

Chapter 4

Pentesting



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

Recap

Why cybersecurity for you?

- Applicatieontwikkeling
- Systemen & netwerkbeheer
- Softwaremanagement

Cybersecurity

- Security as an attitude.
- Security is not a project, it's a process.
- Security by design.

Disclaimer

DO NOT RUN / TRY THIS IN PRODUCTION ENVIRONMENTS

Disclaimer

Kevin tried it

Don't be like Kevin

U.S. Department of Justice
United States Marshals Service

WANTED BY U.S. MARSHALS

NOTICE TO ARRESTING AGENCY: Before arrest, validate warrant through National Crime Information Center (NCIC).
United States Marshals Service NCIC entry number: (NIC/ W721460021).

NAME:MITNICK, KEVIN DAVID
AKS (S):MITNIK, KEVIN DAVID
MERRILL, BRIAN ALLEN

DESCRIPTION:


Sex:MALE
Race:WHITE
Place of Birth:VAN NUYS, CALIFORNIA
Date(s) of Birth:08/06/63; 10/18/70
Height:5'11"
Weight:190
Eyes:BLUE
Hair:BROWN
Skintone:LIGHT
Scars, Marks, Tattoos:NONE KNOWN
Social Security Number (s):550-39-5695
NCIC Fingerprint Classification: ...DOPM20PM13DIPM19PM09

ADDRESS AND LOCALE: KNOWN TO RESIDE IN THE SAN FERNANDO VALLEY AREA OF CALIFORNIA AND
LAS VEGAS, NEVADA

WANTED FOR: VIOLATION OF SUPERVISED RELEASE
ORIGINAL CHARGES: POSSESSION UNAUTHORIZED ACCESS DEVICE; COMPUTER FRAUD
Warrant issued: CENTRAL DISTRICT OF CALIFORNIA
Warrant Number: 9312-1112-0154-C

DATE WARRANT ISSUED: NOVEMBER 10, 1992

MISCELLANEOUS:



1. Enumeration

[illegible]

- Enumeration
 - Network enumeration
 - Host enumeration
 - Service enumeration
- Tools for enumeration
- Enumerating web apps

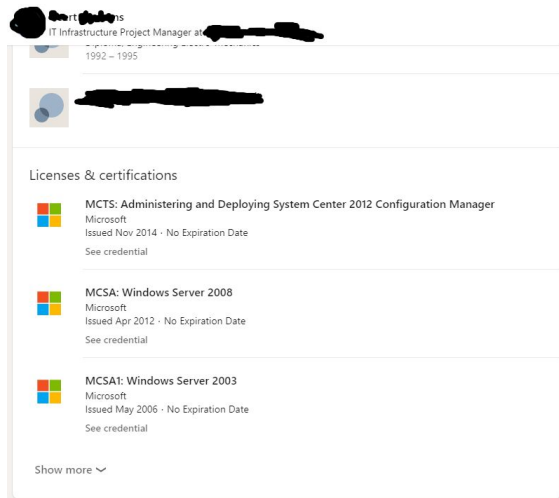
Enumeration

“a computing activity in which usernames and info on groups, shares, and services of networked computers are retrieved. ...”

- Gather information about IT systems
- Commonly used term in ‘red teaming’
- First step in penetration testing process

Enumeration

OSINT (open source intelligence) is the collection and analysis of data gathered from open sources (overt and publicly available sources)



IT Infrastructure Project Manager at [REDACTED]
1992 - 1995

Licenses & certifications

- MCTS: Administering and Deploying System Center 2012 Configuration Manager**
Microsoft
Issued Nov 2014 · No Expiration Date
See credential
- MCSA: Windows Server 2008**
Microsoft
Issued Apr 2012 · No Expiration Date
See credential
- MCSA: Windows Server 2003**
Microsoft
Issued May 2006 · No Expiration Date
See credential

Show more ▾

TOTAL RESULTS	
41,432	
TOP COUNTRIES	
	
United States	5,588
Taiwan	4,421
China	4,343
Viet Nam	3,569
Spain	1,322
TOP SERVICES	
Telnet	9,577
HTTP (8080)	4,971
8081	4,784
8083	2,382
NAS Web Interfaces	2,252
TOP ORGANIZATIONS	
Viettel Group	3,172
Chief Telecom Inc.	2,282
ASIA PACIFIC TELECO...	1,453
MCI Communications S...	599
Aliyun Computing Co., ...	565

New Service: Keep track of what you have connected to the Internet. Check out [Shodan Monitor](#)

RELATED TAGS: [router](#) [default](#) [password](#)

168.181.170.200 
168.181.170.200.spdconnect.net.br
SPEEDCONNECT SERVICOS E TECNOLOGIA LTDA - ME
Added on 2021-04-25 20:09:33 GMT
 Brazil, Seabra

HTTP/1.1 401 N/A
Server: Router Webserver
Connection: close
WWW-Authenticate: Basic realm="TP-LINK Wireless Lite N Router WR740N"
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>Login...

119.203.135.1
Korea Telecom
Added on 2021-04-25 20:13:09 GMT
 Korea, Republic of, Tangjin

C

!!!!!! WARNING !!!!!
Only authorized personnel can access this system.
The unauthorized access will be subjected to a legal penalty.
!!!!!! WARNING !!!!!
#####...

171.40.101.5 
CHINANET Hubei province network
Added on 2021-04-25 20:11:26 GMT
 China, Hangzhou

Enumeration

Network Enumeration is finding out which devices are available on the network

- What networks are available?
- Which endpoints are available on the network?
- eg. We could manually ping all the IP addresses in the subnet

Enumeration

Host Enumeration is finding out which services are running where

- IPconfig on a network already shows you a device that acts as a default gateway
- eg. we could manually check common ports on IP addresses that are linked to servers
 - 80, 8080, 443: HTTP(S)
 - 21: FTP
 - 22: SSH
 - 139, 445: SMB / Samba

Enumeration

- Host **Enumeration** is also:
 - Extracting user names using email ID's
 - Extract information using the default password
 - Brute Force Active Directory
 - Extract user names using SNMP
 - Extract user groups from Windows
 - Extract information using DNS Zone transfer

Enumeration

Service Enumeration is finding out what data, users, groups, passwords, vulnerabilities, ... are available on the host's services

- Services might be outdated and have known vulnerabilities
- Routers / Switches might use default login credentials
- FTP/ Samba services might have anonymous access
- Web applications might have hidden directories
- Web applications might use (unauthenticated) APIs
- ...

Tools for enumeration - Nmap (network mapper)

- Open source security scanner
- Tool for network exploration
- Tool for security auditing
- Designed to rapidly scan large networks
- Used for security audits
- Used for routine tasks (network inventory,...)
- Using raw ip packets
- CLI (Nmap) and GUI (Zenmap)

Don't run this in production environments or on the PXL network.

