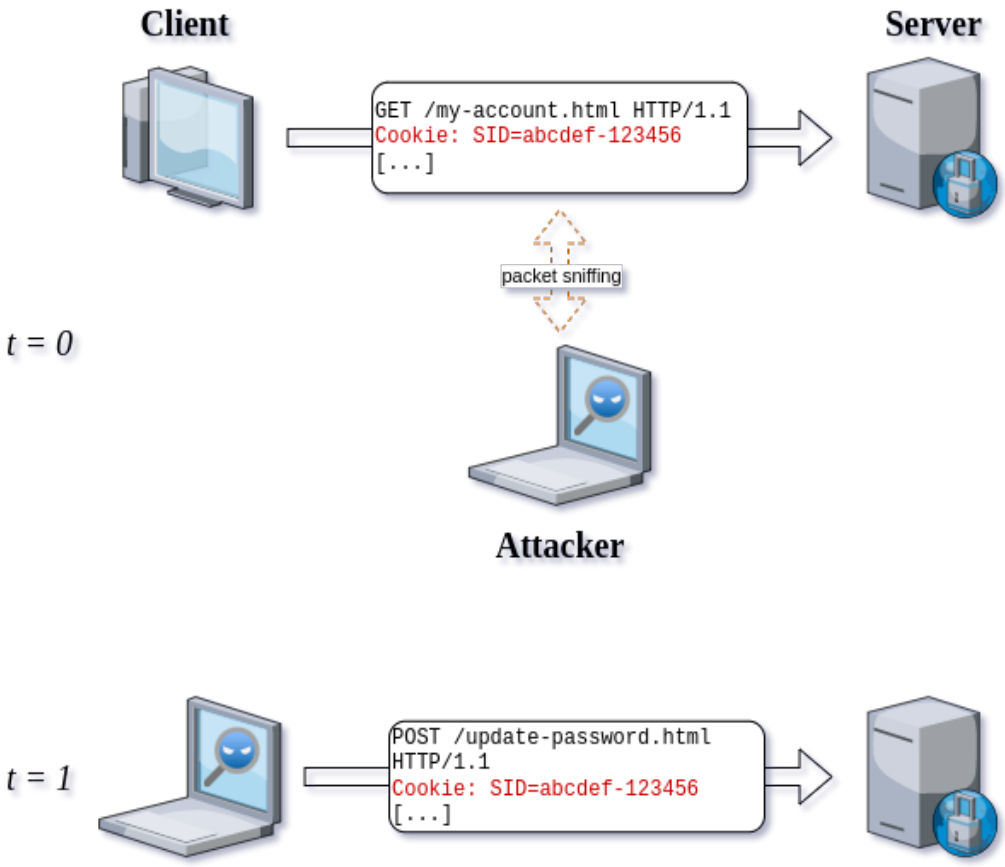
Session Security Threats

Session hijacking

- *Session Hijacking*:
  - attacker intercepts SID and impersonates victim
- SIDs are a valuable target

