positive hack camp

pt

Hacking web applications

positive education

CyberED

positive hack camp

Nikolay Mikryukov

F6 AppSec
Web
Mobile
Pentest
Smart-Contracts
Reverse
Telegram: @Nmikryukov

positive education

CyberED

positive hack camp

pt

Pavel Sorokin

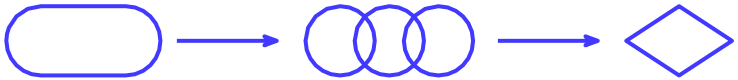
Red Team:

- Infosec.ru
- BiZone

Blue Team:

- Yandex
- Ozon

Current: CTO at CyberEd



positive education

CyberED

positive hack camp

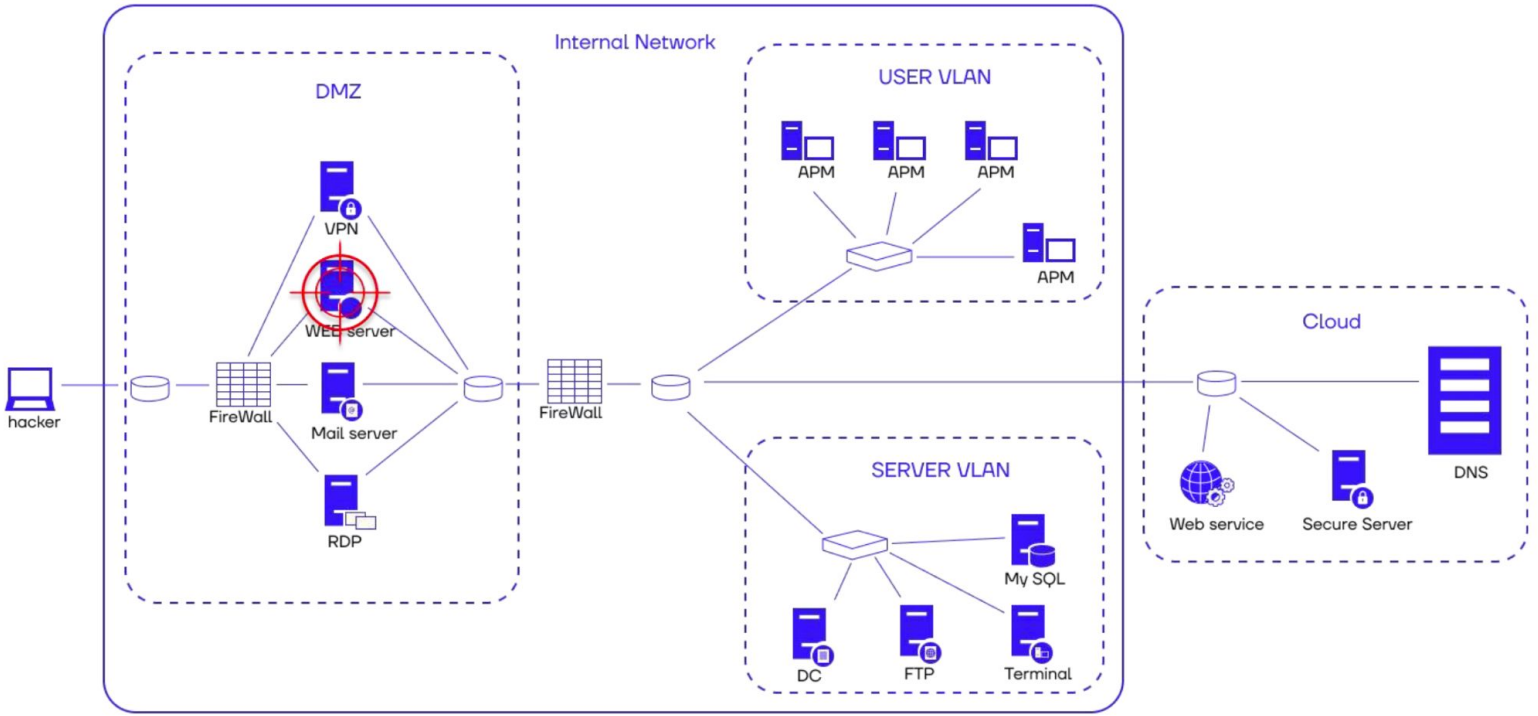
Agenda

- HTTP
- BurpSuite
- Web vulnerabilities
- OS command injection
- Access control
- Authentication
- SQL injections