Tools for enumeration - Nmap (network mapper)

- **Hosts discovery (=network enumeration)**
- **Port scans (=host enumeration)**
- OS detection
- **Services (hosts) detection (=service enumeration)**
- Network inventory (mapping, maintenance, asset management)
- Scriptable interaction with the target
- Further information on targets; reverse DNS names, device types, MAC addresses, ...
- Generate traffic to hosts
- Find and exploit vulnerabilities in a network

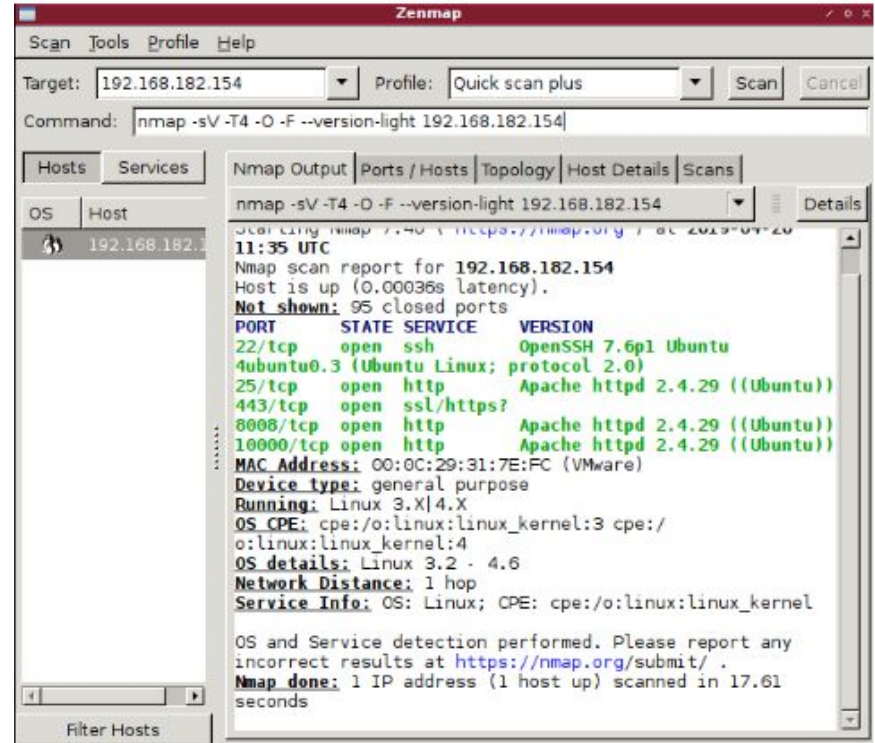
Don't run this in production environments or on the PXL network.

Tools for enumeration - Zenmap / NMap

Zenmap offers a GUI interface

- **-sV** find services & version
- **-O** OS detection
- **-F** fast/limited
- **-T4** skip timeouts
- **-p** define scan range
- **-sn** ping scan

See man page NMap



Enumerating web apps - general

- robots.txt / sitemap.xml
- Look for navigation links / documentation in source code, javascript files and paths
 - try logic & common combinations
- Try default username / password combinations
 - Often given out in service/hardware documentation
 - try logic & common combinations
- Whois - domain name
- <https://dnsdumpster.com/>

Enumerating web apps - finding files & folders

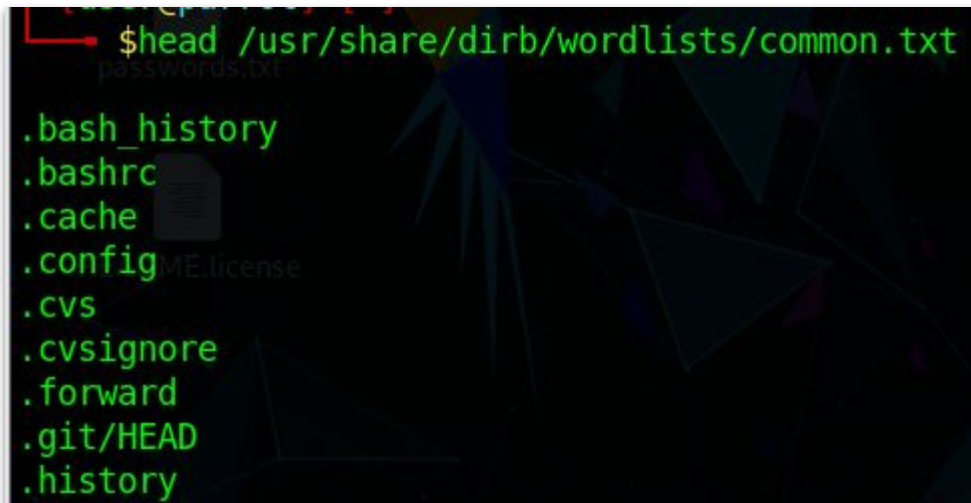
- Dirb(uster)
 - **dirb** is the command, dirbuster is the GUI application
 - comparable to **nmap** vs zenmap
 - Tool to bruteforce common paths/files in web applications
 - Find folders / files that are not meant for the public
 - Either uses bruteforce techniques or **wordlists**
 - often integrated in penetration testing operating systems
- <https://www.kali.org/tools/wordlists/>

Enumerating web apps - finding files & folders

- Basic syntax:

dirb <url_base> [<wordlist_file(s)>] [options]

- wordlist are txt files with common path and file names

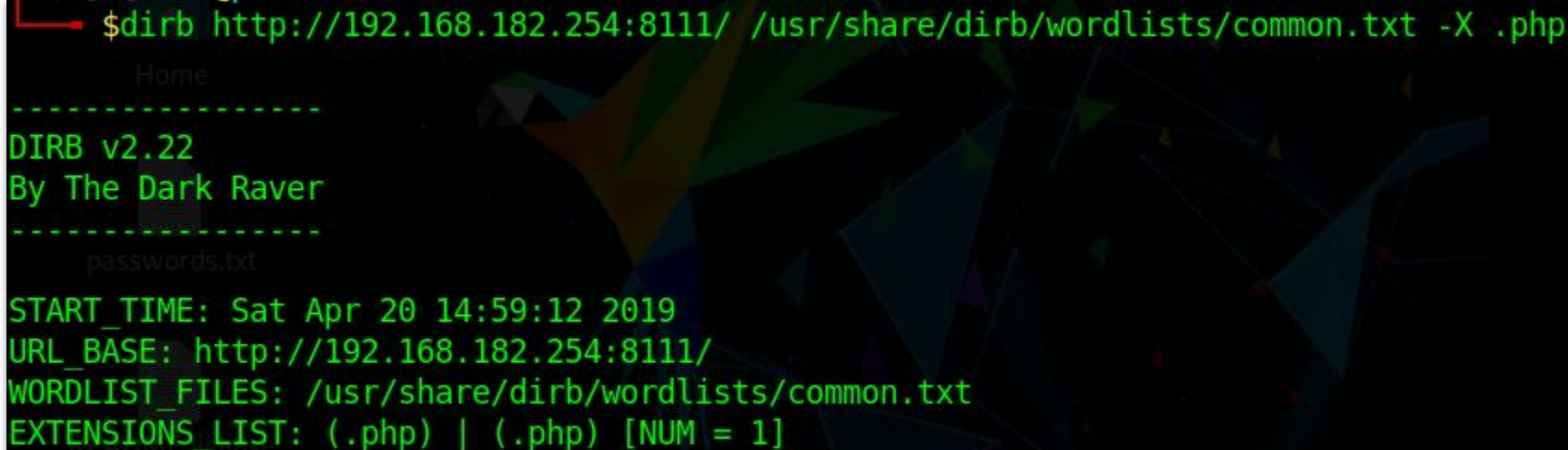
A terminal window with a dark background and green text. The command '\$head /usr/share/dirb/wordlists/common.txt' is entered. The output lists common file and directory names: .bash_history, .bashrc, .cache, .config, .cvs, .cvsignore, .forward, .git/HEAD, and .history. There is a faint, stylized graphic of a star or burst in the background of the terminal output.

```
$head /usr/share/dirb/wordlists/common.txt
.bash_history
.bashrc
.cache
.config
.cvs
.cvsignore
.forward
.git/HEAD
.history
```

