NTNU

Norwegian University of Science and Technology

OWASP Testing Guide - part one

TDT4237 - 2025

Outline

Information gathering → Section 4.1

Injection attacks → Section 4.7

Session management attacks → Section 4.6

https://owasp.org/www-project-web-security-testing-guide/stable/

Reference group

Send an email to jingyue.li@ntnu.no by 1st of Feb.

Information gathering

- Why information gathering?
  - Attacker
    - A map to attack
    - Look for low hanging fruit
    - Improve attack efficiency
  - Developer/internal tester
    - Decide test scope, coverage, prioritization
    - Improve test efficiency

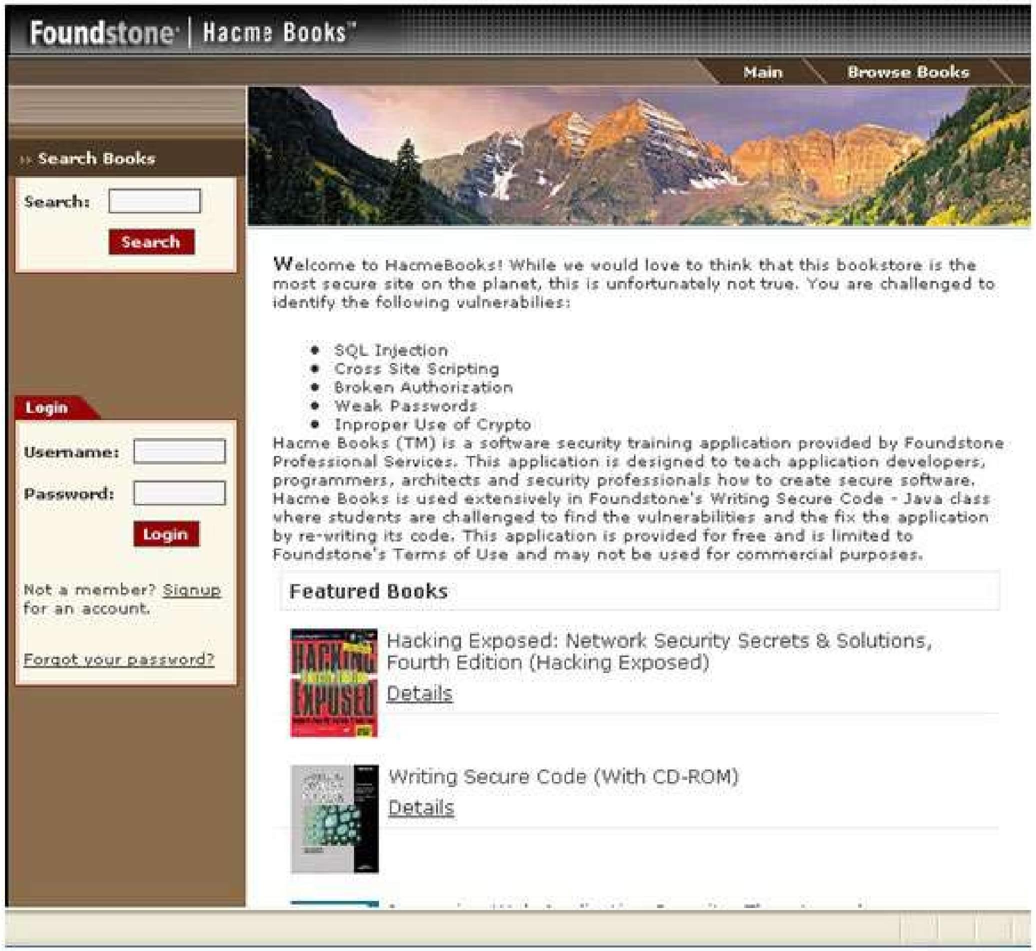
The more you know about the application's structure, the better you can plan your tests!

NTNU

What information to gather?

- Application structure
  - All pages you have found in the application
    - Including subdomains
  - Any external links
  - Trust zones
    - Needs authentication vs. open
- Data flow within the application, e.g.,
  - Parameters and value
  - Get and post, responses

Page map example - Hacmebooks



Not a member? Signup for an account.

Forgot your password?

Welcome to HacmeBooks! While we would love to think that this bookstore is the most secure site on the planet, this is unfortunately not true. You are challenged to identify the following vulnerabilies:

- SQL Injection
- Cross Site Scripting
- Broken Authorization
- Weak Passwords
- Inproper Use of Crypto

Hacme Books (TM) is a software security training application provided by Foundstone Professional Services. This application is designed to teach application developers, programmers, architects and security professionals how to create secure software. Hacme Books is used extensively in Foundstone's Writing Secure Code - Java class where students are challenged to find the vulnerabilities and the fix the application by re-writing its code. This application is provided for free and is limited to Foundstone's Terms of Use and may not be used for commercial purposes.

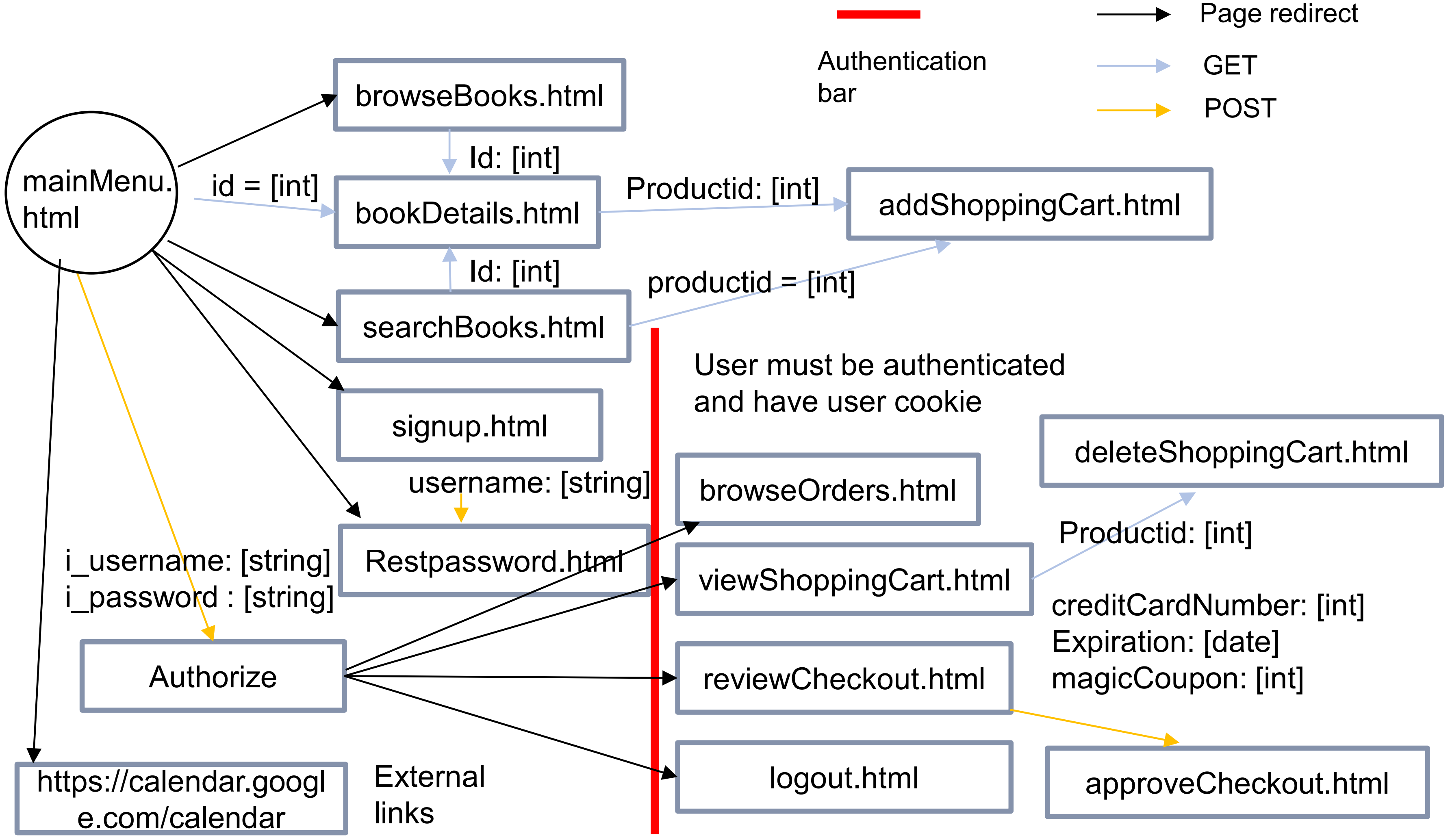
Featured Books

Hacking Exposed: Network Security Secrets & Solutions, Fourth Edition (Hacking Exposed)
Details

Writing Secure Code (With CD-ROM)
Details

https://webapppentest.wordpress.com/2012/11/26/hacme-books-week-1/

Simplified Hacmebooks page map



Other information to gather

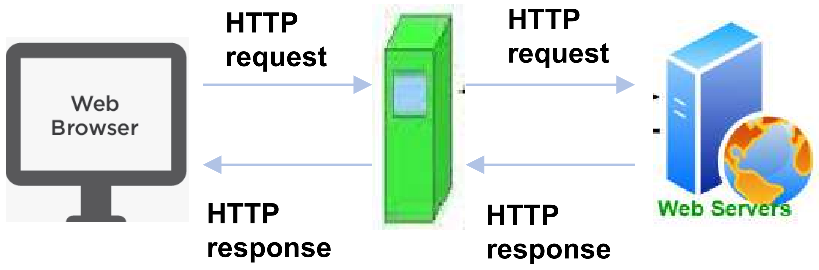
- Infrastructure or platform, e.g.,
  - Web server (WSTG-INFO-02)
  - Applications on the webserver (WSTG-INFO-04)
  - Application entry points (WSTG-INFO-06)
  - Execution path through application (WSTG-INFO-07)
  - Web application framework (WSTG-INFO-08)

The IDs here refer to the ones in OWASP Web Security Testing Guide v4.2

Demo

Why use web debugging proxy?