positive education

CyberED

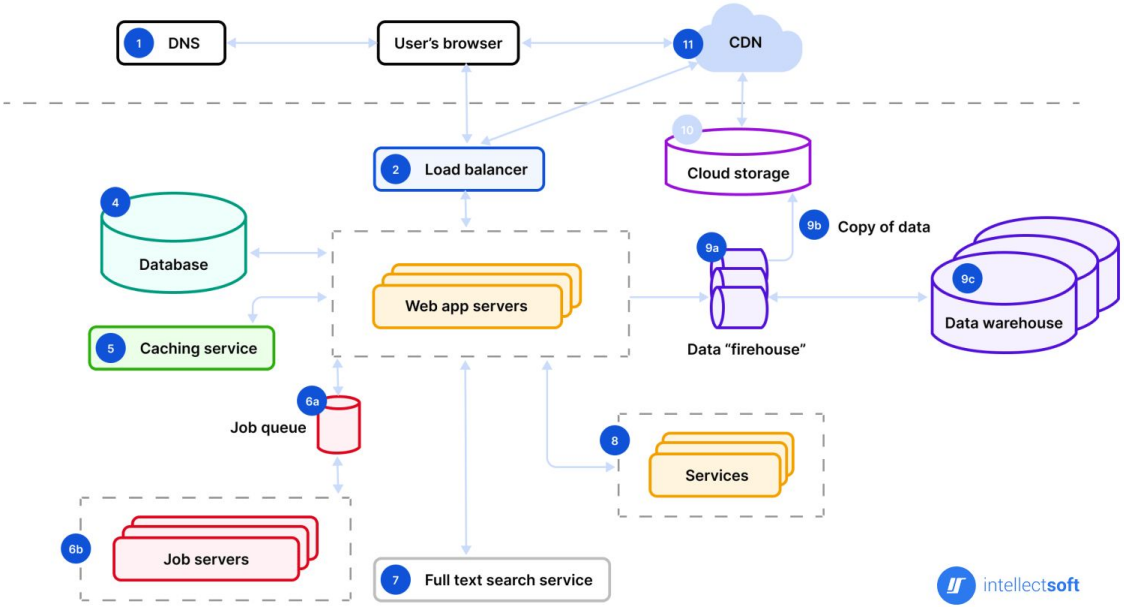
positive hack camp



positive education

CyberED

Web applications



positive hack camp

pt

Web applications

This is a special kind of network services that proceed browser requests made via HTTP protocol

positive education

CyberED

positive hack camp

pt

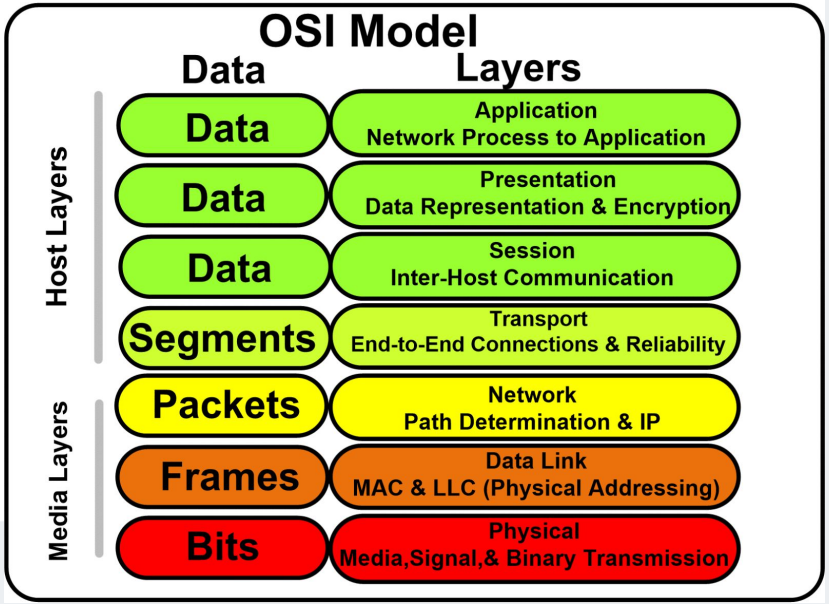
HTTP

positive education

CyberED

HTTP

- 7 level of OSI model - Application
- Works over TCP
- Client-Server protocol
- HTTP/1.1 and HTTP/2 are common
- HTTP/1.1 is human readable



positive hack camp

HTTP Request Example

positive education

CyberED

positive hack camp

pt

HTTP response

positive hack camp

pt

HTTP methods (verbs)