Enumerating web apps - finding files & folders

dirb example syntax

- **-X** attach .php extension to wordlist entries
- no **-X** tries to find folders

A terminal window with a dark background and green text. The command '\$dirb http://192.168.182.254:8111/ /usr/share/dirb/wordlists/common.txt -X .php' is entered. The output shows 'DIRB v2.22 By The Dark Raver' followed by a separator line. Below this, it shows 'passwords.txt' and then configuration details: 'START_TIME: Sat Apr 20 14:59:12 2019', 'URL_BASE: http://192.168.182.254:8111/', 'WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt', and 'EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]'.

```
$dirb http://192.168.182.254:8111/ /usr/share/dirb/wordlists/common.txt -X .php
-----
DIRB v2.22
By The Dark Raver
-----
passwords.txt

START_TIME: Sat Apr 20 14:59:12 2019
URL_BASE: http://192.168.182.254:8111/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]
```

Enumerating web apps - finding files & folders

dirb example syntax

```
-----ftp.halifax.rwth-aachen.de/parrotsec parrot/
to all 1:2.1.6+git20190310-0kali2 [413 kB]
get:294 https://ftp.nluug.nl/os/... Progress t parrot/main
GENERATED WORDS: [ 4612x0]
get:269 https://ftp.halifax.rwth-aachen.de/parrotsec parrot/
+--+ Scanning URL: http://192.168.182.154:8111/ ----
get:272 https://ftp.halifax.rwth-aachen.de/parrotsec parrot/
+ http://192.168.182.154:8111/b.php (CODE:200|SIZE:173)
+ http://192.168.182.154:8111/c.php (CODE:200|SIZE:11) parrot/
+ http://192.168.182.154:8111/index.php (CODE:200|SIZE:218)
get:274 https://ftp.halifax.rwth-aachen.de/parrotsec parrot/
level-tools all 4.6+parrot6 [13.9 kB]
-----ftp.halifax.rwth-aachen.de/parrotsec parrot/
END_TIME: Sat Apr 20 15:05:12 2019
DOWNLOADED: 4612 - FOUND: 13 files/parrot parrot/main amd64 put
```

Enumerating web apps - inspecting app logic

- Most modern web applications use javascript based frontend (=client side) frameworks
 - often obfuscated/minified code
- Communication with backend (=server side)
 - through API endpoints

We can inspect & decompile & manipulate & ... client side code !!

Enumerating web apps - inspecting app logic



VS



CLIENT SIDE

- Frontend
- Collects user input
- Client side scripts mostly deal with visual and user input aspects
- Scripts may be restricted to run in a sandbox

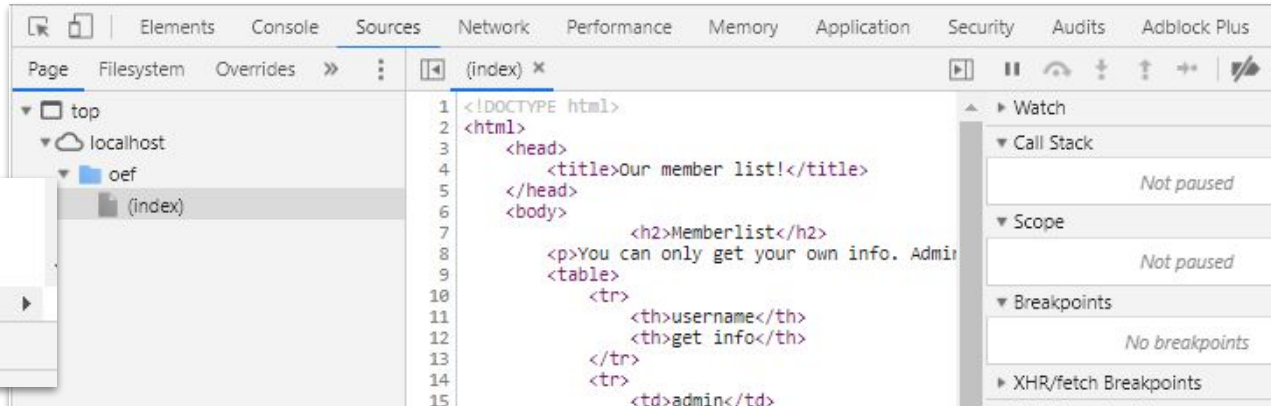
SERVER SIDE

- Backend
- Processes user input
- Server side scripts mostly deal with transactions and complex computations
- Processes are transparent to the users

Enumerating web apps - inspecting app logic

Browser development tools (F12)

- **elements/Inspector:** Provides the ability to inspect CSS and HTML code as well as edit CSS on-the-fly, seeing the effects of your changes in real time.
- **Sources/Debugger:** Lets you debug JavaScript code via a powerful graphical interface.

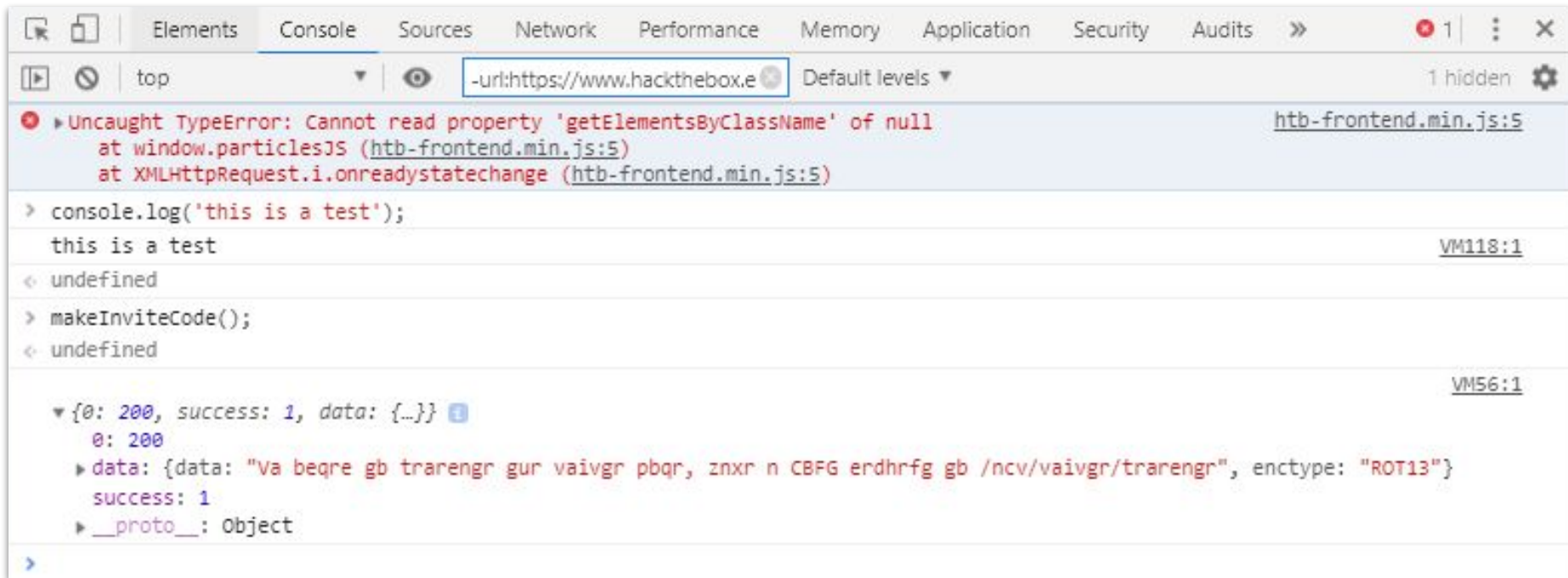


Enumerating web apps - inspecting app logic

Console: Chrome's JavaScript console allows for direct command entry as well as diagnostic debugging.