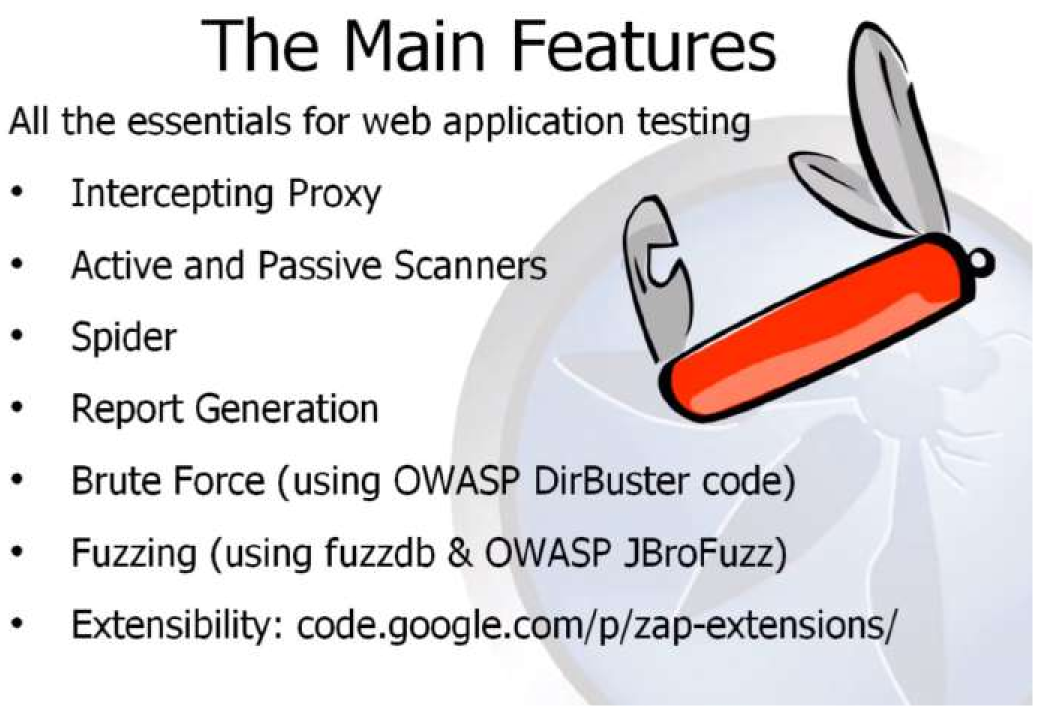
- To capture and examine requests and responses
- To manipulate payloads
- Can also be used for attacks



Tools for information gathering

- Website copier (e.g., HTTtrack, VisualWget)
- Web debugging proxy server (e.g., Firefox Developer Tools, Fiddler)
- Tool sets (e.g., Kali Linux, Burp Suite and OWASP Zap)

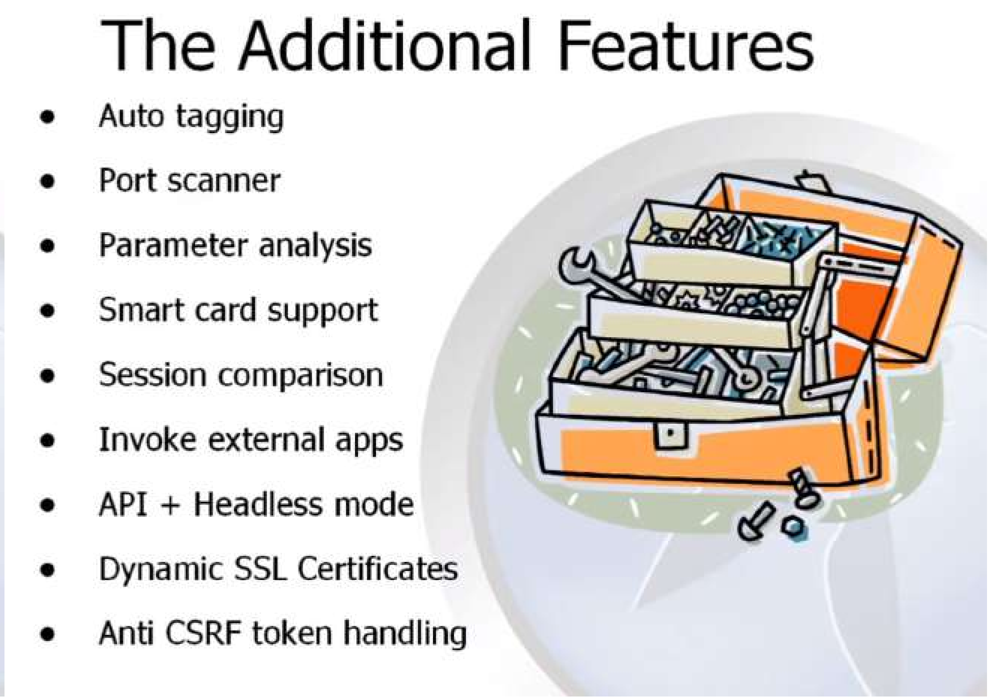
OWASP ZAP



The Main Features

All the essentials for web application testing

- Intercepting Proxy
- Active and Passive Scanners
- Spider
- Report Generation
- Brute Force (using OWASP DirBuster code)
- Fuzzing (using fuzzdb & OWASP JBroFuzz)
- Extensibility: code.google.com/p/zap-extensions/

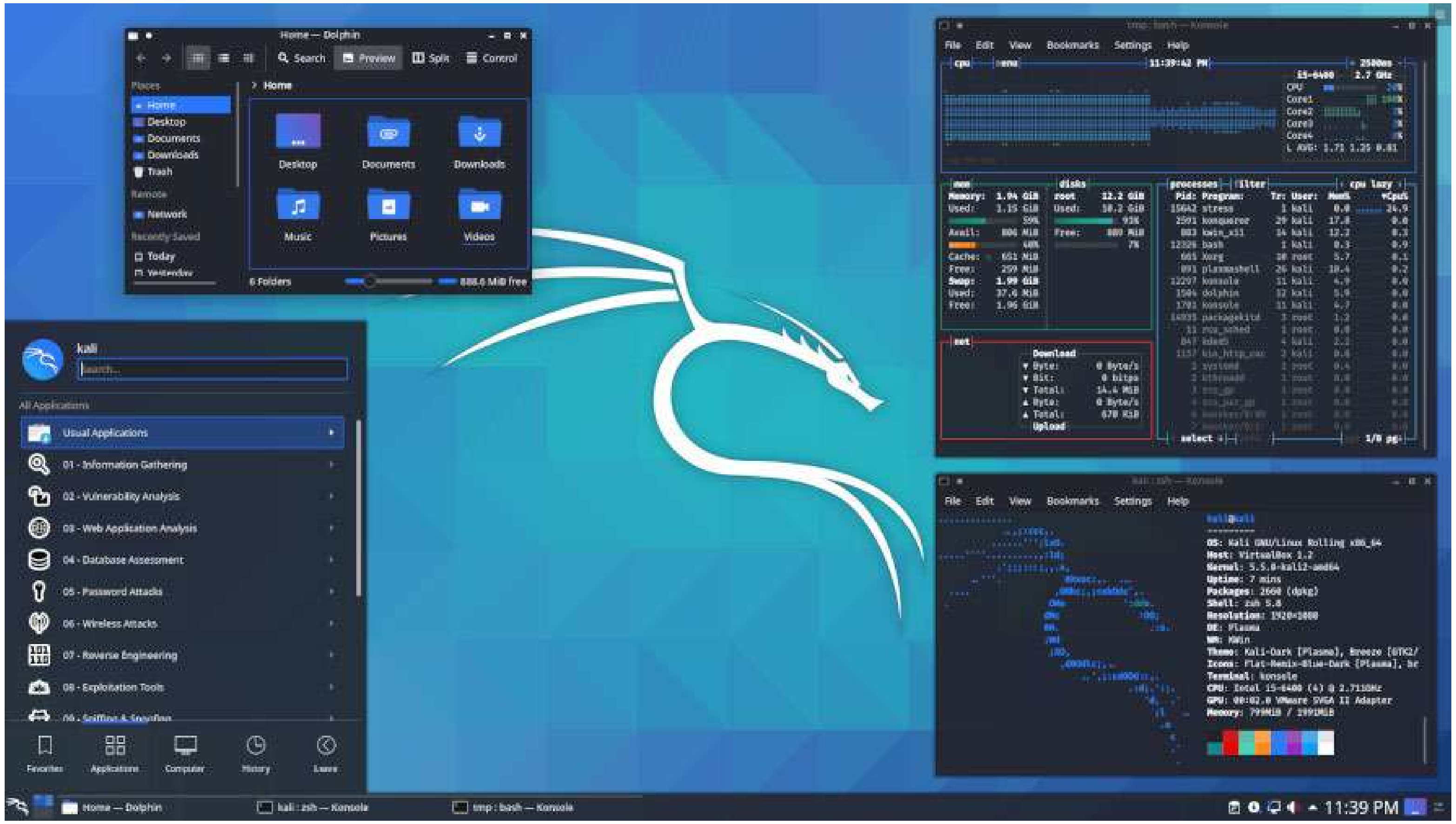


The Additional Features

- Auto tagging
- Port scanner
- Parameter analysis
- Smart card support
- Session comparison
- Invoke external apps
- API + Headless mode
- Dynamic SSL Certificates
- Anti CSRF token handling

