

INFO-6076

Web Security

Sessions, Cookies & Authentication

Agenda

- Sessions
- Cookies
- Session Hijacking
- Session Security
- Authentication Management
- Testing Authentication
- Securing Authentication
- Lab 08 Overview

Sessions

Sessions

Why Do We Need Sessions?

A session refers to all the traffic between a web browser and a web server from the time the user first browses to the web application to the time the user is logged out

- Also known as a "Single Sitting"

HTTP is a stateless protocol

- No reason to store user data between requests
 - Fine for static pages
- To manage sessions the web server application creates an id number to track the HTTP packets related to a particular user
 - Session ID

Sessions & Authorization

Sessions allow the user to login while interacting with a web application

- The internet would be entirely impractical without sessions
 - Especially with interactive web applications

Session Management areas:

- Identifying and keeping track of the user
- Controlling what the user is authorized to do
- Keeping track of what the user has done

Session Tracking

Common methods used to track sessions:

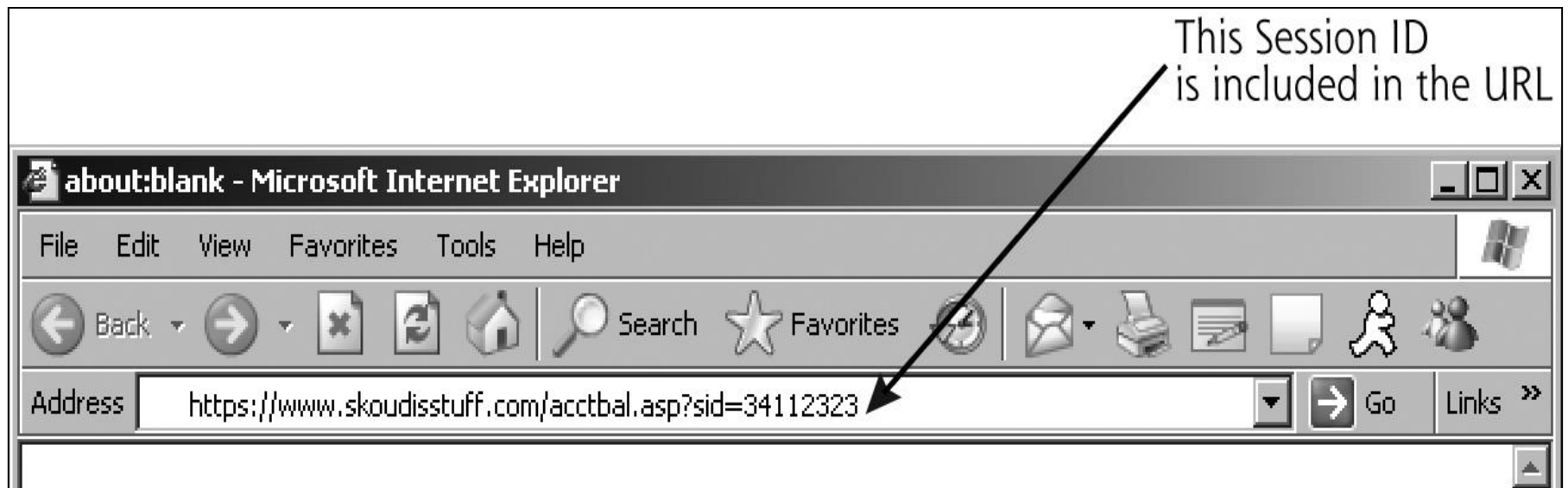
- Cookies
- URL
- Hidden Forms
- IP Address
- Web Storage
- LSOs and Isolated Storage

Session tracking with cookies is the most used client-side method

Session Tracking

URL

- The session ID is written in the browsers address bar
- The session ID is passed as a parameter in the http get request for all subsequent queries to the web



Session Tracking

Problems with URL parameters

- User / Attacker has access to, and can modify, the values
- There is no defined length limit for a URL, in the HTTP protocol, but many browsers, web applications and servers impose limits
 - Reduces the amount of state information that can be maintained

Session Tracking

Hidden Forms

- Session ID is written into hidden form elements
- Hidden elements are not displayed by the browser
 - Can be viewed in HTML page source, but are not normally displayed to the user

```
<input type="hidden" name="Session" value="12345">
```

- **Type** is hidden
- **Name** is Session
- **Value** is 12345



As discussed previously, the input type **hidden** is not normally visible to the end user through a browser window

Session Tracking

IP Address

- Tracking a user based on IP address
- All users behind a NAT address would appear as the same user

Web Storage

- Sometimes called HTML5 Local Storage
- Provides persistent client-side storage
- Attempt to improve on cookies, but the information is still stored client-side
- Information doesn't need to be sent with each request

http://www.w3schools.com/html/html5_webstorage.asp

Session Tracking

Local Shared Objects (LSOs)

- Used by Adobe Flash
- Sometimes referred to as Flash Cookies

Isolated Storage

- Used by Silverlight
- Can provide either client-side or server-side storage

Cookies

- Most widely used method for tracking session IDs
- Small plain text files stored on the user's computer
- Contains no executable code
- limitation (300 cookies, 4096 bytes per cookie, etc.)

Session Tracking

Request	Response
Raw	Headers

```
GET / HTTP/1.1
Host: cbc.ca
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Request	Response
Raw	Headers