- shows console.log/error/.. messages
- Can be used to execute javascript code/functions

Enumerating web apps - inspecting app logic

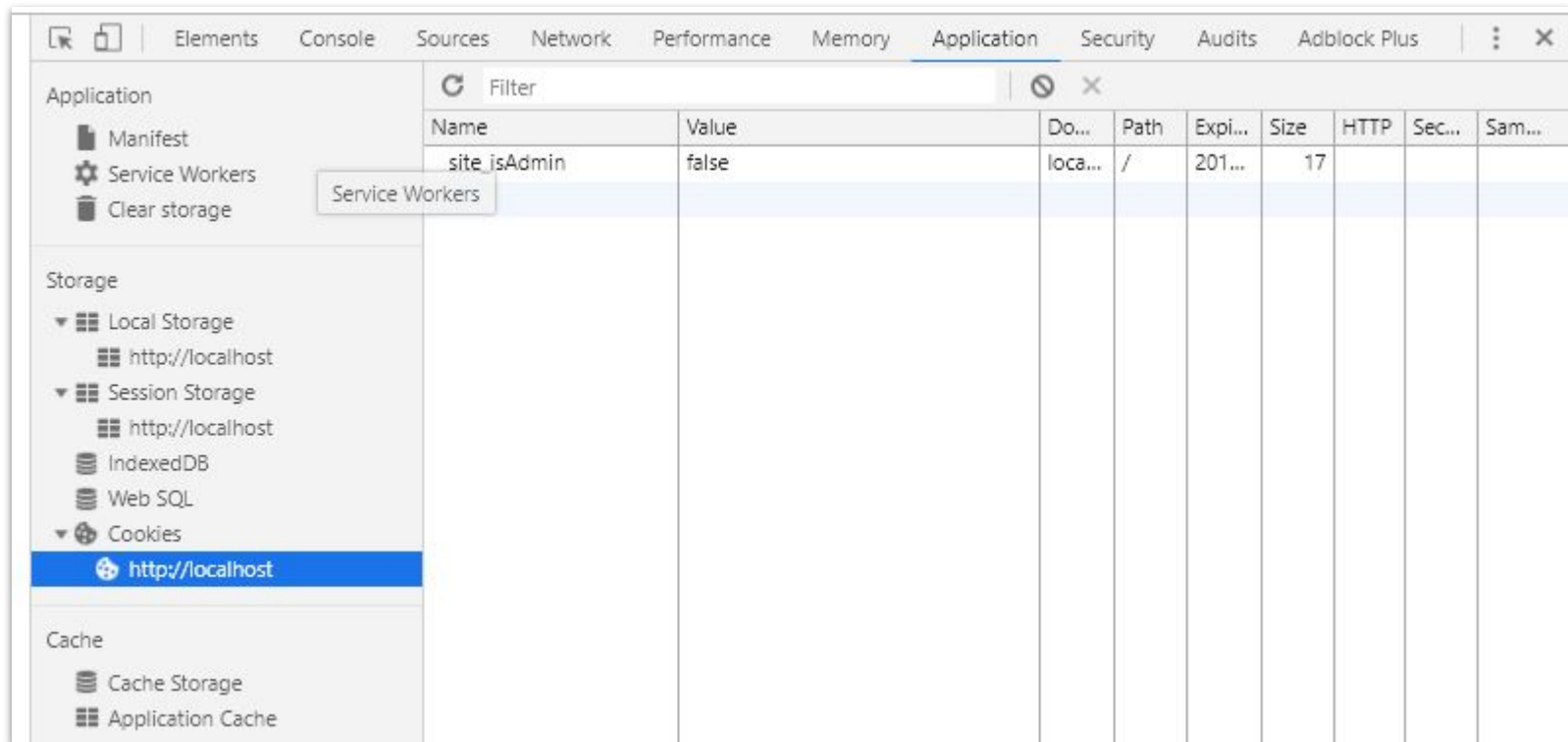


Enumerating web apps - inspecting app client side data

Storage/Application:

- contains information/data stored in the browser
- Cookies (might contain session IDs), Localstorage, cache, ...

Enumerating web apps - inspecting app client side data



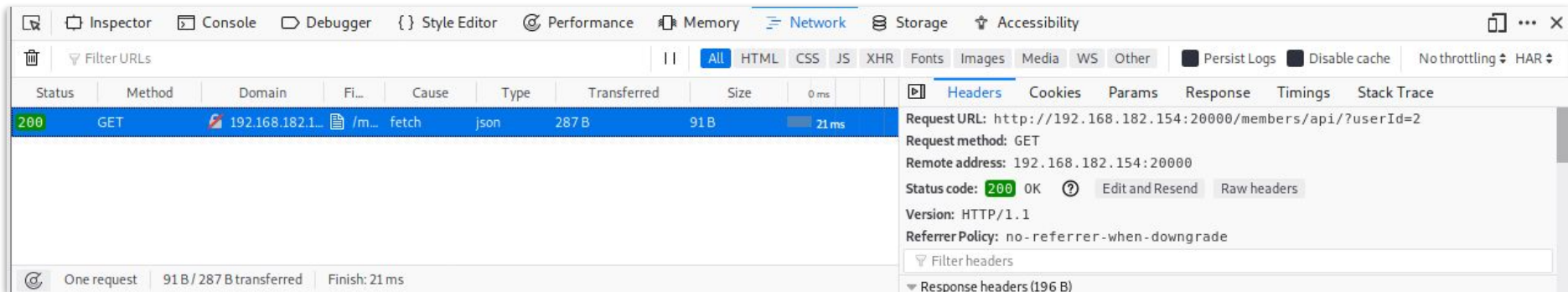
The screenshot shows the Chrome DevTools Application tab. The left sidebar contains a tree view of client-side data. Under the 'Storage' section, 'http://localhost' is selected. The main panel displays a table of application data.