https://www.zaproxy.org/

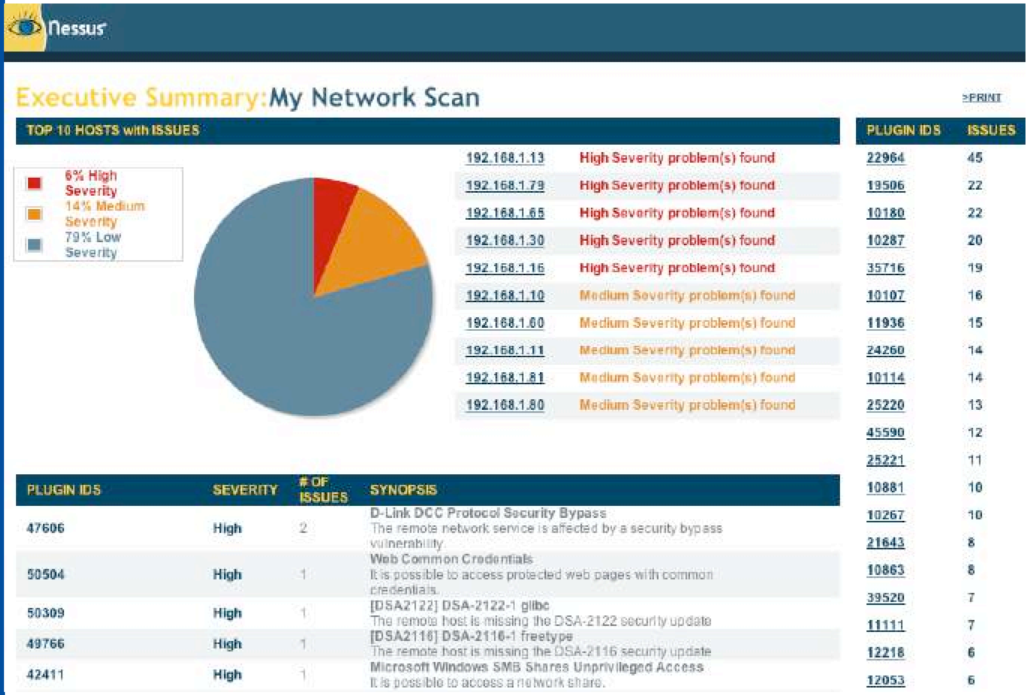
Kali Linux



```
kali@kali
---------
OS: Kali GNU/Linux Rolling x86_64
Host: VirtualBox 1.2
Kernel: 5.5.0-kali2-amd64
Uptime: 7 mins
Packages: 2660 (dpkg)
Shell: zsh 5.8
Resolution: 1920x1080
DE: Plasma
WM: KWin
Theme: Kali-Dark [Plasma], Breeze [GTK2/
Icons: Flat-Remix-Blue-Dark [Plasma], br
Terminal: konsole
CPU: Intel i5-6400 (4) @ 2.711GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 799MiB / 1991MiB
```

https://www.kali.org/

Vulnerability scanners



Executive Summary:My Network Scan

>PRINT

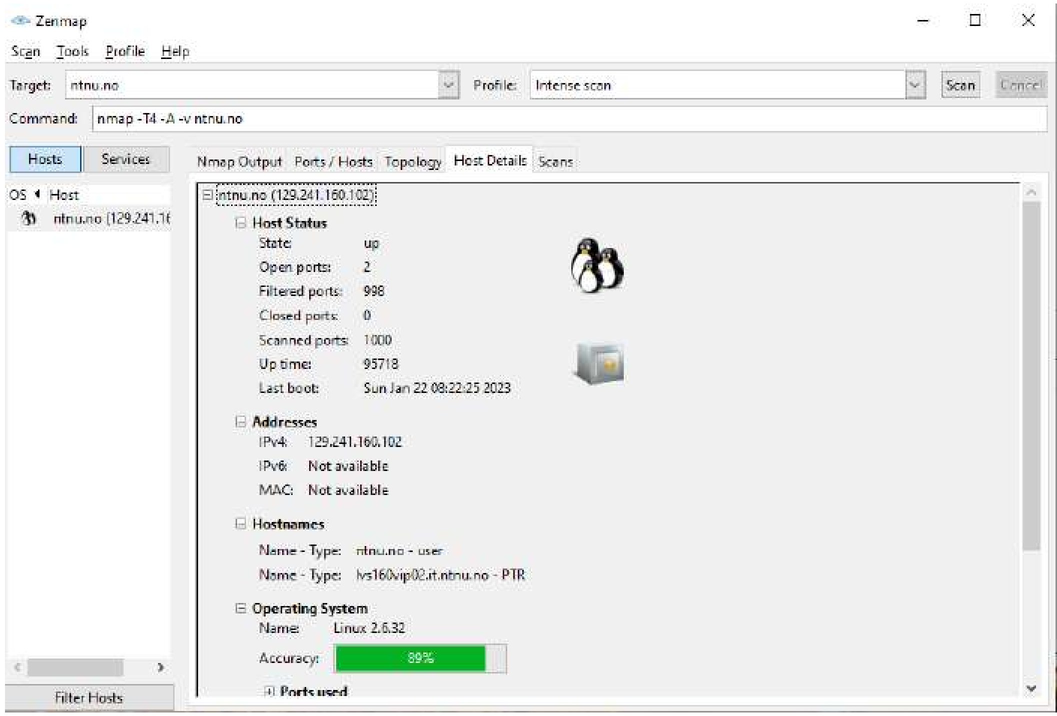
TOP 10 HOSTS with ISSUES

- 🟥 6% High Severity
- 🟧 14% Medium Severity
- 🟦 79% Low Severity

192.168.1.13	High Severity problem(s) found
192.168.1.79	High Severity problem(s) found
192.168.1.65	High Severity problem(s) found
192.168.1.30	High Severity problem(s) found
192.168.1.16	High Severity problem(s) found
192.168.1.10	Medium Severity problem(s) found
192.168.1.60	Medium Severity problem(s) found
192.168.1.11	Medium Severity problem(s) found
192.168.1.81	Medium Severity problem(s) found
192.168.1.80	Medium Severity problem(s) found

PLUGIN IDS	ISSUES
22964	45
19506	22
10180	22
10287	20
35716	19
10107	16
11936	15
24260	14
10114	14
25220	13
45590	12
25221	11
10881	10
10267	10
21643	8
10863	8
39520	7
11111	7
12218	6
12053	6

PLUGIN IDS	SEVERITY	# OF ISSUES	SYNOPSIS
47606	High	2	D-Link DCC Protocol Security Bypass The remote network service is affected by a security bypass vulnerability.
50504	High	1	Web Common Credentials It is possible to access protected web pages with common credentials.
50309	High	1	[DSA2122] DSA-2122-1 glibc The remote host is missing the DSA-2122 security update
49766	High	1	[DSA2116] DSA-2116-1 freetype The remote host is missing the DSA-2116 security update
42411	High	1	Microsoft Windows SMB Shares Unprivileged Access It is possible to access a network share.



Target: ntnu.no
Profile: Intense scan
Command: nmap -T4 -A -v ntnu.no

OS Host
ntnu.no (129.241.16

ntnu.no (129.241.160.102)

Host Status

State:	up
Open ports:	2
Filtered ports:	998
Closed ports:	0
Scanned ports:	1000
Up time:	95718
Last boot:	Sun Jan 22 08:22:25 2023

Addresses

IPv4:	129.241.160.102
IPv6:	Not available
MAC:	Not available

Hostnames

Name - Type:	ntnu.no - user
Name - Type:	lvs160vip02.it.ntnu.no - PTR

Operating System

Name:	Linux 2.6.32
Accuracy:	89%

Ports used

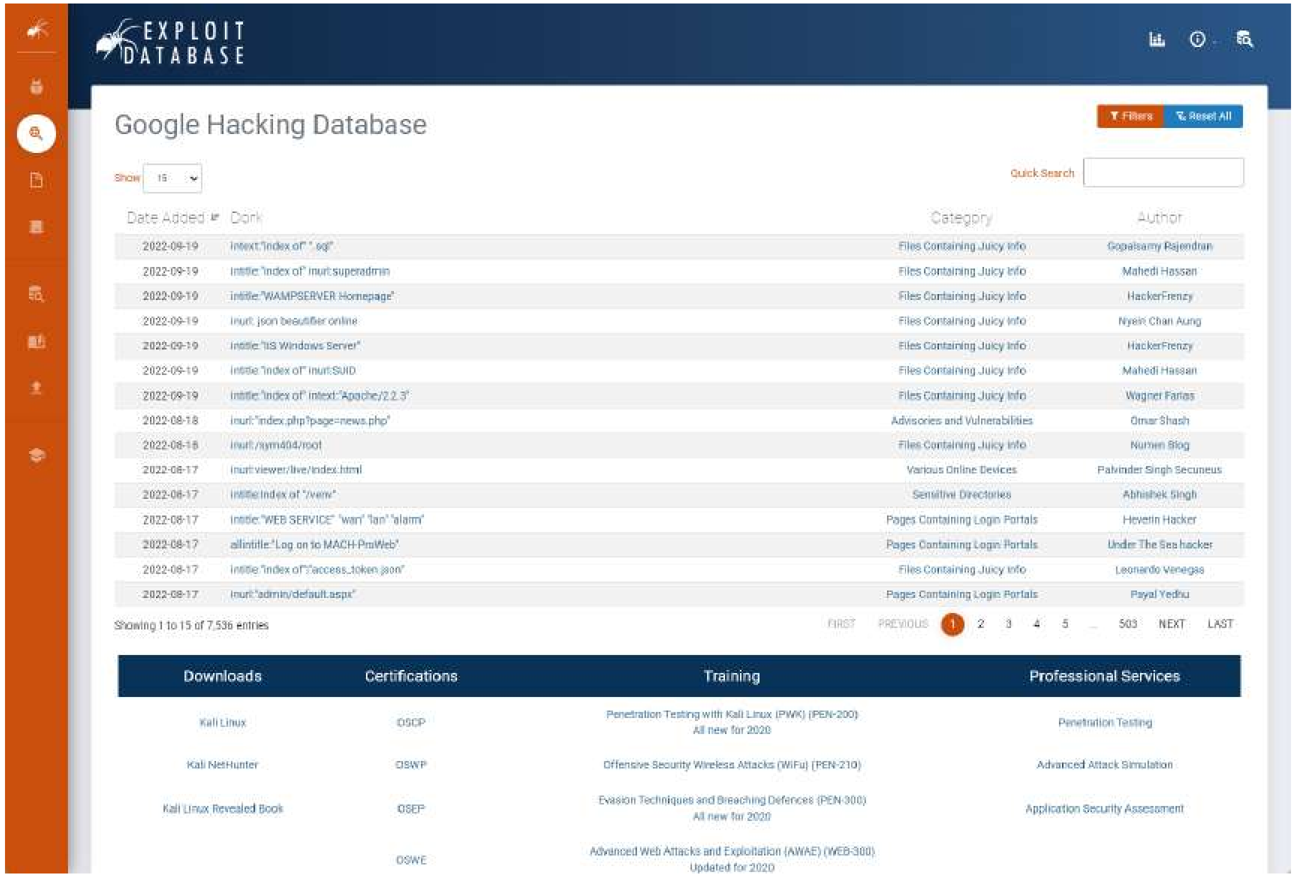
https://community.tenable.com/s/article/Nessus-Essentials