- GET - read
- POST - write or add
- PUT - add
- DELETE - delete
- HEAD, OPTIONS, TRACE, PATCH…
- Custom verbs

positive education

CyberED

positive hack camp

pt

HTTP status code

- 1xx - info (102 Processing)
- 2xx - success (200 OK)
- 3xx - redirect (301 Moved Permanently)
- 4xx - client error (404 Not Found)
- 5xx - server error (500 Internal Server Error)

positive education

CyberED

positive hack camp

pt

HTTP/1.1 is a text protocol

<NETCAT DEMONSTRATION>

positive education

CyberED

positive hack camp

pt

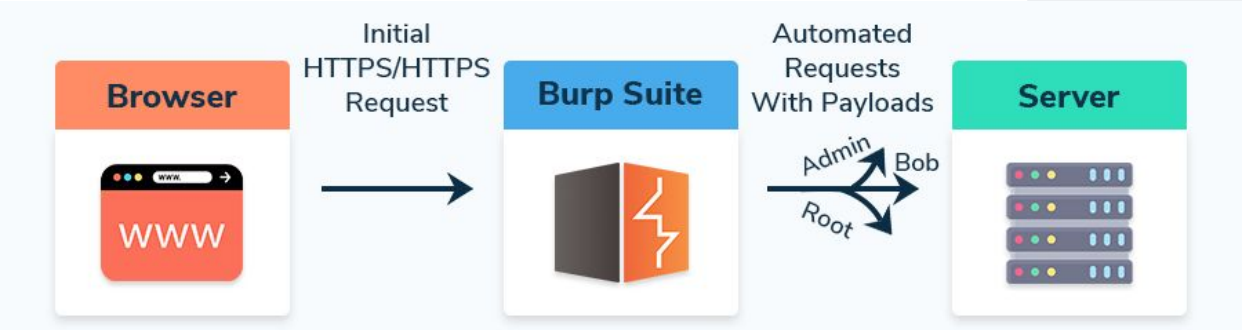
Burp Suite

positive education

CyberED

Burp Suite

- Examine traffic
- Resend arbitrary requests
- Modify any HTTP fields
- Carry out brute force attacks



positive hack camp

pt

Burp Suite

Burp Suite main modules:

- Proxy
- Repeater
- Intruder

https://portswigger.net/burp/documentation/desktop

positive education

CyberED

positive hack camp

pt

Burp Suite

Burp Suite demo

labs.cyber-ed.ru - “Burping” task

positive education

CyberED

positive hack camp

pt

Web Applications Vulnerabilities

positive education

CyberED

positive hack camp

Vulnerabilities in web applications only occur when the developer **has made a mistake.**
Such bugs form a whole group of typical vulnerabilities, i.e. bugs that **systematically occur** in certain mechanisms implemented in web applications.

Examples of such mechanisms are:

- Authentication
- Authorization
- Session Management
- File Storage
- Working with databases
- Media generation (video and images)
- User data output

With knowledge of typical web application vulnerabilities, it is possible to effectively seek out and exploit the opportunities presented by the application to achieve the goals of the audit.

Most of the typical vulnerabilities can be found in projects such as:

OWASP TOP 10 (20**)

The current version is the 2021 version

owasp.org/www-project-top-ten/

PortSwigger Academy

Open source learning platform focused in the direction of web application security

portswigger.net/web-security/learning-path

Let's take a look at the typical vulnerabilities present in these platforms. and talk about which ones you'll encounter more often than others.

Example of vulnerabilities

→ Server-side

- SQL injection
- Business logic vulnerabilities
- File upload vulnerabilities
- Authentication
- Information disclosure
- Command injection
- Directory traversal
- Access control
- Server-side request forgery (SSRF)
- XXE injection

Example of vulnerabilities

→ On the client side

- Cross-site scripting (XSS)
- Cross-site request forgery (CSRF)
- Cross-origin resource sharing (CORS)
- Clickjacking
- DOM-based vulnerabilities
- WebSockets

positive hack camp

pt

We will look at **several vulnerabilities that** occur in the initial stages of finding opportunities to compromise a web application and in the final stages when full compromise becomes possible.