Name	Value	Do...	Path	Expi...	Size	HTTP	Sec...	Sam...
site_isAdmin	false	loca...	/	201...	17			

Enumerating web apps - inspecting app communication

Network: show all http requests / communication made by the application

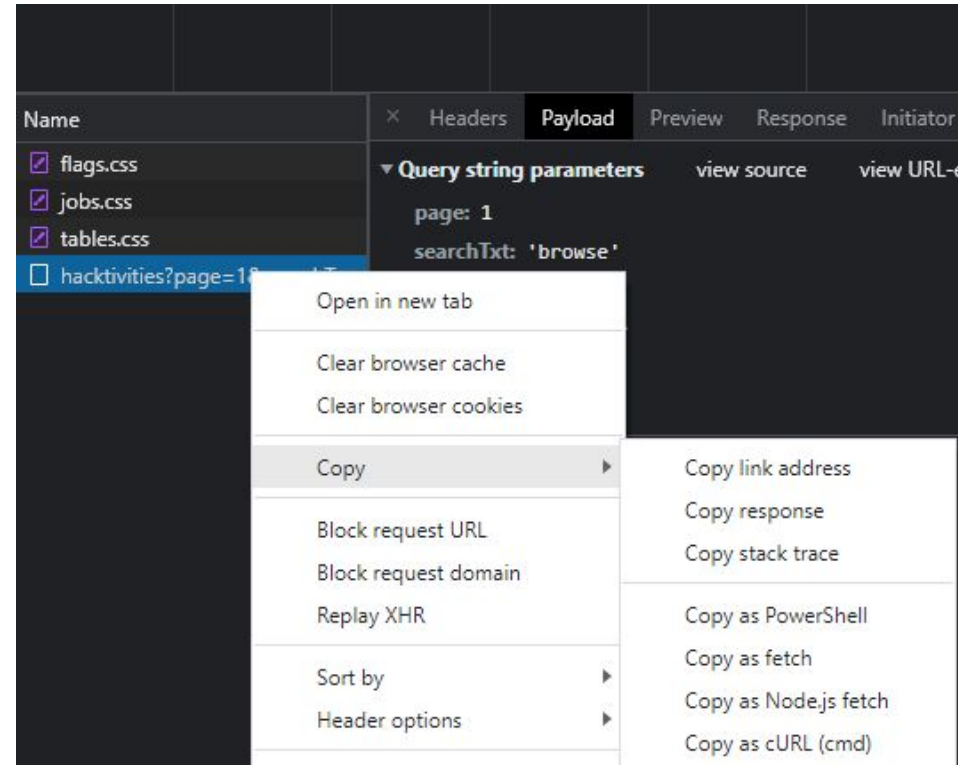
- Shows requests made by the webpage
- Useful to see if any (external) calls are being made
- Can see sent headers, params, payload, ...



Enumerating web apps - inspecting app communication

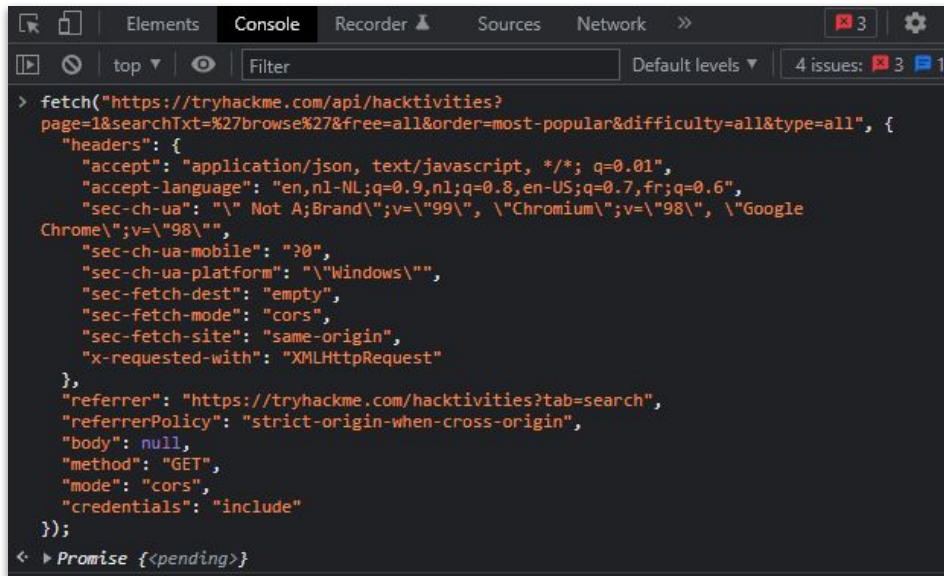
- Possible to edit and resend requests

- adjust all data such as headers, payload, params, ...
- resend using copy function
 - javascript fetch
 - powershell
 - curl
 - ...



Enumerating web apps - inspecting app communication

- We can use the copied fetch request to run it again in the console
 - while manipulating all the data
 - bypassing any client side/frontend restrictions

A screenshot of a web browser's developer console. The 'Console' tab is selected, showing a single log entry for a fetch request. The request is to 'https://tryhackme.com/api/hacktivities?page=1&searchTxt=%27browse%27&free=all&order=most-popular&difficulty=all&type=all'. The console shows the full fetch options, including headers like 'accept', 'accept-language', 'sec-ch-ua', 'sec-ch-ua-mobile', 'sec-ch-ua-platform', 'sec-fetch-dest', 'sec-fetch-mode', 'sec-fetch-site', 'x-requested-with', 'referrer', 'referrerPolicy', 'body', 'method', 'mode', and 'credentials'. The status at the bottom is 'Promise {<pending>}'.

```
> fetch("https://tryhackme.com/api/hacktivities?page=1&searchTxt=%27browse%27&free=all&order=most-popular&difficulty=all&type=all", {
  "headers": {
    "accept": "application/json, text/javascript, */*; q=0.01",
    "accept-language": "en,nl-NL;q=0.9,nl;q=0.8,en-US;q=0.7,fr;q=0.6",
    "sec-ch-ua": "\" Not A;Brand\";v=\\\"99\\\", \\\"Chromium\\\";v=\\\"98\\\", \\\"Google Chrome\\\";v=\\\"98\\\"",
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": "\"Windows\"",
    "sec-fetch-dest": "empty",
    "sec-fetch-mode": "cors",
    "sec-fetch-site": "same-origin",
    "x-requested-with": "XMLHttpRequest"
  },
  "referrer": "https://tryhackme.com/hacktivities?tab=search",
  "referrerPolicy": "strict-origin-when-cross-origin",
  "body": null,
  "method": "GET",
  "mode": "cors",
  "credentials": "include"
});
< > Promise {<pending>}
```