https://nmap.org/

Automated scanners are limited

- Some information and vulnerabilities cannot be found using automated scanners
- Additional manual security testing is always recommended

Dorking (Google hacking)



Google Hacking Database

Date Added	Dork	Category	Author
2022-09-19	intext:"index of" ".sql"	Files Containing Juicy Info	Gopalsamy Rajendran
2022-09-19	intitle:"index of" inurl:superadmin	Files Containing Juicy Info	Mahedi Hassan
2022-09-19	intitle:"WAMPSERVER Homepage"	Files Containing Juicy Info	HackerFrenzy
2022-09-19	inurl: json beautifier online	Files Containing Juicy Info	Nyein Chan Aung
2022-09-19	intitle:"IIS Windows Server"	Files Containing Juicy Info	HackerFrenzy
2022-09-19	intitle:"index of" inurl:SUID	Files Containing Juicy Info	Mahedi Hassan
2022-09-19	intitle:"index of" intext:"Apache/2.2.3"	Files Containing Juicy Info	Wagner Farias
2022-08-18	inurl:"index.php?page=news.php"	Advisories and Vulnerabilities	Omar Shash
2022-08-18	inurl:/sym404/root	Files Containing Juicy Info	Numen Blog
2022-08-17	inurl:viewer/live/index.html	Various Online Devices	Palvinder Singh Secuneus
2022-08-17	intitle:index of "/venv"	Sensitive Directories	Abhishek Singh
2022-08-17	intitle:"WEB SERVICE" "wan" "lan" "alarm"	Pages Containing Login Portals	Heverin Hacker
2022-08-17	allintitle:"Log on to MACH-ProWeb"	Pages Containing Login Portals	Under The Sea hacker
2022-08-17	intitle:"index of" "/access_token.json"	Files Containing Juicy Info	Leonardo Venegas
2022-08-17	inurl:"admin/default.aspx"	Pages Containing Login Portals	Payal Yedhu

Showing 1 to 15 of 7,536 entries

FIRST PREVIOUS 1 2 3 4 5 ... 503 NEXT LAST

Downloads	Certifications	Training	Professional Services
Kali Linux	OSCP	Penetration Testing with Kali Linux (PWK) (PEN-200) All new for 2020	Penetration Testing
Kali NetHunter	OSWP	Offensive Security Wireless Attacks (WiFu) (PEN-210)	Advanced Attack Simulation
Kali Linux Revealed Book	OSEP	Evasion Techniques and Breaching Defences (PEN-300) All new for 2020	Application Security Assessment
	OSWE	Advanced Web Attacks and Exploitation (AWAE) (WEB-300) Updated for 2020	

https://www.exploit-db.com/google-hacking-database
https://resources.bishopfox.com/resources/tools/google-hacking-diggity/

NTNU

Demo

NTNU

Injection Attacks

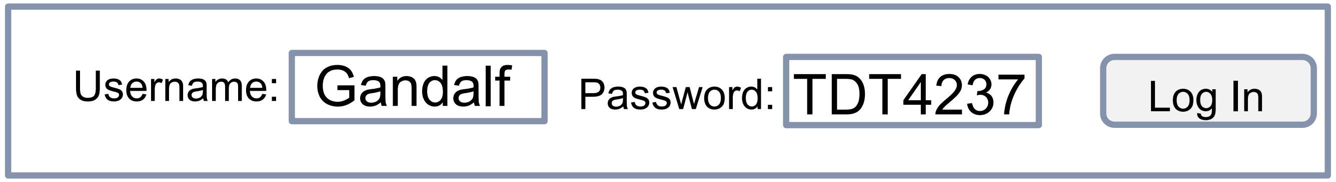
<< All input is evil. >>

Michael Howard
Principal Consultant Cybersecurity with Microsoft

Injection attacks

- SQL injection
- Blind SQL injection
- Xpath injection
- …

SQL injection – normal input


Username: Gandalf Password: TDT4237 Log In

“Server-side login code (E.g., PHP)”

$ result = mysql_query (“ select * from Users where (name = ‘$ user’ and password = ‘$pass’); ”);

Application constructs SQL query from parameter to DB, e.g.,

Select * from Users where name = Gandalf and password = TDT4237

SQL injection – Attack scenario (1)

- Attacker types in the string below in the ***username*** field

  Gandalf ' OR 1=1); --

- At the server side, the code to be executed

$ result = mysql_query (" select * from Users where (name = 'Gandalf' OR 1=1); -- and password = 'whocares'); ");

- SQL query constructed is

Select * from Users

where name = Gandalf OR 1= 1

1=1 is always true.

NTNU

SQL injection – Attack scenario (2)

- Attacker types the following string in the ***username*** field

  Gandalf ' OR 1=1); Drop TABLE Users; --

- SQL query constructed is

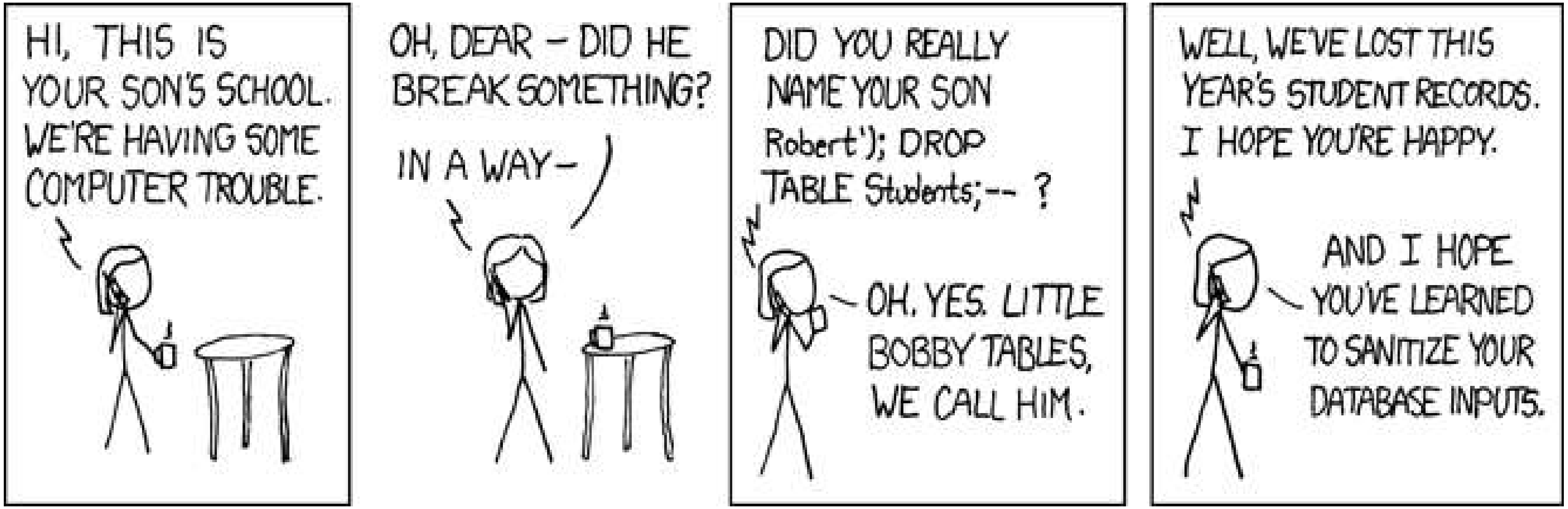
  select * from Users

  where name = Gandalf OR 1= 1;

  drop TABLE Users;