In the initial stages we are faced with mechanisms: **authentication, access control.**

In the final stages, the greatest achievement in "demonstrating the ability to compromise" is demonstrating the ability to **execute arbitrary code** or OS commands on behalf of a web application server.

positive education

CyberED

OS commands injection

positive education

CyberED

by positive technologies

powered by digital ministry_

Injections of OS commands

What are command injections?

OS command injection (also known as shell injection) is a web application vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server running the application, and typically enables the application and all of its data to be completely compromised.

→ Very often, an attacker can exploit an OS command injection vulnerability to compromise other parts of the infrastructure, using trust relationships to redirect the attack to other systems in the organization.

How do command injections arise?

An example of a command injection vulnerability:

```
$command = 'convert -pointsize 72 label:Hello ' .
$_FILES['userfile']['name'];

system($command);
```

Example of exploitation for vulnerability above:

```
------WebKitFormBoundaryePkpFF7tjBAqx29L
Content-Disposition: form-data; name="userfile"; filename="check | ls -la;"

Content-Type: application/x-object


... contents of file goes here ...
```

Often, exploitation of such vulnerabilities is difficult and requires searching for opportunities to execute code through other "contexts" and technologies.
For example, command execution is possible in a number of other vulnerabilities that need to be found and exploited before being able to execute an arbitrary command.

These could be:

- **Insecure deserialization** vulnerabilities;
- Vulnerabilities of **template injection** on the SSTI server side;
- **SQL injection** vulnerabilities;
- **Buffer** / heap / stack **overflow** vulnerabilities;

As well as multiple instances of implementation issues and bugs encountered with process startup and terminal shell operations.

Supplementary materials

Let's consider an example of such a problem:

“OS Command Injection” lab on labs.cyber-ed.ru

positive hack camp

pt

Access control

positive education

CyberED

Access control vulnerabilities

Access control (or authorization) is the application of restrictions on who (or what) can attempt actions or access requested resources.

- Identification
- Authentication
- Authorization

positive hack camp

pt

Access control vulnerabilities

Broken access control is a common and often **critical security vulnerability.** Access control design decisions **must be made by people,** not technology, and the potential for error is high.

positive education

CyberED

Access control vulnerabilities

From the user's perspective, access control can be categorized as follows:

- Vertical access control
- Horizontal access control
- Context-sensitive access control

Vertical access control

Vertical access controls are mechanisms that restrict access to sensitive functions that are not available to other types of users.

- User
- Moderator
- Admin

Example of IFLAC vulnerabilities

Insecure Functional Level Access Control (IFLAC) - can allow attackers to gain access to functions that are unauthorized for the attacker's role.

- Publish comments (all users)
- Edit comments (moderators)
- Delete users (admins)

positive hack camp

pt

HackerOne example

Open source learning platform focused in the direction of web application security

hackerone.com/reports/502593

positive education

CyberED

positive hack camp

pt

Horizontal access control

Horizontal access controls are mechanisms that restrict access to resources to users who are specifically authorized to access those resources.

transferMoney(**user_id_from**, user_id_to, amount)

positive education

CyberED

positive hack camp

pt

Example of IDOR vulnerabilities

Insecure Direct Object Reference (IDOR) occurs when an application uses user input to directly access objects, and an attacker can modify the input to gain unauthorized access.

positive education

CyberED

positive hack camp

pt

- Consider a site that uses the following URL to access a customer account page by retrieving information from an internal database:

  https://bank.com/view_profile?user_id=1337

positive education

CyberED

Examples from HackerOne:

IDOR in DoD

hackerone.com/reports/1004745

Access control in locations
Business logic dependent

Context-sensitive access controls **prevent a user from performing actions in the wrong order.** For example, a retail website may prevent a user from changing the contents of a shopping cart after they have made a payment.