Man in the middle

- network-layer attack
  - packet sniffing
- exploits unencrypted traffic

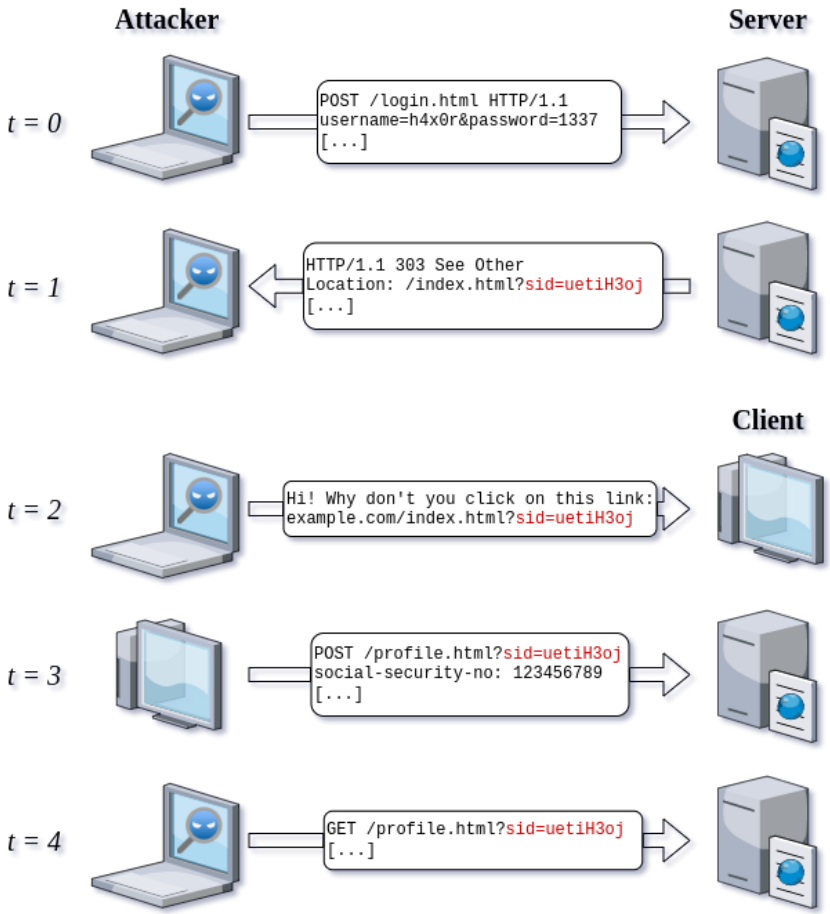


Cross-site scripting (XSS)

- Client-side scripts can access HTML elements
  - unless isolated
- cookies only protected if *HttpOnly* is set
- malicious script sources:
  - attacker injecting script into client browser
  - third-party imported scripts

Session fixation

- attacker sets up a session
- introduces SID into victim's browser



Cross-site request forgery (CSRF)

- **authenticated** victim unknowingly takes an action chosen by the attacker
  - client browser attaches *bank.com* session cookies with every request
  - attacker coerces client browser to make a request

```
<img src="https://bank.com/transfer.php?acct=ATTACKER&amount=100000"
width="0" height="0" border="0">
```

Threat Mitigation

Threat mitigation

- use cookies
- encrypt **all** traffic **always** (HTTPS)
  - set cookies to *Secure*
  - use HSTS with preloading
- prevent script access to session cookies
  - set cookies to *HttpOnly*
  - isolate scripts

Analysis & Prevalence

Analysis

- steps required to audit session security:
  - create a user account
  - sign in
  - identify session cookies
  - find vulnerabilities
  - (assess the damage)
- large-scale studies are difficult!
  - lack of empirical data

Data sources

- OWASP Top Ten
  - public contributions
  - large sample size
  - only qualitative analysis
- Calzavara et al. (2019)
  - less than 23% of websites using session cookies set *HttpOnly*
  - partially automated, n=20
- Sivakorn et al. (2016)
  - 15 major websites expose session cookies in cleartext
  - manual, n=25

Data Sources

- Drakonakis et al. (2020)
  - *Cookie Hunter: Automated Black-box Auditing for Web Authentication and Authorization Flaws*
  - fully automated
  - n=25,000
  - black-box web application session auditing framework

Cookie Hunter: Account Creation

- crawl websites for sign-up and registration forms
  - look for *<input>* fields
  - identify their purpose
    - based on their labels
    - regex detection
    - use of translators

Cookie Hunter: Account Creation

- fill sign-up forms with dummy data and submit
- determine if registration was successful:
  - by trying to log in

Cookie Hunter: Account Login

- Logging in:
  - submit credentials in login form
  - determine success using a login oracle:
    - presence of a *logout* button indicates success

Cookie Hunter: Account Login

- some sites support single-sign-on (SSO)
  - used if direct sign-up did not work
  - using a Google/Facebook account
- sites using CAPTCHA challenges prevent automated sign-up

Cookie Hunter: Audit

- identify session cookies
  - based on trial and error and login oracle
- inspect cookie attributes (*HttpOnly*, *Secure*)
  - is HTTPS/HSTS employed correctly?
  - are there unisolated 3rd party scripts?

Cookie Hunter: Damage Assessment

- privacy auditor on vulnerable websites
  - determine what kind of data can be exfiltrated by an attacker
    - emails
    - addresses
    - phone numbers
    - …

Cookie Hunter: Findings

- 1.5 million domains inspected
  - 200,000 of which support account creation
- 25,000 were fully audited by the framework
  - 12,014 (48.43%) vulnerable to network sniffing
    - 10,495 of which do not deploy HSTS correctly
  - 5,099 did not set *HttpOnly* while importing 3rd party scripts
  - most vulnerable sites leak a lot of sensitive information
    - full addresses
    - phone numbers
    - credit card numbers
    - weakly-hashed passwords

Conclusion

Session hijacking

- Many websites are vulnerable
  - network sniffing
  - XSS through imported scripts
- Causes:
  - insufficient/improper configuration of HSTS
  - unisolated scripts
- Analysis:
  - manual analysis only allows small sample sizes
  - The Cookiehunter successfully implements the first automated black-box auditing framework

Questions