Delete the Table Users

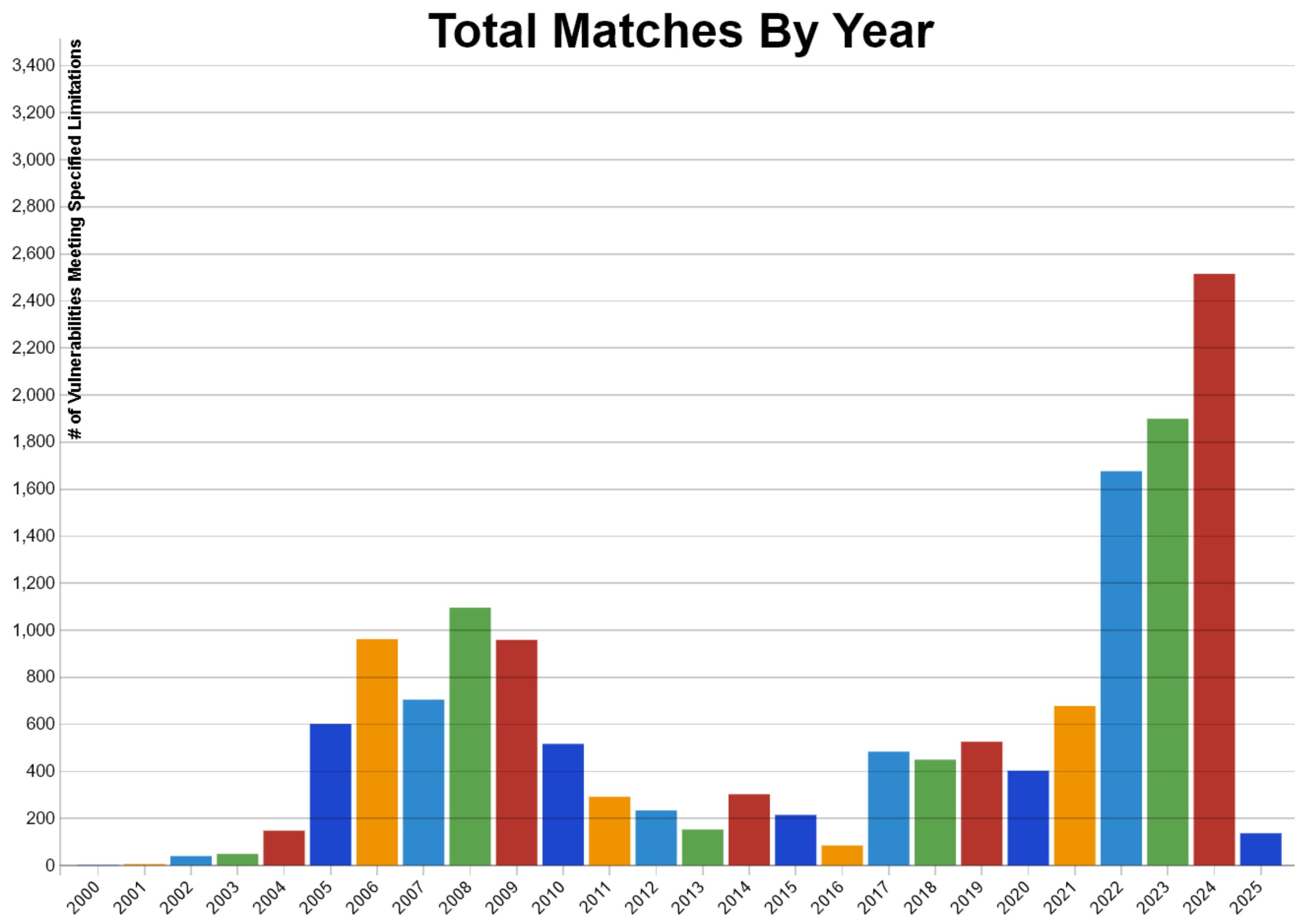
SQL injection humor


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

https://xkcd.com/327/

https://nvd.nist.gov/vuln/search

…not just humor



…some notable events

- **Tesla 2014:** Security researchers breached the website of Tesla using SQL injection, could gain administrative privileges and steal user data.
- **Fortnite 2019:** Fortnite is an online game with over 350 million users. A SQL injection vulnerability was discovered which could let attackers access user accounts.
- **WordPress 2022:** LearnPress plugin vulnerable, 75K sites impacted

https://brightsec.com/blog/sql-injection-attack/
https://www.indiehackers.com/post/sql-injection-real-life-attacks-and-how-it-hurts-business-c7ff42ef30

Why so common?

What can you achieve?

- Bypass authentication
- Privilege escalation
- Stealing information
- Destruction

NTNU

Blind SQL injection

- Is the site vulnerable to SQL injection?
  - First register as a legal user, e.g. “Sauron”
  - Then, run SQL inject attack and see results

Sauron ’ AND 1=1); --

TRUE

Server side: SELECT Id FROM Users WHERE (‘userID= Sauron’AND 1=1); --

Info. Related to the Id shows → web app is vulnerable to SQL injection

Blind SQL injection (cont’)

- Guess DB schema through a binary search

Q: What is the first letter of a Table in DB?

SELECT Id from Users WHERE userID= Sauron AND ascii(low (substring ((SELECT Top 1 name FROM sysobjects WHERE xtype = ‘U’), 1, 1))) > 109

- First letter after m (ascii of m is 109), “*Id*” will show
- First letter before m, “*Id*” will not show

Xpath injection

User/password/account DB in XML (users.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <users>
        <user>
            <username>gandalf</username>
            <password>Abcd3</password>
            <account>admin</account>
        </user>
        <user>
            <username>Stefan0</username>
           <password>w1s3c</password>
           <account>guest</account>
        </user>
    </users>
```

Xpath injection (cont’)

- Normal Xpath query

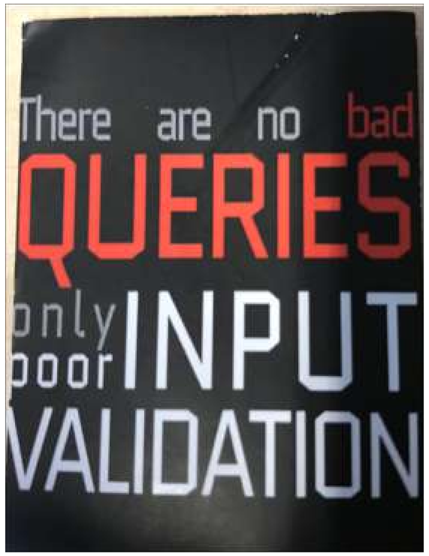
  string(//user[username/text()=’gandalf’ and password/text()=’Abcd3’]/account/text())

- Attack query

  string(//user[username/text()=’’ or ‘1’ = ‘1’ and password/text()=’’ or ‘1’ = ‘1’]/account/text())

SQL injection countermeasures

- Blacklisting
- Whitelisting
- Escaping
- Prepared statement & bind variables
- Mitigating impact



Blacklisting

Filter quotes, semicolons, whitespace, and …?

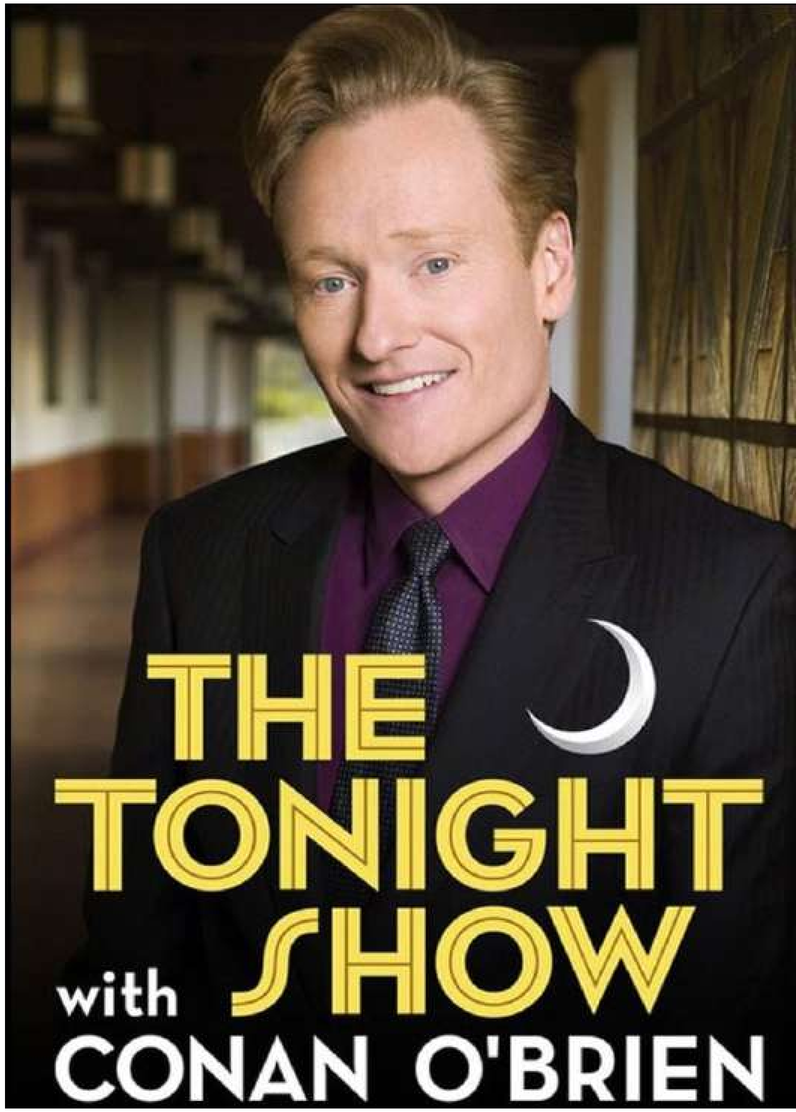
– E.g. kill_quotes (Java) removes single quotes