IDOR practice

→ Insecure Direct Object Reference lab on labs.cyber-ed.ru

positive hack camp

pt

Authentication

positive education

CyberED

positive hack camp

pt

Content:

- What is Authentication?
- The difference between authentication, authorization, and identification?
- How do authentication vulnerabilities arise?
- What is the damage caused by authentication vulnerabilities?
- Vulnerabilities in authentication mechanisms
  - Vulnerabilities in Password-based authentication
  - Vulnerable defense against brute-force attacks
  - Basic HTTP authentication
  - Multifactor authentication vulnerabilities
  - Vulnerabilities of Password Change, Password Recovery, Session Maintenance mechanisms
- How do you protect yourself against authentication mechanism vulnerabilities?

positive education

CyberED

Authentication

This is the process of verifying the identity of a **particular** user or client for subsequent access to a particular resource. In turn, there are such elements of information protection as identification and authorization. They are identical to authentication, they can be a part of it. At the identification stage, information about the user is recognized, such as login and password or a unique identifier. In the authorization process, the user's capabilities are checked, access rights are determined and granted.

→ **Difference between authentication, authorization and identification**

Identification

allows us to determine what kind of object is in front of us by assigning an identifier to subjects and objects.

Authentication

is the process of checking whether a user is really who they say they are, while authorization involves checking whether the user is allowed to do something.

Multi-factor authentication

In this method, identity verification is accomplished using several criteria.

The different types of authentication can be classified according to three factors:

- **Knowledge,** such as a password or the answer to a security question, a one-time code. These are sometimes referred to as "knowledge factors".
- **Possession,** that is, a physical object such as a cell phone or a security token – a magnetic card. These are sometimes referred to as 'possession factors'.
- **Biometrics** such as personal data or patterns of behavior are sensitive information. They are sometimes referred to as "consistency factors".

Authentication mechanisms rely on a range of technologies to verify one or more of these factors

How do authentication vulnerabilities arise?

Broadly speaking, most vulnerabilities in authentication mechanisms arise in one of two ways:

- **Authentication mechanisms are** vulnerable because they cannot adequately protect against brute force attacks.
- **Logical errors** or bad code in the implementation allow an attacker to completely bypass the authentication mechanisms.

Vulnerabilities in Password-based authentication

For websites that use a password-based login process, users either register themselves for an account or an administrator assigns them an account. This account is associated with a unique username and secret password that the user enters on the login form for authentication.

In this scenario, the **mere fact that they know the secret password is taken as sufficient proof of the** user's identity.

Consequently, the security of the site would be **compromised** if an attacker could either obtain or guess the credentials of another user.

Vulnerabilities in Password-based authentication

Vulnerabilities in this mechanism **arise for various reasons,** some of them are:

- Possible **brute-force attacks**
- **Vulnerable defense** against brute-force attacks

Brute-force attacks

A brute force attack is when an attacker uses a trial and error system in an attempt to guess valid user credentials. These attacks are usually automated using username and password dictionaries.

Automating this process, especially with the use of special tools, potentially **allows an attacker to make a huge number** of login **attempts at** a high rate.

Brute-forcing passwords

Passwords can similarly be brute-force, depending on the complexity of the password. Many sites adopt some form of password policy that forces users to create high-entropy passwords that, at least theoretically harder to crack with a single brute force.

This usually includes enforcing usage in passwords:

- Minimum number of characters
- Mixtures of lowercase and uppercase letters
- At least one special character

Brute-forcing passwords

- **High-entropy passwords are** difficult for computers to crack, but we can use basic knowledge of **human behavior** to exploit vulnerabilities that users unwittingly introduce into this system.
- Users often take a password that **they can remember,** and try to use it in accordance with the password policy.
- For example, if no **password** is allowed, users can try something like **P@ssw0rd** or **P4$$w0rd!**
- Knowing predictable patterns helps brute force attacks often be far **more sophisticated and effective** than simply iterating through all sorts of character combinations.

Multifactor authentication vulnerabilities

Many websites rely solely on single-factor authentication using a password to authenticate users. However, some require users to prove their identity using multiple authentication factors.

Biometric factor verification is impractical for most web sites. However, **mandatory and optional two-factor authentication (2FA) based on what "you know" and "what you have" is becoming more common**. This typically requires users to enter both a traditional password and a temporary verification code from a third-party physical device in their possession.