```
HTTP/1.1 301 Moved Permanently
Server: AkamaiGHost
Content-Length: 0
Location: http://www.cbc.ca/
Cache-Control: private, max-age=60
Expires: Wed, 22 Aug 2018 21:44:58 GMT
Date: Wed, 22 Aug 2018 21:43:58 GMT
Connection: close
Set-Cookie: akaas_feed=2147483647~rv=87~id=aeb327912de8263b8635e122f87b9ea8; path=/
```

First Request [cbc.ca](http://www.cbc.ca)
(no cookie file yet)

Session Tracking

Request Response

Raw Params Headers Hex

GET /g/stats/js/cbc-stats-top.js HTTP/1.1
Host: www.cbc.ca
User-Agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
Accept: */*
Referer: http://www.cbc.ca/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: akacd_akcx=2177452799~rv=20~id=17feb7bc7c7272f07400e9ac7455019a; akaas_feed=2147483647~rv=15~id=3942b7a75a2dfa8ae817cc736b47542b
Connection: close

Request Response

Raw Headers Hex

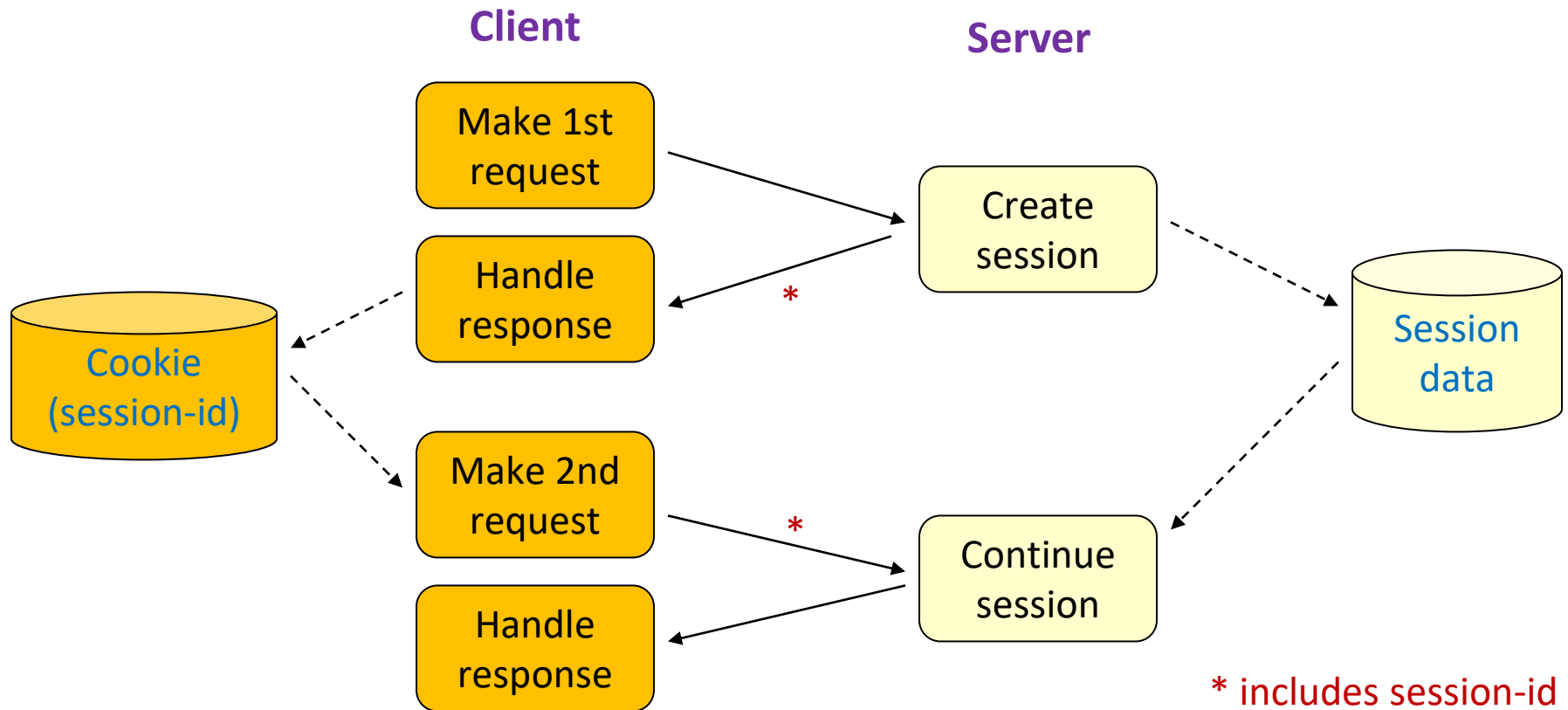
HTTP/1.1 200 OK
Server: Apache/2.2.15 (Red Hat)
Content-Type: text/javascript
Access-Control-Allow-Origin: *
Server-Timing: edge; dur=0
Server-Timing: cdn-cache; desc=HIT
Vary: Accept-Encoding
X-Origin-Server: static01_cache04
Cache-Control: max-age=27
Date: Wed, 22 Aug 2018 21:44:07 GMT
Connection: close
Connection: Transfer-Encoding
Content-Length: 139624

Second Request cbc.ca
(cookie file is set)

"use strict";var _typeof="function"==typeof Symbol&&"symbol"==typeof Symbol.prototype.constructor===Symbol&&!t={i:t.lib={},n:function(){}}r=i.Base.extend({function

HTTP Session Management

HTTP Session is a series of HTTP transactions between a client and a server. **Cookies** are typically used to identify sessions.

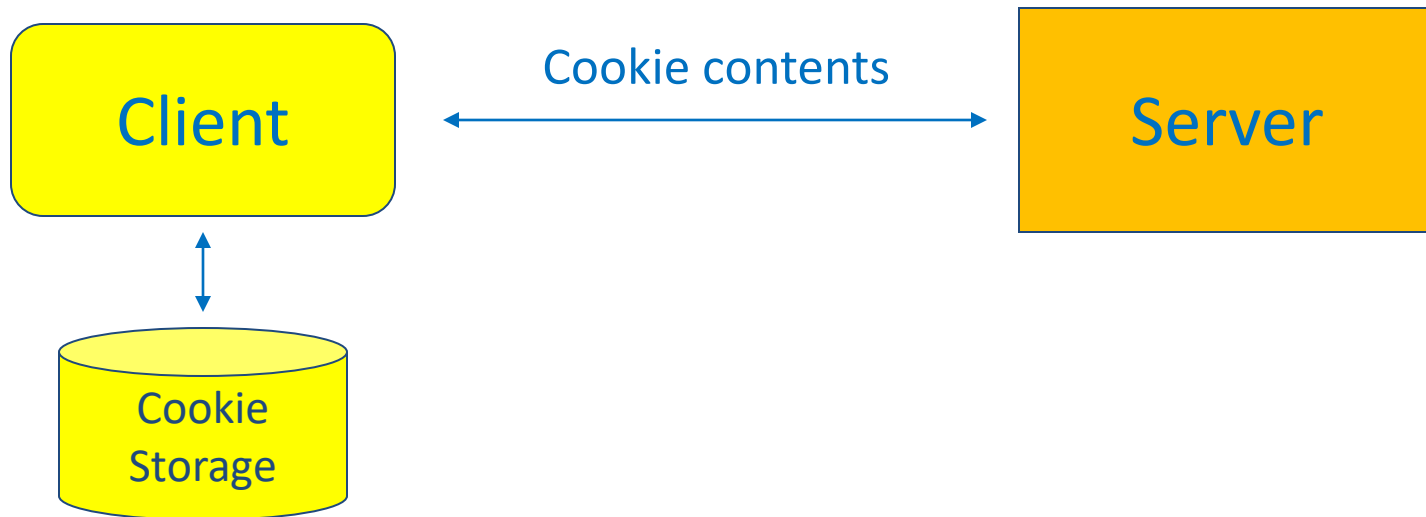


Cookies

HTTP Cookie Management

HTTP cookie is a small file that is

- Provided by the server as an **HTTP response header**
- Stored by the client
- Returned to the server as an **HTTP request header**



Cookies

Temporary / Per-Session cookies

- Stored in memory
- Expire when client closes browser

Permanent / Persistent cookies

- Written to disk
- Store long term user preferences
- Expire when they reach their expiration date

Secure

- Has the secure attribute and is only used via HTTPS
- Ensures cookie is encrypted in transit

Cookies

HttpOnly Cookie Flag

- Restricts session cookie access to HTTP or HTTPS
 - No access from non-HTTP APIs (scripting languages)
- Helps to mitigate the threat of session cookie theft via cross site scripting

Third Party Cookie

- Set with a different domain than the domain or subdomain shown in the address bar
- Often used by advertising sites to track users across many sites

Cookies

Supercookie

- In older browsers you could set a cookie that applied to a top level domain (TLD)
 - .com .ca .us
- This would allow you to set a cookie that would apply to the entire TLD and its subdomains

Zombie Cookie

- Refers to cookies that are regenerated despite being deleted by the user
 - Flash cookies are known to use this technique
 - Script is used to pull cookie content from storage

Cookies

Primary Uses of Cookies

■ **Session Management**

- Shopping cart contents
- Authentication status

■ **Personalization**

- User preferences
- User specific content

■ **Tracking**

- What pages a user has visited
- What sequence they visited pages in
- How long they viewed each page

Session IDs

The primary security concern when dealing with session IDs is that the **ID might be stolen**

- Allowing the attacker to impersonate a user and act on their behalf

Poorly protected session IDs can be altered by the hacker and resubmitted by the browser

- Hackers can use a proxy or cookie editor

In addition to protecting your Session IDs, they shouldn't be easy to predict

- They need to have good entropy
- Long enough and with different characters

Session IDs

Characteristics of good Session IDs:

- Truly random
- Have large enough range for user base
- Numbers should not be repeated
 - Otherwise high traffic situations could result in session ID repetition (Session conflict)

Remember: if a Session ID can be predicted it will be much easier for an attacker to steal a user's session

Session ID Quality

You can use tools like the **Burp Suite Sequencer** or **WebScarab** to analyze the true randomness of a Session ID

- Burp Sequencer gives you a better indication of randomness
- WebScarab provides a more visually appealing output
- You need to keep your audience in mind

Session Management

To increase usability, once a user has authenticated to the web server they should not be asked for account and password information again

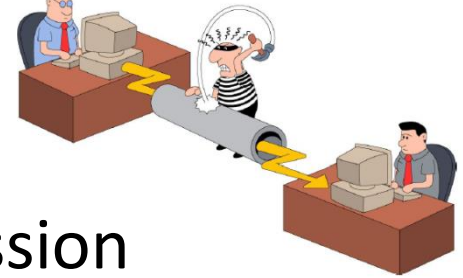
- The session ID is used to identify the user submitting the request, or data, to the web server throughout the session
- Session ID management is transparent to the user

Session Hijacking

Session Hijacking

Session hijacking is the process of taking over an established session

A Session Hijack specifically refers to the moment when an attacker uses a stolen session token in order to impersonate a user/application



Takes advantage of the fact that there is already an established session between the client and server

- Authentication has already taken place

Session Hijacking