```
String kill_quotes(String str) {
  StringBuffer result = new    StringBuffer(str.length());
  for (int i = 0; i < str.length(); i++) {
    if (str.charAt(i) != '\'')
      result.append(str.charAt(i));
  }
  return result.toString();
}
```

user1 ’ OR 1=1); --

Pitfalls of Blacklisting

- Could miss dangerous characters
- May conflict with functional requirements
  - E.g., a user with name O’Brien



Whitelisting

- Only allow well-defined safe inputs
- Using RegExp (regular expressions) match string
  - E.g., *month* parameter: non-negative integer
    - RegExp: ^[0-9]+$
    - ^ beginning of string, $ end of string
    - [0-9] + matches a digit, + specifies 1 or more
- Pitfalls: Hard to define RegExp for all safe values

Escaping

- Could escape quotes instead of blacklisting
  - E.g., Escape(O’Brien) = O’’Brien

INSERT INTO USERS(username, passwd) VALUES (‘O’’Brien’, ‘mypasswd’)

- Pitfalls: like blacklisting, could always miss a dangerous character

Prepared statements & Bind variables

- Root cause of SQL injection attack
  - Data interpreted as control, e.g., Gandalf ' OR 1=1); --,
- Idea: decouple query statement and data input

NTNU

Example of Java prepared statement

```
PreparedStatement stmt=con.prepareStatement("update emp set name=?
where id=?");

stmt.setString(1,“Gandalf"); //1 specifies the first parameter in the query

stmt.setInt(2,101);

int i=stmt.executeUpdate();
```

https://www.javatpoint.com/PreparedStatement-interface

Example of Python prepared statement

```
query = """Update employee set Salary = %s where id = %s"""

input = (8000, 101)