Multifactor authentication vulnerabilities

Bypassing two-factor authentication is possible for a variety of reasons:

- Sometimes the **implementation of** two-factor authentication is so flawed that it can be **bypassed entirely.**
- Sometimes the **faulty logic of** two-factor authentication means that after a user has performed the initial login step, the website cannot adequately verify that the same user performs the second step.
- Also, as with passwords, websites must take measures to **prevent the 2FA verification code from being brute-forced.** This is especially important because the code is often a simple 4 or 6 digit number. Without adequate protection, cracking such a code is trivial.

Vulnerable defense against brute-force attacks

It is highly likely that a brute force attack will involve multiple failed attempts before the attacker successfully compromises the account. Logically, brute-force defense is an attempt to slow down the rate of password automation at which an attacker can attempt to log in.

The two most common ways to prevent brute-force attacks are as follows:

- **Lock the account that a** remote user is trying to access if they make too many unsuccessful login attempts.
- **Block a** remote user's **IP address** if they make too many login attempts.

Supplementary materials

You can practice the exercises on open source platforms.
One of the best platforms to learn web application security issues:

PortSwigger Academy

positive education

CyberED

Practice

- Authentication Bypass lab on labs.cyber-ed.ru

positive hack camp

pt

SQL Injections

positive education

CyberED

positive hack camp

pt

What is SQL?

Structured Query Language (SQL)

is a programming language to store and process information in a relational database. SQL instructions can be used to store, update, select rows.

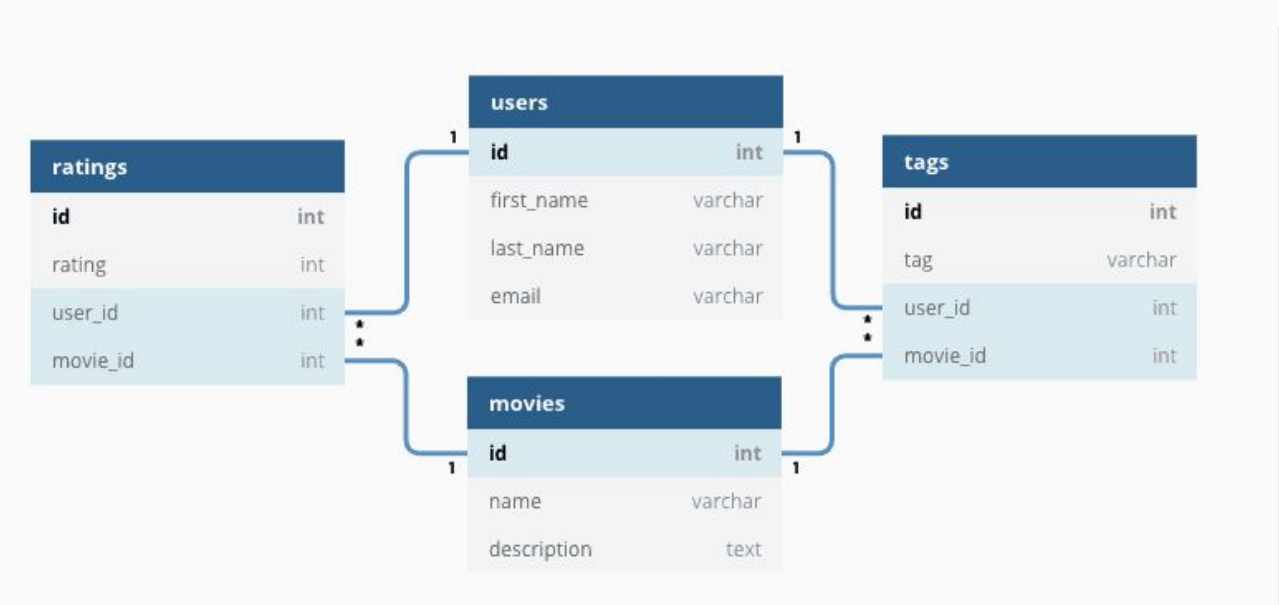
positive education

CyberED

positive hack camp

pt

Relational database



positive education

CyberED

positive hack camp

pt

SQL Demonstration

https://sqliteonline.com

```
SELECT id, name FROM demo;
INSERT INTO demo(id, name, hint) VALUES (27, 'Test', 'Hint text');
UPDATE demo SET Name='Changed' where id=27;
DELETE from demo where id=27;
```