The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to restricted area (Web Server, Databases, etc.)

The session token could be compromised in different ways

The most common are:

- Predictable / Easy to crack session token
- Session Sniffing
- Man-in-the-middle attack
- Man-in-the-browser attack
- Client-side attacks (XSS, malicious JavaScript codes, Trojans, etc.)

Session Hijacking Steps

Typical Session Hijacking Attack:

Find a victim

- Network scanner, sniffers, etc.

Get inline, or between the victim and server

- Session Sniffing
- Man-in-the-Middle
- Man-in-the-Browser

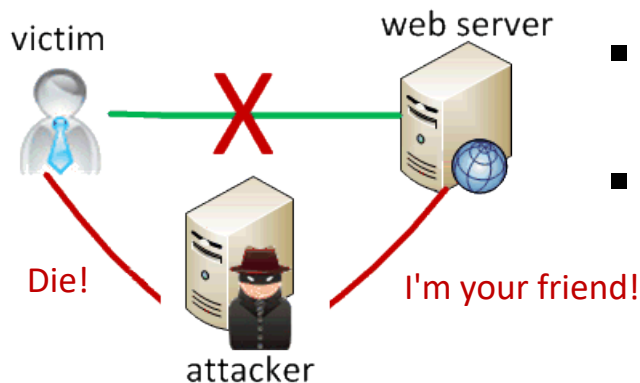
Monitor the traffic between the victim and server

- Used to determine the protocol being used, or web application being accessed
- Must be susceptible to session hijacking

Session Hijacking Goal

When an attacker hijacks the session, they can present themselves to the server as if they are the victim

The attacker will often use a DoS attack against the victim when they steal the session



- Prevents the victim from accessing the server at the same time as the hacker
- Users will often assume they simply lost network connection

Session Hijacking Levels

Network Level Hijacking

Involves the interception of packets between the victim and server

- ARP poisoning
- DNS Spoofing
- Guessing the next sequence number in the TCP/IP conversation
- In most cases attacker must be on the same network as the victim

Application-Level Hijacking

Involves gaining control of the HTTP session by obtaining the session ID

- Guessing / Brute forcing session IDs
- Sniffing unencrypted traffic for session IDs
- *Using HTML injection or Cross Site Scripting to get the victim to divulge the session data*
- Compromising the victim machine and pulling the Session ID off the computer

Session Hijacking Types

Two main types of session hijacking attacks

Active

The attacker finds a session between the victim and server and takes it over

Passive


The attacker gets between the victim and server, but simply records the traffic for later use

An active hijacking attack usually begins with a passive hijacking attack

Session Prediction / Cracking Attack

This attack focuses on predicting / cracking / brute forcing session ID values. An attacker can bypass the authentication schema of an application.

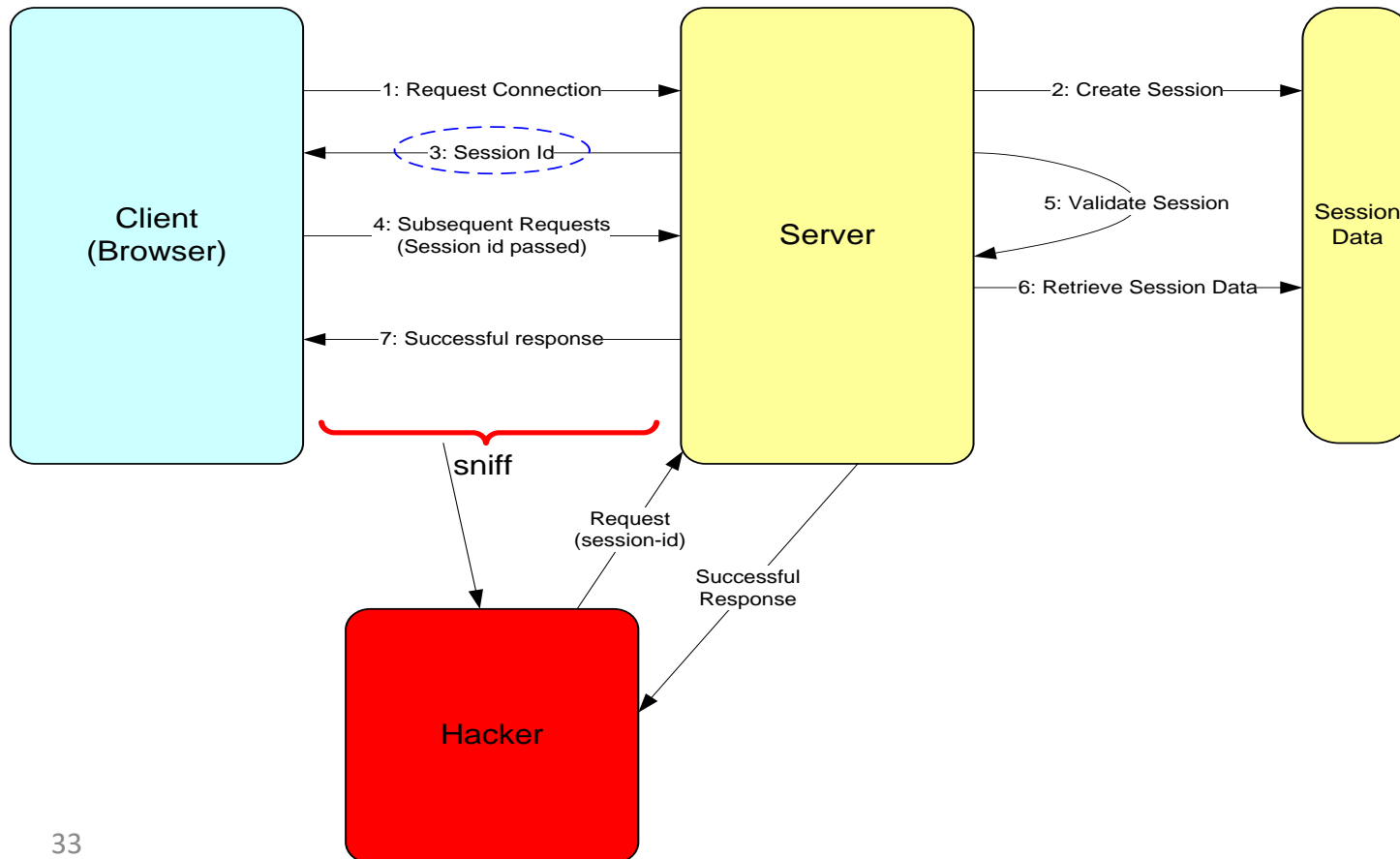
```
GET http://janaina:8180/WebGoat/attack?Screen=17&menu=410 HTTP/1.1
Host: janaina:8180
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://janaina:8180/WebGoat/attack?Screen=17&menu=410
Cookie: JSESSIONID=user01
Authorization: Basic Z3Vic3Q6Z3Vic3Q=
```



Example: The session ID variable is represented by JSESSIONID and its value is "user01", which corresponds to the username. By trying new values for it, like "user02", it could be possible to get inside the application without prior authentication.

Session Sniffing Attack

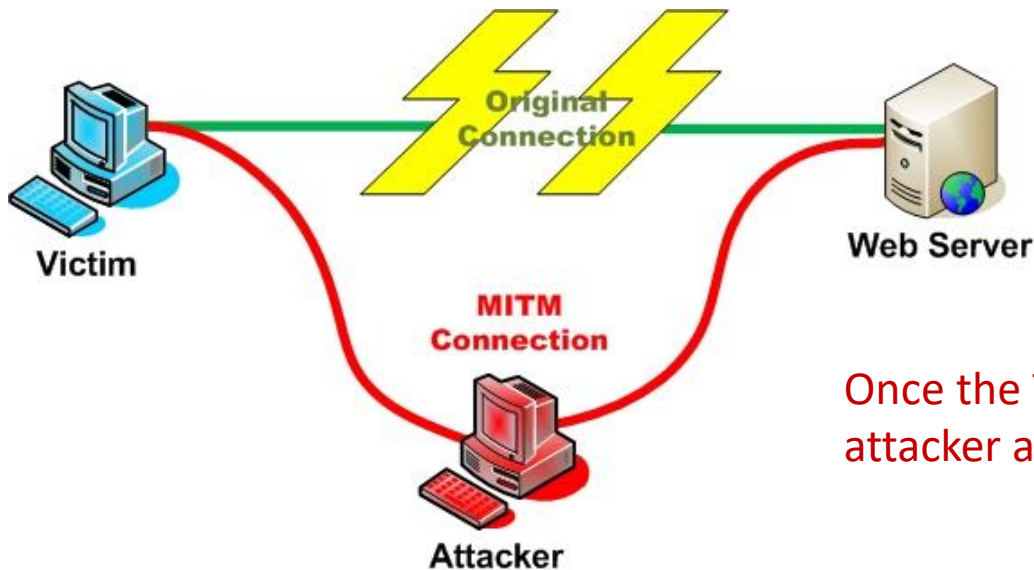
The attacker uses a sniffer to capture a valid session token (session-id), then he/she uses the valid token session to gain unauthorized access to the Web Server.



Man-in-the-Middle Attack (MITM)

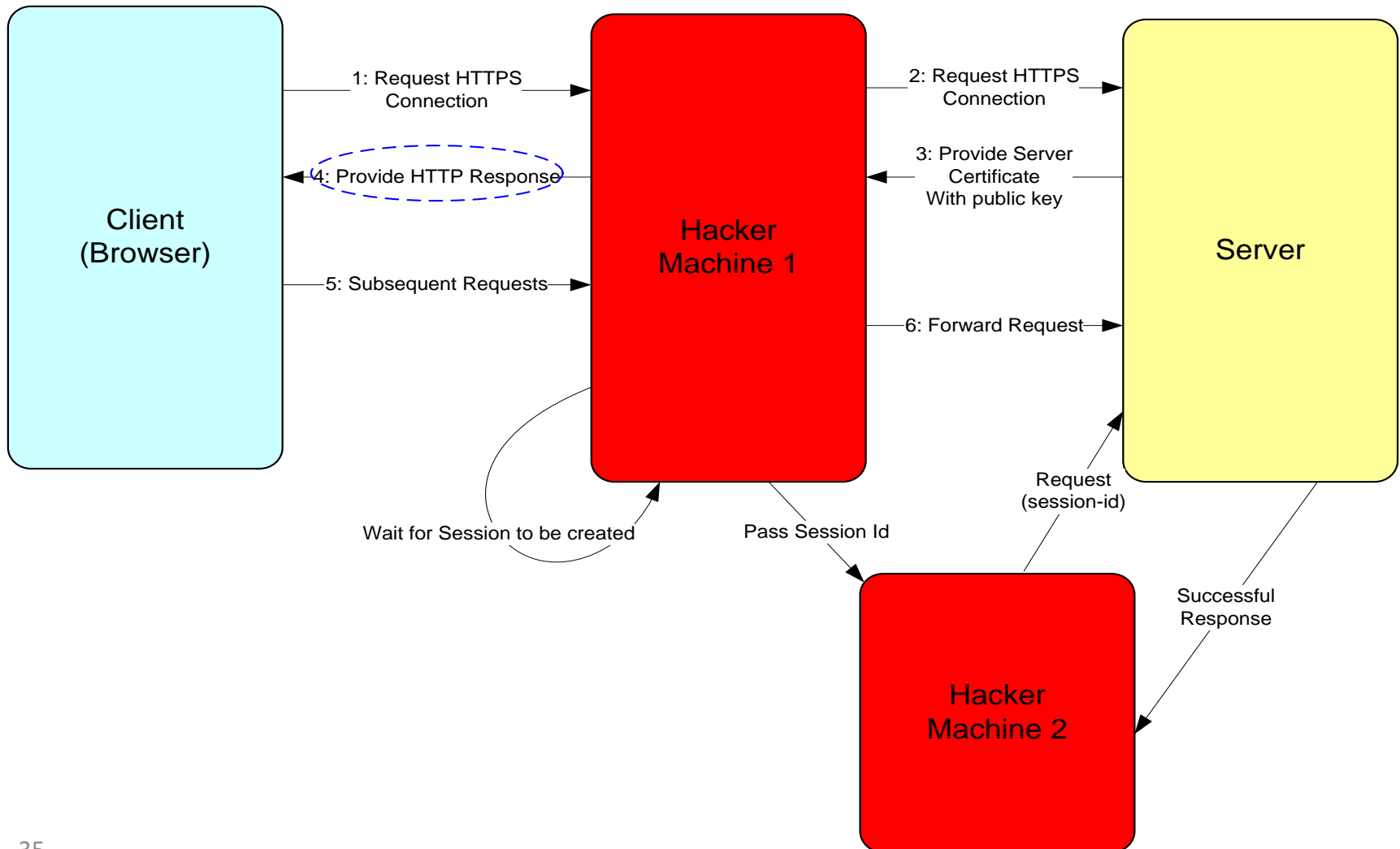
The man-in-the middle attack intercepts a communication between two systems

- Involves the attacker placing themselves between the victim and server
- Data can be intercepted, modified and retransmitted between the source and destination



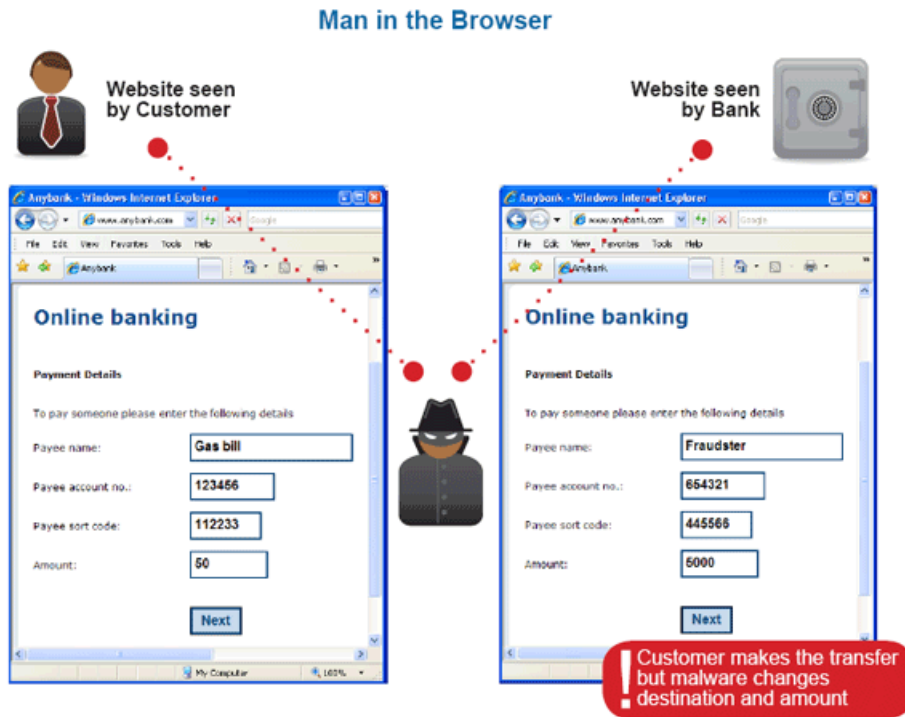
Once the TCP connection is intercepted, the attacker acts as a proxy

MITM Advanced Attack



Man-in-the-Browser Attack (MITB)

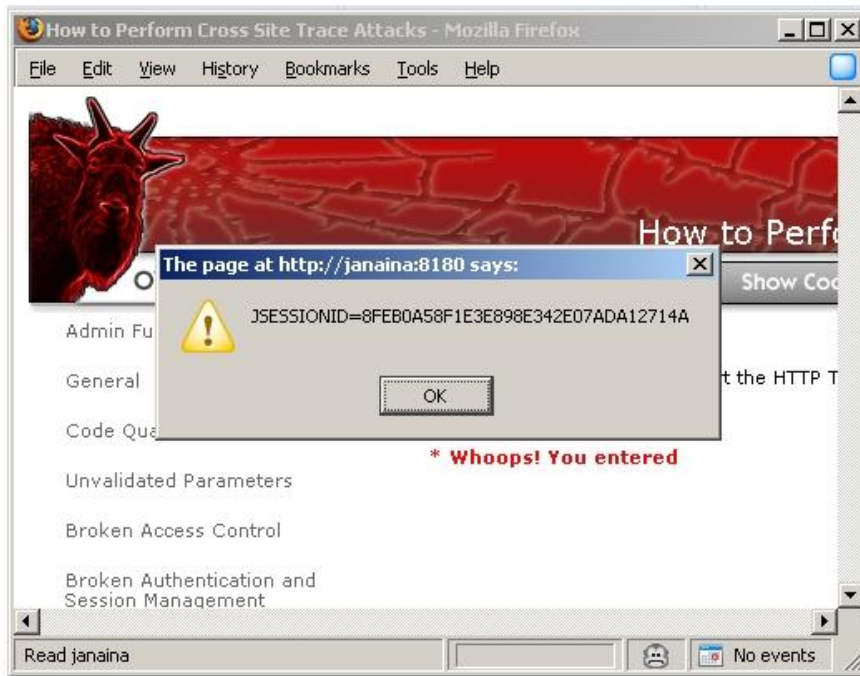
Same approach as MITM, but relies on malware / Trojan horses to intercept and manipulate calls between the browser and the browser's security mechanism, sniffing or modifying transactions as they are formed on the browser, but still displaying back the user's intended transaction



The most common objective of this attack is to cause financial fraud by manipulating transactions of Internet Banking systems, even when other authentication factors are in use.

Cross-Site Scripting Session Attack

The attacker can compromise the session token by using malicious code or programs running at the client-side.



Output of
`<SCRIPT>alert(document.cookie);</SCRIPT>`
includes the session token

It's possible to create specific JavaScript code that will send the cookie info to the attacker

Example: An attacker sends a crafted link to the victim with the malicious JavaScript, when the victim clicks on the link, the JavaScript will run and complete the instructions made by the attacker.

MITM and MITB Uses

Take over the session (session hijacking)

- User might know (if DoS is used), but server won't

Gather information for later use

- If poor session expiration management is used

Gathering information for concurrent use

- If concurrent sessions are allowed

Simply Gather Information

- Steal the user's logon credentials
- Steal other information about the client

More Terminology

The following attacks are often discussed alongside Session Hijacking:

- Session Spoofing
- Session Replay
- Session Sidejacking
- Session Fixation

Session Hijacking vs. Session Spoofing