cursor.execute(query, input)
```

https://pynative.com/python-mysql-execute-parameterized-query-using-prepared-statement/

Mitigating impact

- Avoid information leakage
  - Don’t display a detailed error message to external users
  - Don’t display stack traces to external users
- Limiting privileges
  - No more privileges than users need
  - E.g., No drop table privilege for a typical user

Mitigate impact (cont')

- Encrypt sensitive data, e.g.,
  - Username, credit card number, magical powers

- Key management precautions
  - Do not store the encryption key in DB

- Hash password

OWASP SQL injection test cases

- Testing for SQL Injection (WSTG-INPV-05)
  - Oracle Testing
  - MySQL Testing
  - SQL Server Testing
  - Testing PostgreSQL
  - MS Access Testing
  - Testing for NoSQL injection

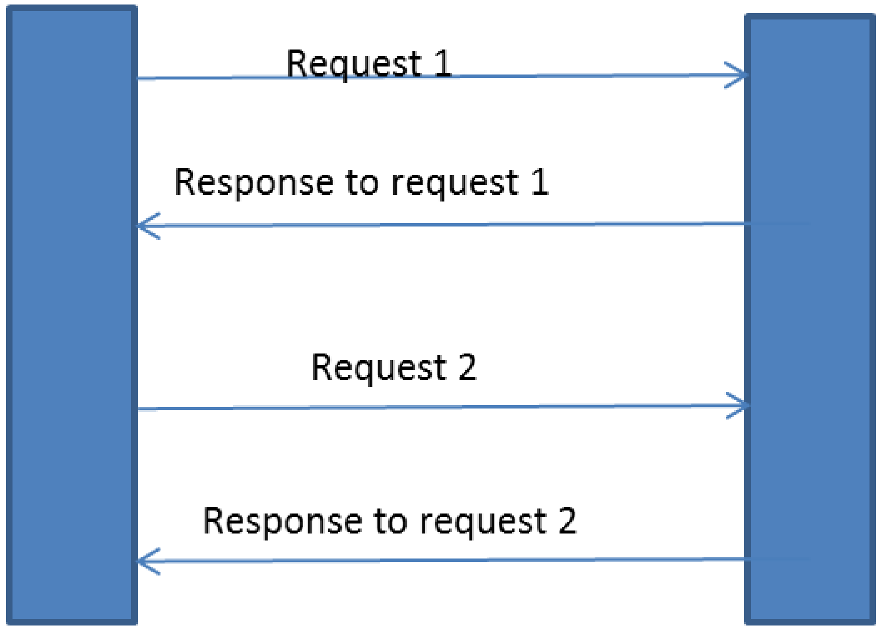
OWASP other injection test cases

- Testing for LDAP Injection (WSTG-INPV-06)
- Testing for XML Injection (WSTG-INPV-07)
- Testing for SSI Injection (WSTG-INPV-08)
- Testing for XPath Injection (WSTG-INPV-09)
- IMAP/SMTP Injection (WSTG-INPV-10)
- Testing for Code Injection (WSTG-INPV-11)

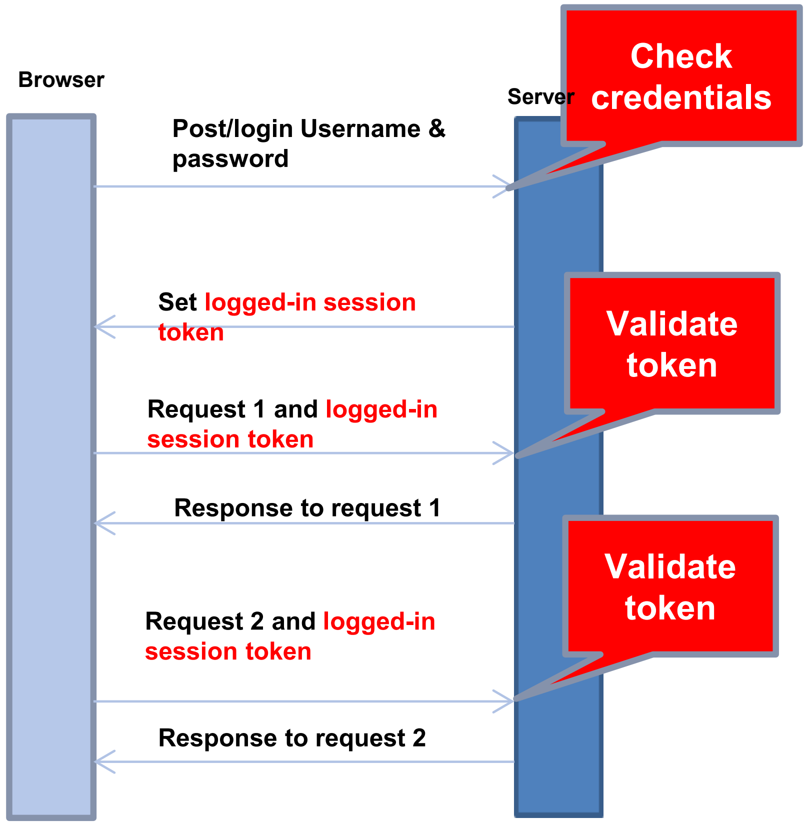
Session Management Attacks

Why session management?

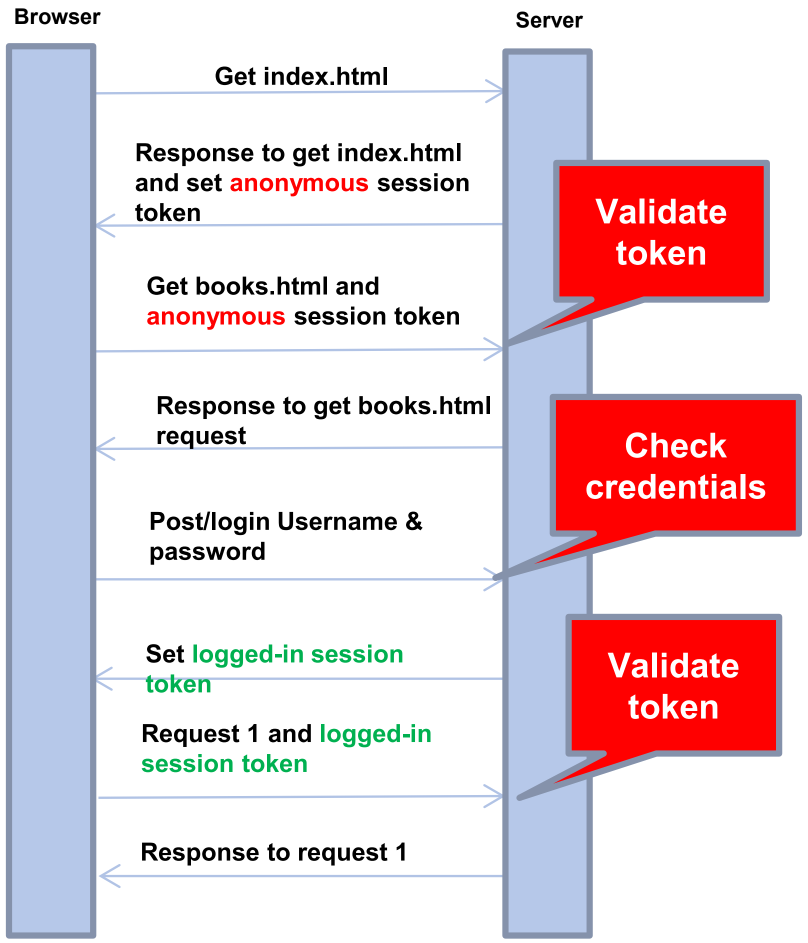
- HTTP is stateless
- Impossible to know if Req1 and Req2 are from the same client
- Users would have to constantly re-authenticate
- Session management
  - Authenticate user once
  - All subsequent requests are tied to the user



Session tokens


Browser
Server
Check credentials
Post/login Username & password
Validate token
Set logged-in session token
Request 1 and logged-in session token
Response to request 1
Validate token
Request 2 and logged-in session token
Response to request 2

e.g., https://ntnu.inspera.no/admin


Browser
Server
Get index.html
Response to get index.html and set anonymous session token
Validate token
Get books.html and anonymous session token
Response to get books.html request
Check credentials
Post/login Username & password
Set logged-in session token
Validate token
Request 1 and logged-in session token
Response to request 1

e.g., amazon.com

Where to store session token

- Embed in all URL links

  https://site.com/checkout?sessionToken= 1234

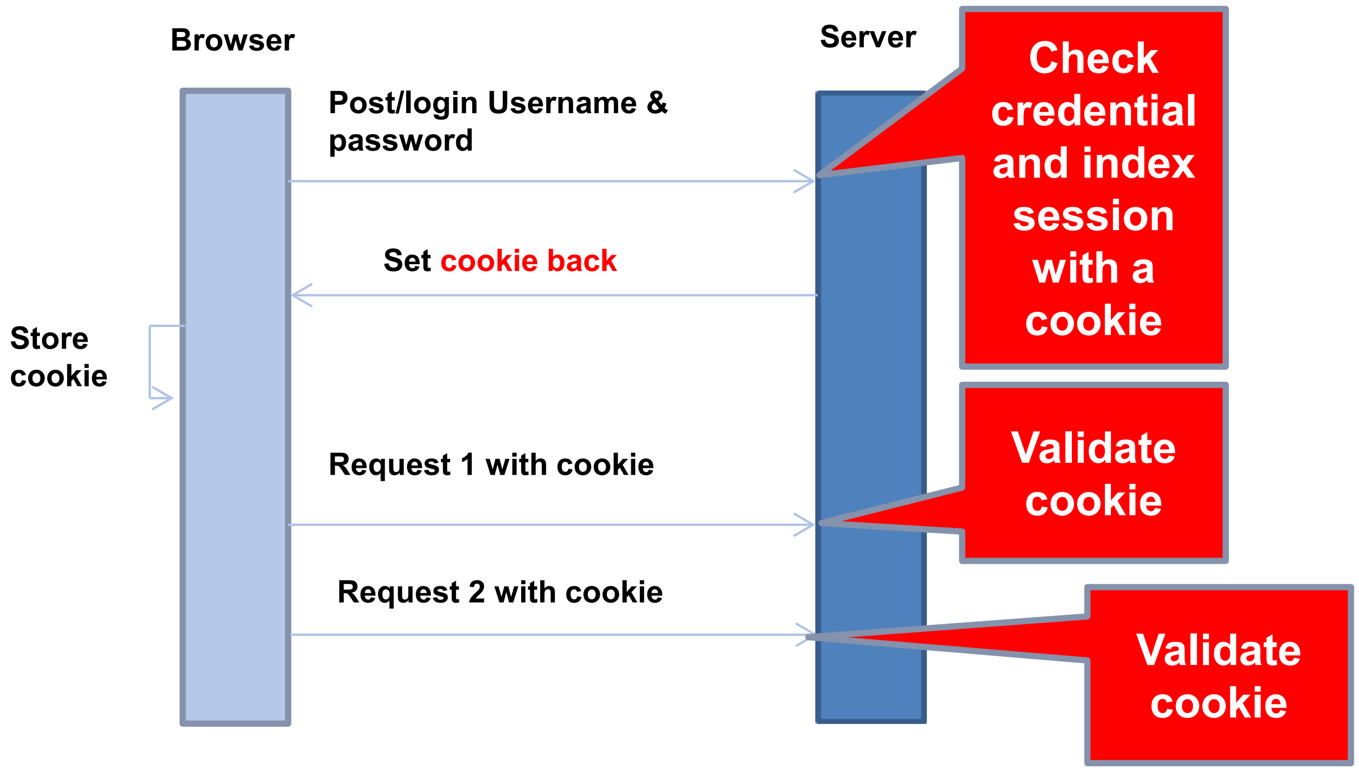
- In hidden form field

  <input type= “hidden” name = “sessionToken” value = “1234”>

- Browser cookie

  setcookie: sessionToken = 1234

Session management with cookie



How cookies work

- Setting and sending cookies
  - In header of HTTP response (Server to browser)

set-Cookie: token=**1234**; expire=Wed, 3-Aug-2025 08:00:00; path=/; domain = idi.ntnu.no

  - In header of HTTP request (Browser to server, when visiting the domain of the same scope)

Cookie: token=**1234**

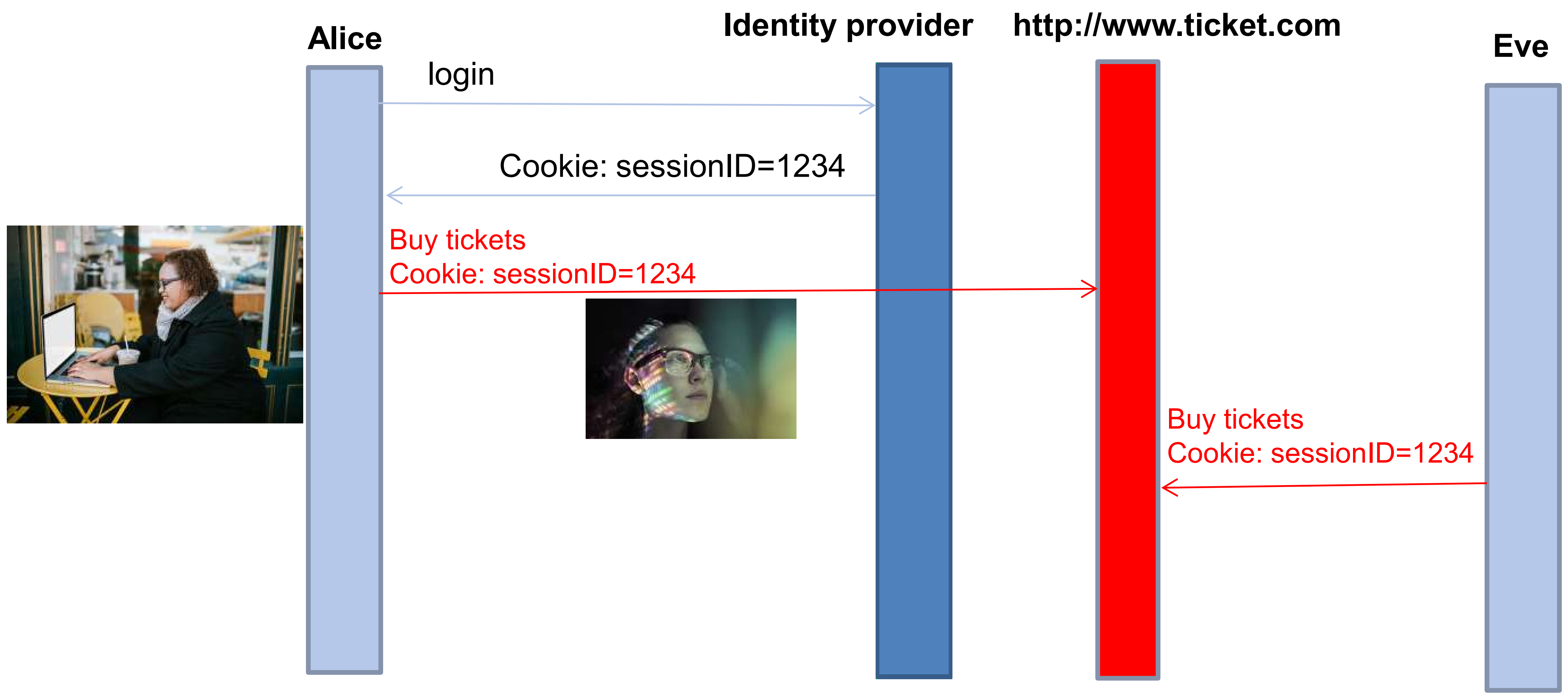
Vulnerable to session management attacks

- Cookie protocol problem
  - Sever only sees Cookie: NAME = VALUE
  - Server does not see which domain sends the cookie

Session management attacks and countermeasures

- Session token theft
- Session token predication attack
- Session fixation attack

Session token theft – Sniff network



Session token theft – Logout problem

- What should happen during logout
  - 1. Delete session token from the client
  - 2. Mark session token as expired on the server
  - Many do (1) but not (2)!!
- Attacker
  - If he can impersonate once, he can impersonate for a long time
  - E.g., Twitter sad story
    - Tokens not invalidated, replay attacks!

      https://packetstorm.news/files/id/119773

Solutions to Session token theft

- Once user logged in (i.e., session token issued), all later communication between browser and server shall use an encrypted channel (e.g., HTTPS)
- Remember to log out
- Time-out session ID
- Delete expired session ID
- Binding session token to the client’s IP or computer

NTNU

More about cookies

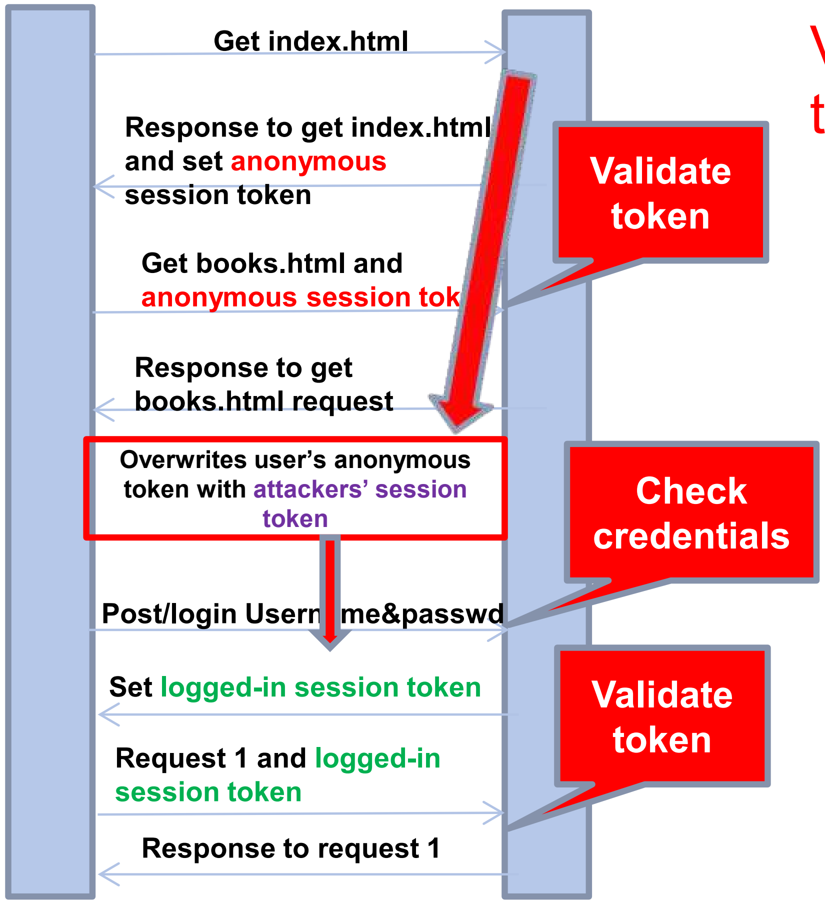
Session cookies	Persistent cookies