Enumerating web apps - postman

- Using development tools for network communication is not that user friendly

=> Postman

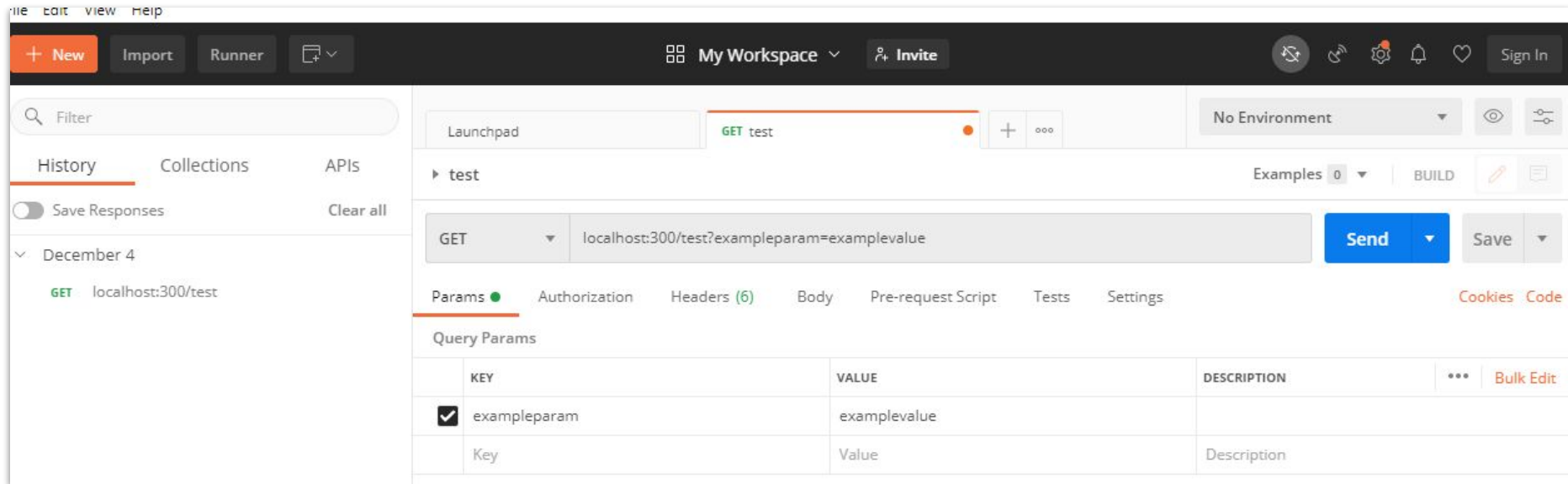
<https://www.getpostman.com/>

<https://learning.getpostman.com>

Enumerating web apps - postman

- Postman
 - A tool to perform HTTP request
 - GET, POST, DELETE, PUT, PATCH, OPTIONS
 - Provide custom arguments / parameters / headers / tests / ...
 - Used in API development & testing
 - Used in enumeration

Enumerating web apps - postman



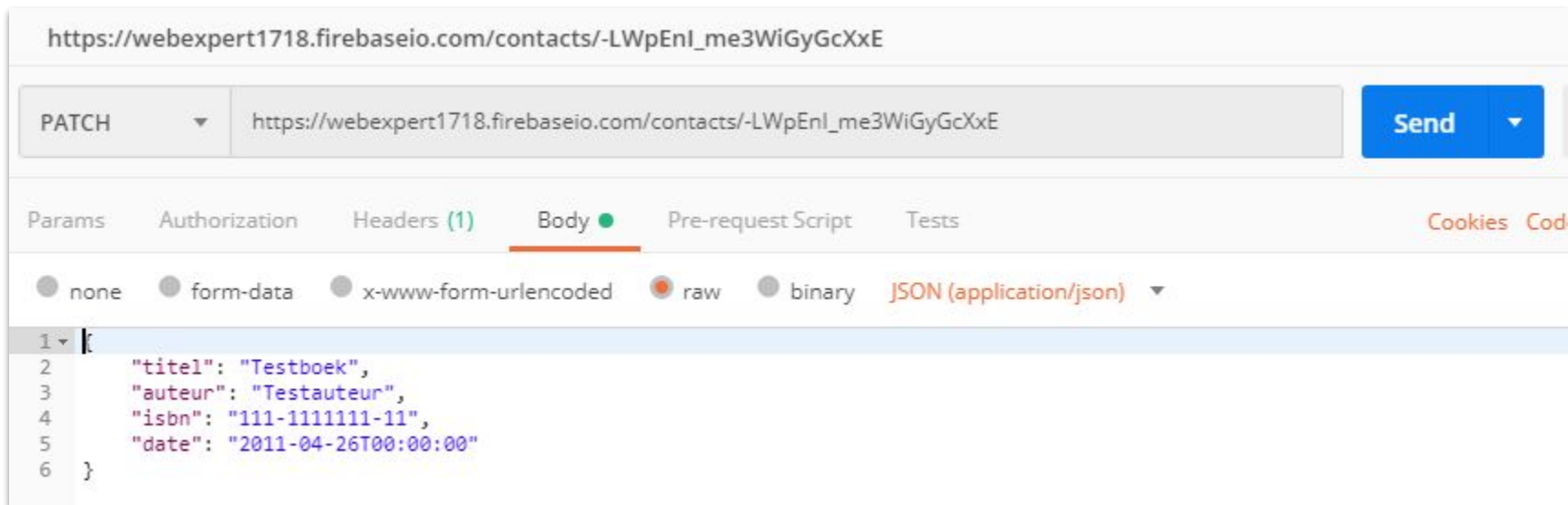
Enumerating web apps - postman

The screenshot shows the Postman interface for a POST request. The URL is `https://www.hackthebox.eu/api/v1/users/1`. The request has a single header: `Content-Type: application/json`. The response is a JSON object with the following structure:

```
{
  "0": 200,
  "success": 1,
  "data": {
    "code": "FVkkTVE1LVVgtV1RHRkEtUVdVTFo=",
    "format": "encoded"
  }
}
```

The response status is 200 OK, the time is 117 ms, and the size is 622 B.

Enumerating web apps - postman



2. Security testing of applications



Penetration testing

- attempt to pierce the armor of an organization's cyber defenses, checking for exploitable vulnerabilities in networks, web apps, devices, and user security based
 - Information gathering / enumeration
 - Exploitation
 - Privilege escalation
 - post-exploitation

Penetration testing

- Not enough technical knowledge for a deep dive at this point
- What can we do? Test (web) applications for *known* vulnerabilities / exploits

=> recap SQL injection, XSS, code injection, ...

Security driven testing of (web) applications

- "Happy Path" vs "Sad Path"
- Do not only test expected input/usage
- Test / build applications with security in mind
- How?
 - [OWASP Top 10](#)
 - Analyse framework/library documentation
 - Analyse vulnerable web applications



3. OWASP Top 10



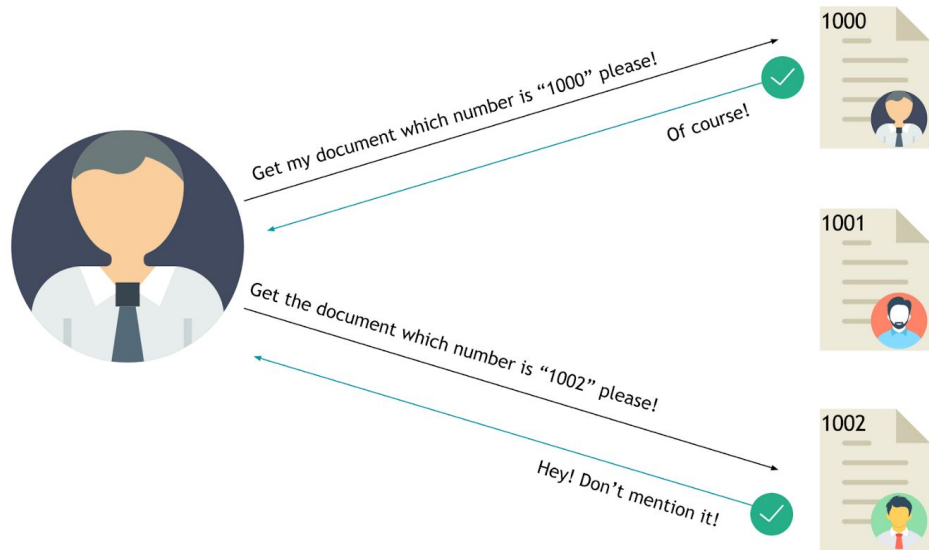
TOP10

<https://owasp.org/Top10/>

<https://blog.intigriti.com/2021/09/10/owasp-top-10/>

A01 - Broken access control

- Bypassing access control checks
 - parameter tampering
 - force browsing
 - manipulate API requests
- Elevation of privilege.
- Force browsing to authenticated pages
- Unauthenticated API endpoints / Tampering with JWT tokens.