Session Spoofing is like **Session Hijacking** except that the attacker doesn't actively take the victim offline

- With Session Hijacking the victim, attacker and server all need to be online at the same time
 - The attacker needs the victim to be online so they can take over the session
- With session spoofing only the attacker and server need to be online at the same time
 - The attacker pretends to be victim while the victim is offline

Session Hijacking vs. Session Replay

Both are forms of Man-in-the-Middle attacks

- With session replay the attacker captures packets then sends them to the server at a later time
 - An attacker could capture a valid user's authentication process
 - After the session is over, the attacker could replay the captured authentication process to initiate another session

Session Sidejacking

Takes advantage of sites that use SSL encryption (HTTPS) for login pages, but revert to unencrypted HTTP after authentication

- Attacker can't see the initial authentication, but can see everything else sent between the client and server
- The attacker can now steal the cookie being passed between the server and client, as this often isn't part of the authentication process and is done over HTTP

Session Fixation

Takes advantage of web applications that initially use anonymous sessions / tokens

- Attacker needs to get token and pass it to the victim
- After authentication the same session is upgraded to an authenticated session



In these attacks the session ID isn't changed when the authentication state changes!

Session Security

Expiration

- Session ID expiration is used to define when the session is over
- This is a difficult problem as the server only sees page requests from the user
 - Doesn't necessarily see when the user closes the web page
- Expiration ensures that, if a cookie is acquired by a hacker it will be of limited use/value
 - Duration of client session
 - Accessing limited information

Expiration

- There are a number of ways expiration can be handled:
 - Browser Close
 - Fixed Time After Login
 - Fixed Time After New Request
 - Never
 - Authentication on Action
- You can see some of these settings with the **Edit Cookie** add-on for Firefox
 - You will be using this in the lab

Expiration

■ Browser Close

- The session stays open until the user closes the browser
- Even if they immediately open the browser again they will be required to log in

■ Fixed Time After Login (Absolute Timeout)

- A session duration is set (e.g. 1 hour)
- If the user logs in at 11am, then closes their browser, opens it, and navigates to the site again before 12pm, they will still be logged in
- If they waited until after 12pm then would be required to log in again

Expiration

- **Fixed Time After New Request (Idle Timeout)**
 - As with fixed time after login, you set a duration for which the session is valid
 - Unlike fixed time after login, every time the user makes a new page request the timer is reset
 - A benefit of this method is that you can reduce the expiration time
 - Reducing the period of time a stolen session ID is useful for
 - A downside to this method is that many AJAX applications continually poll the server
 - Basically become Browser Close method

Expiration

- **Never**

- Once a user is logged on they could close the browser and open it three years later and still be logged on
- Only used when security is not a concern

- **Authentication on Action**

- Regardless of when a user logged on, if they are going to perform a **high risk action**, they will need to authenticate again
- Need to pay special attention to your definitions regarding high risk actions

Expiration

- Session ID Expiration Methods can be combined
 - Fixed Time After Login
 - Browser Close
- You get the benefits of both
- Session ID Expiration attempts to reduce the likelihood of a stolen session ID being useful to an attacker
- As with all security it is a balance between usability and security

Cookie Security

- Limit third party cookies
- Set browser to delete temporary internet files when the browser closes
- Take the time to look at what cookies are being set on your machines
- All the major browsers allow you to do this

Session ID Defense

- **Enforce Absolute Session Timeouts**
 - If a session ID never expired it would be a permanent key to your web application
- **Enforce Idle Session Timeouts**
 - If the user is inactive for a set period of time
- **Limit Concurrent Sessions**
 - Prevents an attacker from using the stolen session ID while the user is also using it
- **Mandate Secure Cookies**
- **Use the HttpOnly Flag**

Session ID Defense

- Use Cryptographically Random Session IDs
 - Prevents guessing of session IDs
- Destroy Invalidated Session IDs
 - Once the session ID has expired get rid of it
- Use Encrypted Cookies
 - If you need to store state information on the client encrypt it
- Logging Out
 - The best way to confirm that a session is over is to have the user log out
 - Make it easy

Session Hijacking Prevention

General Recommendations

- Educating the users
 - Paying attention to https vs. non-https, properly signing out, not clicking on links
- Using high entropy in session token generation
 - Higher the entropy more difficult to predict