• Temporary cookies stored in the browser’s memory just until the browser is closed • Lower risks • E.g. Online banks	• Longer-term cookies that are tagged by the issuer with an expiration date • Stored by the browser even after the browser is closed • E.g., Google or Facebook to create a log of user activity • When clicking “Remember me”

https://www.cookiebot.com/en/session-cookies/

Session token predication attack

- Predicable tokens, e.g., counter
  - jsessionid=user001
  - jsessionid=user002
  - jsessionid=user00?
- Non-predicable token: Seeing one or more token, should not be able to predict other tokens
- Solution:
  - Do not invent your own token generator algorithm
  - Use token generators from frameworks (e.g., ASP, Tomcat, Rails, Django)

Session fixation attack


Get index.html
Response to get index.html and set anonymous session token
Validate token
Get books.html and anonymous session tok
Response to get books.html request
Overwrites user's anonymous token with attackers' session token
Check credentials
Post/login Usern me&passwd
Set logged-in session token
Validate token
Request 1 and logged-in session token
Response to request 1

Vulnerability: Server elevates the anonymous token without changing the value

Attack steps

1. User (e.g., Alice):
   Visits site using an anonymous token
2. Attacker
   Overwrites user's anonymous token with own token
3. User:
   Logs in and gets anonymous token elevated to logged-in token
4. Attacker:
   Attacker's token gets elevated to logged-in token after user logs in

How to overwrite session token?

- Tampering through network
  - Alice visits server using non-encrypted channel (HTTP)
  - The attacker injects into the response to overwrite the secure cookie

    Set-cookie: SSID=maliciousToken;
- Cross-site scripting (XSS)
  - How? Will explain more in XSS attack slides

Mitigate session fixation

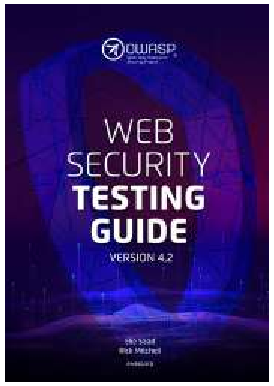
- Always issue a new session token, when elevating from anonymous token to logged-in token

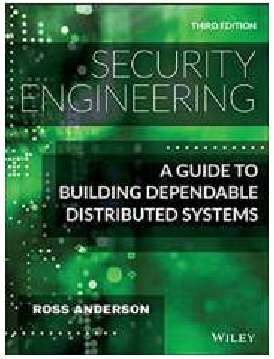
Session management tests

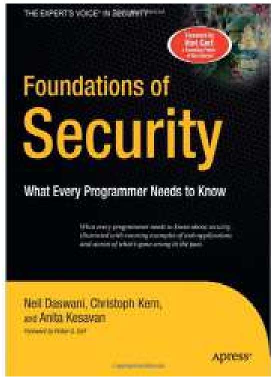
- Testing for Bypassing Session Management Schema (WSTG-SESS-01)
- Testing for Cookies attributes (WSTG-SESS-02)
- Testing for Session Fixation (WSTG-SESS-03)
- Testing for Exposed Session Variables (WSTG-SESS-04)
- Testing for logout functionality (WSTG-SESS-06)
- Test Session Timeout (WSTG-SESS-07)
- Testing for Session puzzling (WSTG-SESS-08)

To read before next lecture

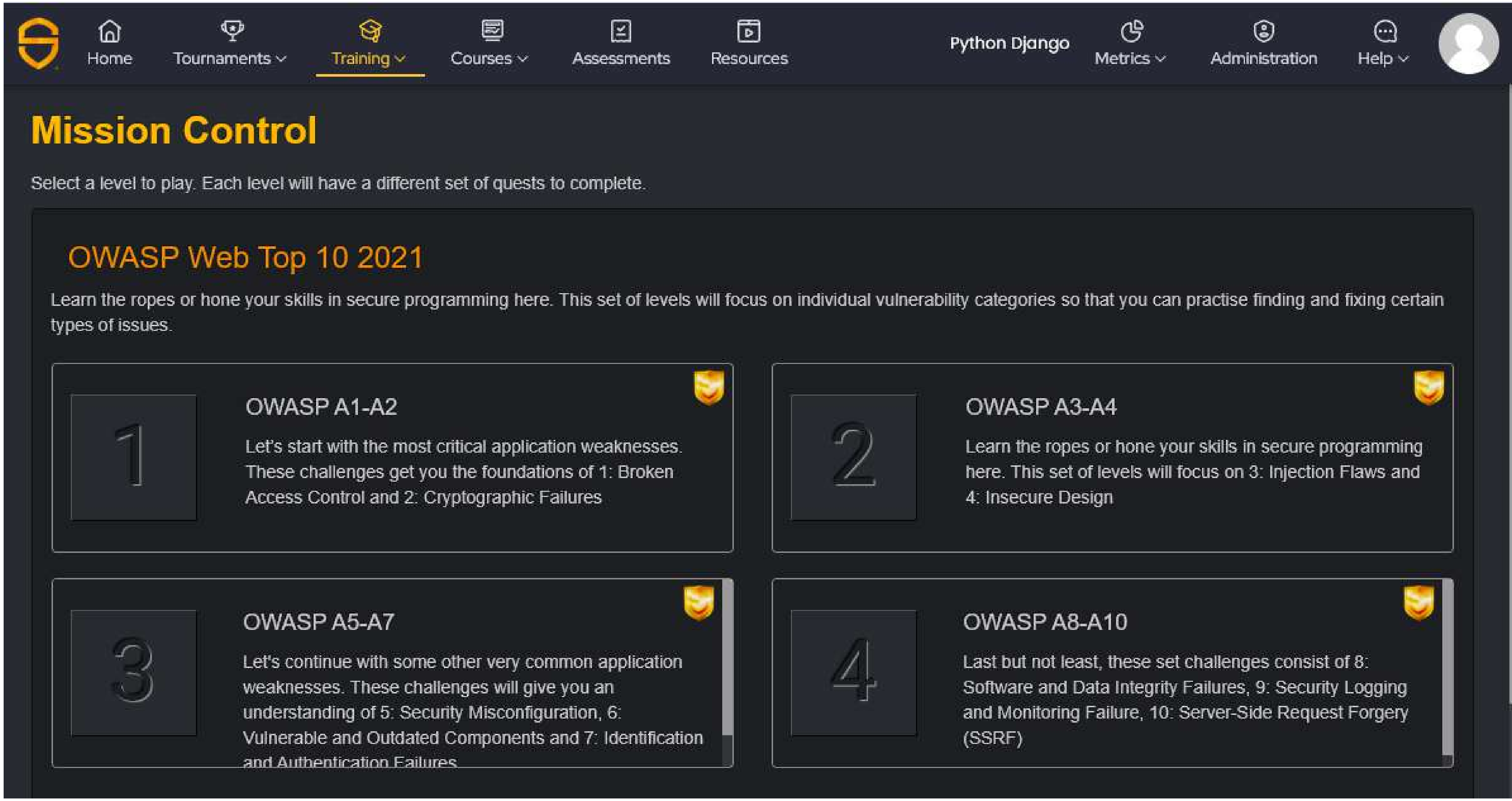
- OWASP Testing guide
  - Authentication testing
  - CSRF testing
  - XSS testing (Cross-site scripting)
  - SSRF testing (Server-side request forgery)
- Security engineering book
  - Chapter 3.4 Passwords
  - Chapter 3.5 CAPTCHAs
- (Foundations of security book)
  - Chapter 8: SQL injection
  - Chapter 9: Password security
  - Chapter 10: Cross-domain security






Foundations of Security
What Every Programmer Needs to Know
Neil Daswani, Christoph Kern, and Anita Kesavan
Apress

NTNU



Mission Control

Select a level to play. Each level will have a different set of quests to complete.

OWASP Web Top 10 2021

Learn the ropes or hone your skills in secure programming here. This set of levels will focus on individual vulnerability categories so that you can practise finding and fixing certain types of issues.

1 OWASP A1-A2

Let's start with the most critical application weaknesses. These challenges get you the foundations of 1: Broken Access Control and 2: Cryptographic Failures

2 OWASP A3-A4

Learn the ropes or hone your skills in secure programming here. This set of levels will focus on 3: Injection Flaws and 4: Insecure Design

3 OWASP A5-A7

Let's continue with some other very common application weaknesses. These challenges will give you an understanding of 5: Security Misconfiguration, 6: Vulnerable and Outdated Components and 7: Identification and Authentication Failures

4 OWASP A8-A10

Last but not least, these set challenges consist of 8: Software and Data Integrity Failures, 9: Security Logging and Monitoring Failure, 10: Server-Side Request Forgery (SSRF)