A01 - Broken access control

Example Attack Scenarios

Scenario #1: The application uses unverified data in a SQL call that is accessing account information:

```
pstmt.setString(1, request.getParameter("acct"));  
ResultSet results = pstmt.executeQuery( );
```



An attacker simply modifies the browser's 'acct' parameter to send whatever account number they want. If not correctly verified, the attacker can access any user's account.

```
https://example.com/app/accountInfo?acct=notmyacct
```



Scenario #2: An attacker simply forces browses to target URLs. Admin rights are required for access to the admin page.

```
https://example.com/app/getappInfo  
https://example.com/app/admin_getappInfo
```

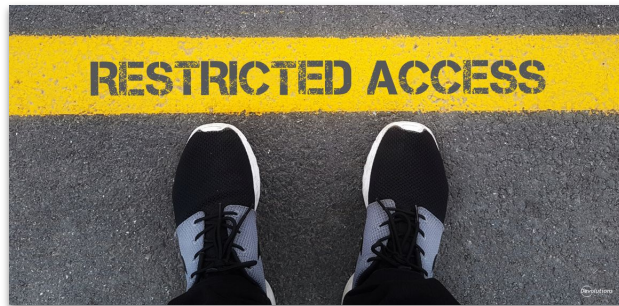


If an unauthenticated user can access either page, it's a flaw. If a non-admin can access the admin page, this is a flaw.

A01 - Broken access control

Countermeasures

- Principle of Least Privilege
- Access Control mechanisms
- Logging & Alerting
- Rate limit API and Controller access
- Authentication tokens should be invalidated (= out of scope)



A02 - Cryptographic failures

Previously named "sensitive data exposure"

- usage protocols that don't use (strong) encryption (data in transit)
 - http, ftp, older SSL versions, weak keys
- usage of old/weak encryption algorithms and/or default/easy keys
- usage of old/weak hashing algorithms
 - md5, sha1, ...

A02 - Cryptographic failures

Example Attack Scenarios

Scenario #1: An application encrypts credit card numbers in a database using automatic database encryption. However, this data is automatically decrypted when retrieved, allowing a SQL injection flaw to retrieve credit card numbers in clear text.

Scenario #2: A site doesn't use or enforce TLS for all pages or supports weak encryption. An attacker monitors network traffic (e.g., at an insecure wireless network), downgrades connections from HTTPS to HTTP, intercepts requests, and steals the user's session cookie. The attacker then replays this cookie and hijacks the user's (authenticated) session, accessing or modifying the user's private data. Instead of the above they could alter all transported data, e.g., the recipient of a money transfer.

Scenario #3: The password database uses unsalted or simple hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password database. All the unsalted hashes can be exposed with a rainbow table of pre-calculated hashes. Hashes generated by simple or fast hash functions may be cracked by GPUs, even if they were salted.

A02 - Cryptographic failures

Countermeasures

- Make sure to **encrypt all sensitive data** at rest and in transit.
- Ensure **up-to-date and strong standard algorithms, protocols, and keys** are in place; use proper key management.
 - Store passwords using strong & salted hashing functions such as Argon2, scrypt, bcrypt or PBKDF2
 - Usage of modern encryption standards
- Do not use legacy protocols such as FTP and SMTP
- Keys should be generated cryptographically randomly

https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

A03 - Injection

This category is probably the most known of all, including **SQL** injections, **command** injections, **LDAP** injection, and **XSS**.

- User-supplied data is not validated, filtered, or sanitized by the application.
- Dynamic queries or non-parameterized calls without context-aware escaping
- Hostile data is directly used or concatenated.

A03 - Injection

[check "Chapter 3 - Attacks"](#)

Example Attack Scenarios

Scenario #1: An application uses untrusted data in the construction of the following vulnerable SQL call:

```
String query = "SELECT \* FROM accounts WHERE custID='" + request.getParameter("id") +  
"''";
```

Scenario #2: Similarly, an application's blind trust in frameworks may result in queries that are still vulnerable, (e.g., Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" +  
request.getParameter("id") + "''");
```

In both cases, the attacker modifies the 'id' parameter value in their browser to send: ' UNION SLEEP(10);--. For example:

```
http://example.com/app/accountView?id=' UNION SELECT SLEEP(10);--
```

This changes the meaning of both queries to return all the records from the accounts table. More dangerous attacks could modify or delete data or even invoke stored procedures.

A03 - Injection

Countermeasures

- server-side **input validation**
- **sanitation** & escaping
- Whitelisting
 - allows only pre-approved and trusted entities, such as IP addresses, applications, or email addresses, to access a system or network, while blocking all other entities
- Never blindly trust input data
- Use **LIMIT** in SQL queries when applicable



```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection

A04 - Insecure design

New category since 2021 that focuses on risks related to design and architectural flaws

- Most abstract & overlapping with other categories
- Not focussing on secure design in sprint process
- Not using a secure development lifecycle
 - *security as an attitude*
 - *security as a process*
- **Out of scope for security essentials => next year: Security Advanced**



A04 - Insecure design

Countermeasures

- Establish and use a secure development lifecycle with AppSec professionals to help evaluate and design security and privacy-related controls
- Establish and use a library of secure design patterns or paved road ready to use components
- Use threat modeling for critical authentication, access control, business logic, and key flows
- Integrate security language and controls into user stories



A05 - Security misconfiguration

focus on incorrect/insecure configuration of applications / services / ...

- Missing security hardening
- Unnecessary features are enabled or installed
- Default user accounts are still enabled
- Error handling reveals stack traces.
- Security settings in the application servers, application frameworks, libraries, databases, etc., are not set to secure values

A05 - Security misconfiguration

Example attack scenarios

- Using default username & password combinations to login
- The application configuration shows detailed error messages leaking sensitive information or underlying flaws
- Default services ports left open, no (secure) configurations

IP camera default password list

Camera Manufacturer	Username	Password
3xLogic	admin	12345
ACTi	Admin	123456
ACTi	admin	123456
Amcrest	admin	admin
American Dynamics	admin	admin
American Dynamics	admin	9999
Arecont Vision	admin	<blank>
AvertX	admin	1234
Avigilon	admin	admin
Avigilon	administrator	<blank>
Axis	root	pass

A05 - Security misconfiguration

Countermeasures

- A **minimal platform** without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks.
- A task to review and update the configurations appropriate to all security notes, updates, and patches as part of the **patch management** process
- Disable default username / passwords combinations
- **Error handling** in the application