- Timing out and/or re-generating sessions
 - reduce window of vulnerability
- Using SSL for all communications
 - difficult to sniff
- Forcing Re-authentication or step-up authentication
 - limit damage if session is hijacked
- Encrypting session and cookies data
 - prevents session spoofing
- Using Context data for validating session-ids / Secondary checks for Sessions
 - make it difficult to use a hijacked id
- Input validation
 - prevent XSS and other vulnerabilities

Session Hijacking Prevention

Protection from Sidejacking

- Using SSL for all communications
 - Client can use tools such as HTTPS-Everywhere, force the sites to use HTTPS protocol, etc.
 - Make sure you are only connecting to WPA2 encrypted Access points
 - Use VPN connection
- Encrypting session and cookies data

Protection from Session Fixation

- Forcing Re-authentication or step-up authentication
 - Issue a new token when the state goes from unauthenticated to authenticated or from unidentified to identified.
- Using Context data for validating session-ids / Secondary checks for Sessions
 - Use per-page tokens in addition to the main session token

MITM Tools

MITM Attack Tools

- PacketCreator
- Ettercap
- Dsniff
- Cain & Abel

MITM Proxy-only Tools

- OWASP WebScarab
- Paros Proxy
- Burp Proxy
- ProxyFuzz
- Odysseus Proxy
- Fiddler (by Microsoft)

Authentication Management

Authentication Management

- Authentication is how web applications determine who gets what access to data
- It is the front line defense against malicious users
- In theory, it is simple, but in reality it can cause problems
- If a malicious user can bypass authentication systems, they can potentially obtain unrestricted access to the application and data stored within it

Authentication Management

- There are numerous authentication methods available:
 - HTML authentication
 - SSL certificates
 - Multifactor authentication systems with passwords and tokens
 - Integrated authentication using NTLM or Kerberos
- Most web applications use HTML to retrieve a username and password

Authentication Management

- Despite the authentication systems employed by web applications, attacks can focus on the client side
 - Phishing
 - Client side Trojans
- Intranet users typically have integrated systems in place where a user provides their domain credentials to get access to network resources

Authentication Management

- Password requirements are often insufficient to provide security
 - Most web applications do not require lengthy passwords and users are often careless with their passwords
 - The longer the password, the more difficult it will be to brute force

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15 bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	6tn years	100 tn years	7qd years

**TIME IT TAKES
A HACKER TO
BRUTE FORCE
YOUR
PASSWORD**

security.org/how-secure-is-my-password/



How Secure Is My Password?

✓ The #1 Password Strength Tool. Trusted and used by millions.



It would take a computer about

15 billion years

to crack your password

Let's take a look at some of the password requirements from the banks...

RBC Password Requirements

← → ↻ 🔒 https://www.rbcroyalbank.com/onlinebanking/remember_my_card/about.html



Royal Bank

FAQ about signing in

1. [Why isn't my Client Card number, username or password working?](#)
2. [How do I reset the password for my personal account?](#)
3. [How do I reset the password for my business account?](#)
4. [When I try to reset my password, I get an error on Step 1 – Verify Your Identity that says, "Please correct the informat](#)
5. [How do I sign in if I haven't enrolled in Online Banking yet?](#)
6. [How does the Remember Me feature work?](#)
7. [Why am I seeing a "Page Cannot Display" message?](#)

Why isn't my Client Card number, username or password working?

Here are some suggestions that might help:

- Make sure to enter your 16-digit Client Card number with no spaces.
- If you've created a username, use it to sign in instead of your Client Card number.
- **Your password would have 8 to 32 letters and numbers.**
- If you're signing in using a mobile device with a small screen, it might be more difficult to see what you've typed. Try sig
- If you're signing in to a joint account, use your own Client Card number and password.
- If you have more than one Client Card, make sure to enter the Client Card number for the account you want to access.

CIBC Password Requirements

Privacy and Security

Debit and Credit Card Fraud Alerts

Online Government Authentication

Digital Banking Guarantee

Banking Fraud

Online Banking Safety Tips

Privacy Policies

Mobile Banking Security

Browser Requirements

Digital Privacy Statement

Digital Privacy Statement

Email and Text Message Fraud

Debit and Credit Card Fraud

Identity Fraud and Theft

Other Common Fraud

Spyware and Removal

Free Anti-virus Software Trial

Ad Preferences

Verified.Me

Privacy and Security ► Online Banking Safety Tips

Online Banking Safety Tips

Here are some simple tips that you can follow to ensure that your online banking experience is safe and hassle free.

Keep your passwords, Personal Identification Number (PIN) and card numbers confidential