positive education

CyberED

positive hack camp

pt

SQL tutorials

https://www.w3schools.com/sql/

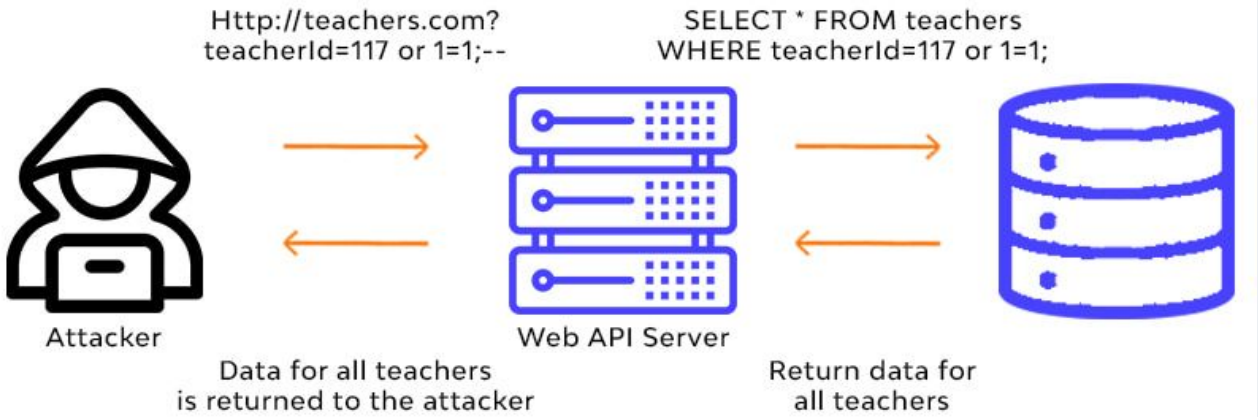
https://www.sqltutorial.org/

positive education

CyberED

What is SQL injection?

SQL injection is a web application vulnerability that allows an attacker to interfere with queries that an application makes to its database. It allows an attacker to access data in the database.



positive hack camp

pt

SQLi code sample

```php
$mysqli = new mysqli(/*...*/);

$username = $_POST['username'];
$password = $_POST['password'];

$query = "SELECT * FROM user WHERE user.username = '$username' AND user.password = '$password'";
$result = $mysqli->query($query);
```

positive education

CyberED

positive hack camp

pt

What SQL injection vulnerabilities can lead to:

- Data extraction and database research capabilities
- Modification of information in the database
- Circumventing logic
- Bypassing authorization mechanisms and authentication
- Reading or writing OS files
- Executing OS commands
- Denial of Service

positive education

CyberED

positive hack camp

pt

Examples of SQL injections

Scenario #1

The application uses untrusted data when creating the next vulnerable SQL call:

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

positive education

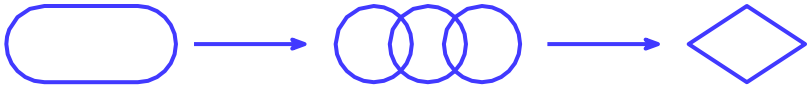
CyberED

In both cases.

the attacker changes the value of the the value of the parameter 'id' to send **' or '1'='1.**

For example:
http://example.com/app/accountView?id='+or+'1'='1

Modifying both queries allows you to retrieve all records from the credentials table of the credentials table.
More serious attacks allow modification or deletion.



About attack techniques

There are a wide range of vulnerabilities, attacks, and SQL injection techniques that occur in different situations.

Some of the most common examples of SQL injections include:

- SQL Injection Bypass
- Stacked queries
- union-based
- Error-based
- Boolean blind
- Time-base
- Out of band

Supplementary materials

- PayloadsAllTheThings - SQL injection
  PayloadsAllTheThings - NoSQL injection
- OWASP NoSQL injection slides
- PortSwigger - SQL injection cheat sheet
- SQLMap - Automation Tool

positive hack camp

pt

Let's practice attacking on SQL injection:

“SQL-injection Bypass” lab
“SQL-injection Union-Based” lab
on labs.cyber-ed.ru

positive education

CyberED

positive hack camp

“SQL-injection Bypass” lab
on labs.cyber-ed.ru

positive education

CyberED

positive hack camp

pt

What SQL injection vulnerabilities can lead to:

- Data extraction and database research capabilities
- Modification of information in the database
- Circumventing logic
- Bypassing authorization mechanisms and authentication
- Reading or writing OS files
- Executing OS commands
- Denial of Service

positive education

CyberED

positive hack camp

```
SELECT 1, 'hello', password FROM users
```

No comma ,

```
SELECT * FROM (SELECT 1) AS a JOIN (SELECT 'hello') AS b
  JOIN (SELECT password FROM users) AS c
```

No quotes

```
SELECT * FROM (SELECT 1) AS a JOIN (SELECT 0x68656c6c6f) AS b
  JOIN (SELECT password FROM users) AS c
```

No SQL words
In lower/uppercase

```
SelEcT * FRom (SeLeCt 1) As a JOin (seLECt 0x68656c6c6f) As b
  JoiN (SElECt password frOM users) As c
```

No spaces

```
SelEcT/**/*/**/FRom/**/(SeLeCt/**/1)/**/aS/**/a/**/JOin/**/
(seLECt/**/0x68656c6c6f)/**/As/**/b/**/
JoiN/**/(SElECt/**/password/**/frOM/**/users)/**/AS/**/c
```

positive education

CyberED

positive hack camp

Same queries

```
MariaDB [chat]> SELECT 1, 'hello', password FROM users;
+---+-------+-------------------+
| 1 | hello | password          |
+---+-------+-------------------+
| 1 | hello | admin_s3cr3t_p4ss |
| 1 | hello | testpass          |
+---+-------+-------------------+
2 rows in set (0.00 sec)

MariaDB [chat]> SelEcT/**/*/**/FRom/**/(SeLeCt/**/1)/**/aS/**/a/**/JOin/**/
    -> (seLECt/**/0x68656c6c6f)/**/As/**/b/**/
    -> JoiN/**/(SElECt/**/password/**/frOM/**/users)/**/AS/**/c;
+---+--------------+-------------------+
| 1 | 0x68656c6c6f | password          |
+---+--------------+-------------------+
| 1 | hello        | admin_s3cr3t_p4ss |
| 1 | hello        | testpass          |
+---+--------------+-------------------+
2 rows in set (0.00 sec)
```

positive education

CyberED

Examples of SQL injections

Scenario #1

The application uses untrusted data when creating the next vulnerable SQL call:

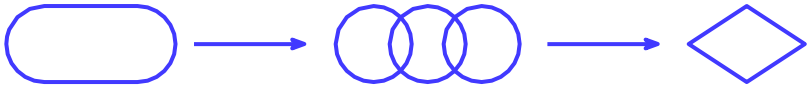
```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") + "'";
```

In both cases.

the attacker changes the value of the the value of the parameter 'id' to send **' or '1'='1.**

For example:
http://example.com/app/accountView?id='+or+'1'='1

Modifying both queries allows you to retrieve all records from the credentials table of the credentials table.
More serious attacks allow modification or deletion.



positive education

CyberED

About attack techniques

There are a wide range of vulnerabilities, attacks, and SQL injection techniques that occur in different situations.

Some of the most common examples of SQL injections include:

- SQL Injection Bypass
- Stacked queries
- union-based
- Error-based
- Boolean blind
- Time-base
- Out of band

UNION-based SQL injections

In case the web application displays data retrieved from the database, an attacker can use the UNION operator to read information from other tables.

Example: http://books.com/?id=1 UNION SELECT username, password FROM users

SELECT author, title FROM books WHERE id = 1 UNION SELECT username, password FROM users

UNION-based SQL injections

In practice, an attacker does not know the names of tables and columns stored in the database. However, fortunately for the attacker, database management systems (DBMS) usually contain metadata tables that store this information.

For MySQL, PostgreSQL and MSSQL an attacker can use:

- information_schema.tables — to retrieve the list of tables
- information_schema.columns — to retrieve column names

Supplementary materials

- PayloadsAllTheThings - SQL injection
  PayloadsAllTheThings - NoSQL injection
- OWASP NoSQL injection slides
- PortSwigger - SQL injection cheat sheet
- SQLMap - Automation Tool

positive hack camp

pt

Let's practice attacking on SQL injection:

“SQL-injection Union-Based” lab
on labs.cyber-ed.ru

positive education

CyberED