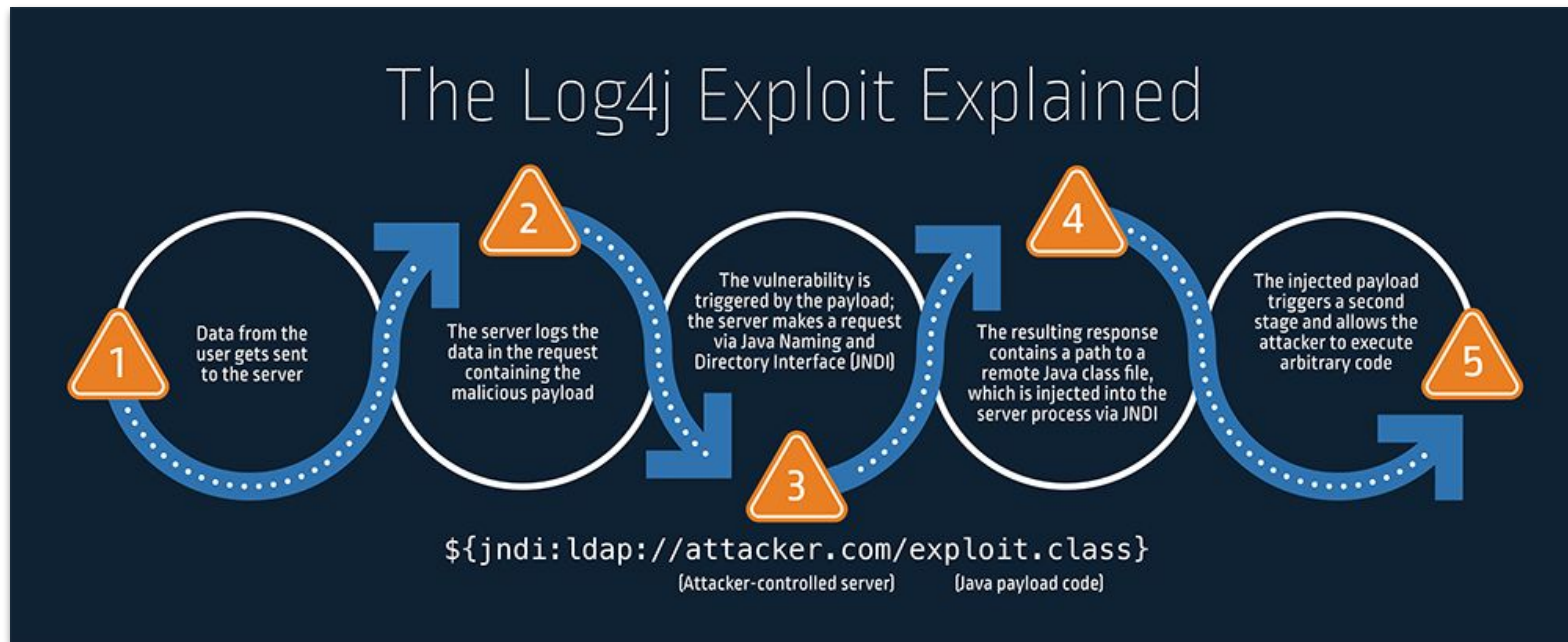
A06 - Vulnerable & outdated components

Not updating / patching older software, services, libraries, ...

- Software is vulnerable, unsupported, or out of date.
 - OS
 - Services
 - Applications & packages/libraries
- Not scanning for vulnerabilities regularly
- Not testing the compatibility of updated, upgraded, or patched libraries.

A06 - Vulnerable & outdated components

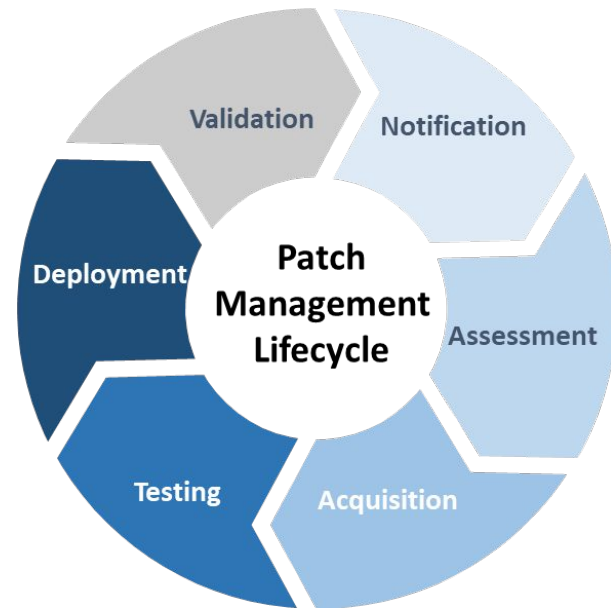
Example attack scenarios: Log4J ([CVE-2021-44228](#))



A06 - Vulnerable & outdated components

Countermeasures

- Identify & inventory used components & versions
 - **patch management**
- Vulnerability scanning
- Remove unused dependencies, unnecessary features, components, ...
- Only obtain components from official sources



A07 - Identification & Authentication failures

known as "broken authentication" in previous top 10

- Permits automated attacks such as **credential stuffing**, where the attacker has a list of valid usernames and passwords.
- Permits brute force or other automated attacks.
- Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin".
- Uses weak/ineffective credential recovery & forgot-password processes
- Uses plain text, encrypted, or weakly hashed passwords data stores

A07 - Identification & Authentication failures

Example Attack Scenarios

Scenario #1: Credential stuffing, the use of lists of known passwords, is a common attack.

Suppose an application does not implement automated threat or credential stuffing protection. In that case, the application can be used as a password oracle to determine if the credentials are valid.

Scenario #2: Most authentication attacks occur due to the continued use of passwords as a sole factor. Once considered best practices, password rotation and complexity requirements encourage users to use and reuse weak passwords. Organizations are recommended to stop these practices per NIST 800-63 and use multi-factor authentication.

Scenario #3: Application session timeouts aren't set correctly. A user uses a public computer to access an application. Instead of selecting "logout," the user simply closes the browser tab and walks away. An attacker uses the same browser an hour later, and the user is still authenticated.

A07 - Identification & Authentication failures

Countermeasures

- Implement **multi-factor authentication** to prevent automated credential stuffing, brute force, and stolen credential reuse attacks.
- Implement weak password checks & define a good password policy.
- Limit or increasingly delay failed login attempts, but be careful not to create a denial of service scenario.
- Log all failures.

A08 - Software & data integrity failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations

- Applications often include auto-update functionality, where updates are downloaded without sufficient integrity verification
- An application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks
- Insecure deserialization
- **Out of scope for security essentials => security advanced**



A08 - Software & data integrity failures

Countermeasures

- Use digital signatures or similar mechanisms to verify the software or data is from the expected source and has not been altered.
- Ensure libraries and dependencies are consuming trusted repositories.
- Ensure that unsigned or unencrypted serialized data is not sent to untrusted clients without some form of integrity check or digital signature to detect tampering



A09 - Security monitoring & logging failures

to help detect, escalate, and respond to active breaches.

- Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.

A09 - Security monitoring & logging failures

Countermeasures

- Ensure all login, access control, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts.
- Ensure that logs are generated in a format that log management solutions.
- Ensure high-value transactions have an audit trail with integrity controls.
- Establish or adopt an incident response and recovery plan.

A10 - Server side request forgery (SSRF)

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL

- As modern web applications provide end users with convenient features, fetching a URL becomes a common scenario.
- **Out of scope for security essentials => security advanced**



end