Do not share your CIBC Online Banking password or bank machine Personal Identification Number (PIN) with anyone. Giving your password or PIN to another person or company places your finances and privacy at risk.

Change your password regularly and use a different password than you use for other websites. Make it difficult for others to guess your password by using a combination of letters and numbers in your password. If you think someone knows your password, change it right away.

Never share, disclose, or provide your card number or password to another party or website other than CIBC. CIBC will never send you an e-mail requesting this information.

Changing your password

Don't use a public computer (for example, in a library or internet café) to change your password. Don't save your card number or password on a public computer.

Choose a password that no one else can guess easily.

Password guidelines:

> New password must be 6 to 12 characters

> Use a combination of letters and numbers

> Don't use special characters, such as #@!*&

> Current password must match the one on file

> Passwords are case sensitive

> Re-enter your new password (New password in both fields must match)

BMO Password Requirements

← → ↻ Bank of Montreal [CA] | <https://www.bmo.com/olbb/help-centre/en/my-profile/change-password.html>

Help Centre

You can find helpful information about Online Banking in the Help Centre. Simply navigate to the relevant page to find the information you are looking for.

My Accounts | Payments & Transfers | **My Profile & Preferences** | My Messages

My Profile & Preferences

My Contact Information

Mobile Preferences

Change My Password

My Security Settings

Help with Change My Password

How to choose a good password

DO

- Use 6-digit passwords
- Avoid birthday dates, numeric sequences such as 123456 or any other combinations that can be easily guessed
- Change your passwords frequently
- Use different passwords for every system you access

DON'T

- Use words from dictionaries, names of friends or relatives, calendar dates or common phrases
- Use combinations of your name and initials
- Tell anyone your password
- Write passwords on easily accessible places such as your desk calendar or under your keyboard

For more information about protecting yourself when banking online, view our [Online Banking security tips](#).

Authentication Management

■ Password requirement example from Facebook:

facebook

Connect with friends and
around you on Facebook

Enter a combination of at least six numbers, letters
and punctuation marks (like ! and &).

Make a strong Facebook password

When you create a new password, remember:

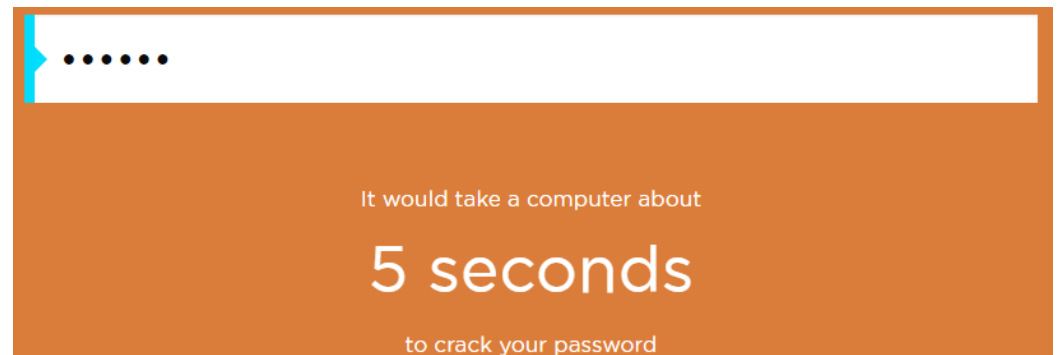
- Your password should be easy for you to remember but difficult for others to guess.
- Your Facebook password should be different than the passwords you use to log into other accounts, like your email or bank account.
- Longer passwords are usually more secure.
- Your password should not be your email, phone number or birthday.
- Avoid using common words, like "Password".
- Use a password manager. There are many different applications that can store your passwords securely.
- Don't share your passwords with anyone, online or in person. If you do, change them as soon as possible.

If you see a message letting you know the password you entered isn't strong enough, try mixing together uppercase and lowercase letters. You can also make the password more complex by making it longer with a phrase or series of words that you can easily remember, but no one else knows.

Learn how to [add two-factor authentication to your Facebook account](#) for additional security.

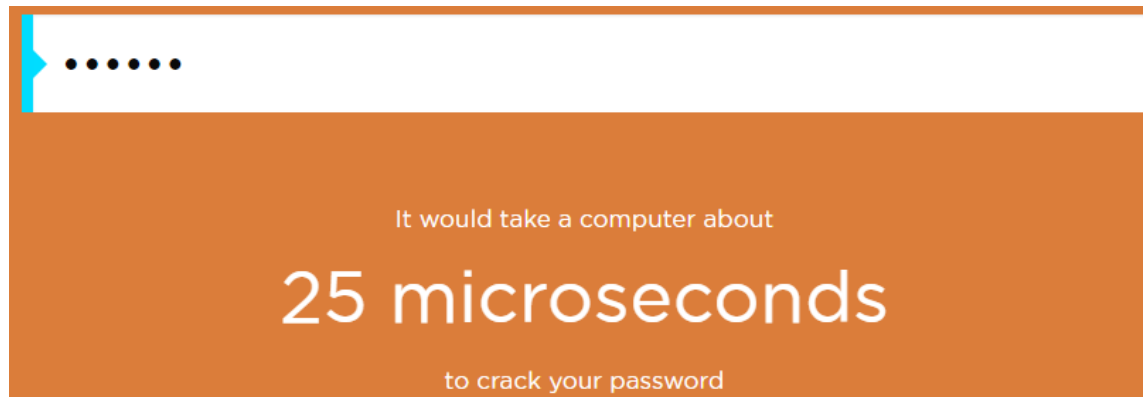
Authentication Management

- Using a password of 6 characters that includes the numbers and letters (both uppercase and lowercase) of the English alphabet gives a number of 62 potential characters
- This is a total of 56,800,235,584 potential passwords



Authentication Management

- Using a password of 6 characters that consists of only numbers gives us a total of 1,000,000 potential passwords
- If a malicious user has the bank card number, the only thing left is the password to obtain access



Authentication Management

- Using a standard internet connection, you can make thousands of attempts per minute
- Client side controls such as cookies, can be bypassed to allow these attempts
- Server side controls often times return a different message if a password was correct even though the account has been locked out for failed attempts

Testing Authentication

Testing Authentication

- Obtain a list of valid usernames
- Test each username with a list of commonly used passwords:
 - ✓ password
 - ✓ The name of the website
 - ✓ 123456
 - ✓ qwerty
 - ✓ letmein
 - ✓ password123
 - ✓ Etc.

Testing Authentication

- If you have a valid account, test to see if the application has a lock out policy
- If it does and you are locked out, test to see if the application reacts differently when proper credentials are supplied than when incorrect credentials are used during the lockout

Testing Authentication

- Most applications have a username and password combination
 - If the application specifies which of the two was incorrect during a login attempt, it narrows down the attack exponentially
- Test for valid usernames either through the login page or by using other functions such as:
 - User registration (Username exists!)
 - Forgot Password

Testing Authentication

- If attempting to brute force email accounts, tools can be used to obtain valid email addresses that can serve as the username
- A lot of organizations use the staff member's email address as the username
- Burp Suite's **Comparer** tool can be used to check if there are any subtle differences in the response from a server if a valid or invalid username has been used

Testing Authentication

- Password change functions in web applications can offer different security practices than login pages:
 - May provide more verbose responses when a username is incorrect
 - Lock lock out rules when entering the **existing password**
 - Offer insight on the existing password when entering a new password and confirming it

Testing Authentication

- Password recovery functions may contain secondary challenges such as:
 - What street did you live on as a child?
 - What was the name of your first pet?
 - What is your mother's maiden name?
- If an application sends a password recovery link to a specified email address, it could be found in a hidden field or a cookie

Testing Authentication

- If you are able to obtain a valid account in the web application, you can try:
 - To analyze any URLs created for password recovery options
 - The **password hints** available in the application
 - Any questions used as a secondary challenge
 - Map out the password recovery function

Testing Authentication

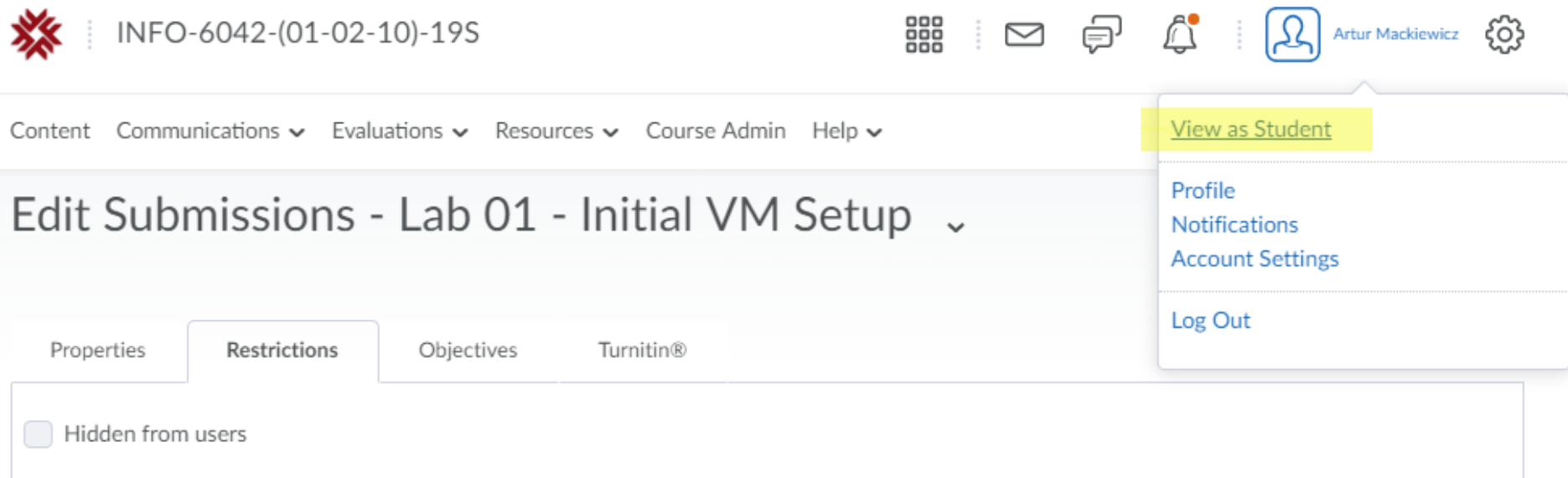
- Check any **remember me** or **stay logged in** type of function by analyzing the cookies
- Check to see if the web application requires a valid user to re-enter their credentials if the option is selected
- Does the session store the username?

Testing Authentication

- Some applications allow users to impersonate or view the app from another user's perspective
- This is often used by support or helpdesk to help users by viewing the portal through their account
- This can be done by setting cookie information, or by using a **backdoor** password that works for every username
 - A Brute force attack may match more than one user with the same password

Testing Authentication

- Example of impersonation on Fanshawe Online that allows faculty to view content as a student



The screenshot displays the Fanshawe Online interface. At the top, the course identifier 'INFO-6042-(01-02-10)-195' is shown. The navigation bar includes links for 'Content', 'Communications', 'Evaluations', 'Resources', 'Course Admin', and 'Help'. The main content area is titled 'Edit Submissions - Lab 01 - Initial VM Setup'. Below this, there are tabs for 'Properties', 'Restrictions', 'Objectives', and 'Turnitin®'. The 'Restrictions' tab is currently selected, showing a checkbox for 'Hidden from users'. On the right side, a user profile for 'Artur Mackiewicz' is visible, with a dropdown menu open showing options: 'View as Student' (highlighted in yellow), 'Profile', 'Notifications', 'Account Settings', and 'Log Out'.

Testing Authentication

- Test out the way an application creates users
 - Does it allow for the same username to be used more than once?
 - Does it auto generate usernames or passwords?
 - These could have a pattern
- Intranet applications typically have a format that is used when creating credentials for new employees

Testing Authentication

- Applications may send an email with credentials or provide a link via email for newly registered users
- You can test sequences in these cases by registering several new accounts and seeing if there is a pattern in the URLs sent to you by the application through email
- Test the activation URL more than once to see if it still works

Testing Authentication

- If an application is vulnerable to command or injection execution, try to find out where user credentials are being stored
 - Some of these credentials will be stored in plain text format (Sony)
 - If they are hashed, check to see if they are being salted
 - Online hash databases can be used to match hashed credentials to plain-text formats

Securing Authentication

Securing Authentication

- All things security related are calculated risks
- The key to securing authentication lays in the balance between security and usability
- Things to consider:
 - Cost of security and support
 - The level of security users are willing to deal with
 - The risk involved with the functionality of the app
 - The value of the data

Securing Authentication

Credential Security

- Long passwords should be used
- Set minimum lengths / Complexity
- Enforce unique usernames
- Utilize high entropy for auto-generated or mass produced passwords

Securing Authentication

Credential Handling

- All client-server communication should be done over encrypted channels
- POST requests should be used to transmit credentials
 - Do not use GET
 - URL parameters
 - Cookies

Securing Authentication

Credential Handling

- Passwords should be stored as salted hashes in the database to prevent recovery
- Passwords should be changed regularly
- Passwords should never be the same as usernames
- Users should be required to change their passwords on initial login

Securing Authentication

Credential Validation

- Applications should not truncate passwords
- Special characters need to be considered even with input validation
- Catch all approaches should force closure of any sessions or login attempts
- User impersonation should be limited to authorized or elevated users only

Securing Authentication

Credential Validation

- Multi stage verification should always be stored on the server side
- Data should not be retransmitted from a client
- At no point should the application allow the client to know which stage of the authentication process has failed

Securing Authentication

Credential Validation

- Any challenge questions should be stored server side and not transmitted to the client
- Challenge questions should not be verified until the entire authentication process has been concluded

Securing Authentication

Information Leakage

- A generic error message should be used for failed login attempts regardless of why the attempt has failed
- An attacker should not receive a different result based on what changed in the authentication parameters

Securing Authentication

Information Leakage

- If users are permitted to create their own username during the registration process, email should be used as the username
- This prevents the attacker from enumerating usernames because a generic message can be displayed
 - “Please check your email to continue registration”

Securing Authentication

Brute-Force

- Accounts can be disabled after a number of failed login attempts
 - This can lead to DOS for legitimate users
- Would be better to implement account suspension
 - Lock account out for 30 mins
 - Slows down a brute force attack considerably

Securing Authentication

Brute-Force

- Accounts that have been suspended should not divulge any information regarding the lock out policy in place
 - Use a generic error message
- The application should not process login attempts for suspended accounts
 - This could potentially lead to information leakage allowing for brute-force attacks to continue

Securing Authentication

Brute-Force

- If using technologies such as CAPTCHA to enhance the prevention of brute-force attacks, ensure that it is implemented correctly
- Answers to the puzzle should not be available in the HTML form
 - Alt attribute of images
 - Hidden input fields

Securing Authentication

Password Change

- Users should be allowed to periodically change their passwords
- This function should only be allowed for authenticated sessions and force the user to enter their current valid password
 - The username should not be used as there is no reason why you should need to change another user's password

Securing Authentication

Password Change

- This function should also have a limit on how many attempts a user can have to change their password
- Disable the function if a user has incorrectly entered their current password or made mistakes entering the new password
 - Passwords don't match error

Securing Authentication

Password Change

- Users should be notified if their password has been changed through a means outside of the application
- An email can be generated to the user's email address but must not contain any information about the new or previous password

Securing Authentication

Password Recovery

- Security critical applications such as online banking, should not use automated password recovery functions but rely on other methods
 - Telephone
 - Visit a branch
 - Etc.

Securing Authentication

Password Recovery

- When implementing an automated password recovery function:
 - Do not use password hints
 - Send an email to the address on the account with a link to a URL that is:
 - Unique
 - Time-limited
 - Using high entropy

Securing Authentication

Password Recovery

- Challenge questions should be set by the application and not by the user
 - Users can create weak challenge questions or make the answer obvious in the question
- Answers should use high entropy
 - What was the name of the street you grew up on?
Not What is your favourite colour?

Summary

- Authentication functions are accessible by anonymous and unauthenticated users making them a target for attacks
- In addition to the main login page, other functions such as change password, reset password, and register user must be secured so that things such as brute-force attacks and information leakage are minimized

Summary

- Authentication related events need to be logged
 - Login/logout times
 - Password changes/resets
 - Account suspension
 - Username
 - IP
- These logs should never contain a user's credentials

Summary

- Users should be notified within the application of things such as
 - Last login date and time
 - IP address of last login
 - Type of device (Windows, Linux, Android, etc.)
- Users should be notified outside of the application of critical security events
 - Password change
 - Password recovery

Lab Details

LAB-08: Overview

Lab-08: Session Attacks

- Capture user data
- Cain and Abel
- Burp Suite Sequencer
- Send cookie information to the attacker
- Juice Shop Challenge