Міністерство освіти і науки України Харківський національний університет радіоелектроніки

Факультет Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра Інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий(магістерський)

Дослідження методів захисту веб-сервісів від кібератак
(тема)

Виконав:
Студент2 курсу,групи ІМІзм-20-2

Коваленко О.Г
(прізвище,ініціали)

Спеціальність172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник доцент Золотарьов В.А
(посада,прізвище,ініціали)

Допускається до захисту
Зав.кафедри

(підпис) (прізвище,ініціали)

2022р.

Не містить відомостей заборонених до відкритого публікування

Студент /Коваленко О.Г./

Керівник / Золотарьов В.А./

Харківський національний університет
радіоелектроніки

Факультет Навчально-науковий центр заочної форми навчання

Кафедра Інформаційно-мережної інженерії

Рівень вищої освіти другий(магістерський)

Спеціальність172 Телекомунікації та радіотехніка
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійнаабоосвітньо-наукова)

Освітня програма Інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав.кафедри

(підпис)

«____»_____2022 р.

ЗАВДАННЯ
НАКВАЛІФІКАЦІЙНУРОБОТУ

Студентові Коваленку Олександру Геннадійовичу

(прізвище,ім'я,по батькові)

1. Тема роботи Дослідження методів захисту веб-сервісів від кібератак

Затверджена наказом університету від 25 березня 2022р. №34

2. Термін подання студентом роботи до екзаменаційної комісії20 травня 2022 р.

3. Вихідні дані дороботи Об'єкт дослідження;веб-сервісів.Дослідити найактуальніші ризики веб-сервсів;з'ясувати чинники загроз та вектори атак:визначити слабкі місця безпеки;запропонувати засоби контролю безпеки;визначити можливі технічні наслідки вдалої кібератаки.Визначити та описати типи кібератак,спрямованих на на веб-сервіси;мета атаки;джерело загрози;вразливості веб-сервісів;які застосовуються при кібератаці;об'єкт атаки;опис атаки;наслідки атаки.Запропонувати методи захисту веб-сервісів для кожної кібератаки.Розробити лабораторну роботу «Конфігурація веб-сервісів».

4. Перелік питань,що опрацювати в роботі;Перелік умовних скорочень.Вступ.1.Аналіз вразливостей веб-сервісів.2.Кібератаки на веб-сервіси.3.Захист веб-сервісів.Висновки. Перелік посилань. Додаток А. Методичні вказівки до розробленої лабораторної роботи «Конфігурація веб-сервісів». Додаток Б. Слайди презентації.
5. Перелік графічного матеріалу із зазначенням комп'ютерних ілюстрацій (слайдів) Слайди у форматі PowerPoint: мета роботи; вразливості веб-сервісів за методологією OWASP; Види кібератак на веб-сервіси, Методи захисту від кібератак. Алгоритм виконання лабораторної роботи. Висновки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ.	26.03.2022	
2	Аналіз завдання та літературних джерел.	01.04.2022	
3	Написання першого розділу	08.04.2022.	
4	Написання другого розділу	15.04.2022	
5	Написання третього розділу	28.04.2022	
6	Написання додатку А	08.05.2022	
7	Написання вступу та висновків	11.05.2022	
8	Оформлення презентаційного матеріалу та підготовка до захисту у ДЕК	12.05.2022	

Дата видачі завдання 25 березня 2022 р.

Студент _____ Коваленко О.Г.

(підпис) (прізвище, ініціали)

Керівник роботи _____ к.т.н., доц. Золотарьов В.А.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 93 с., 17 рис., 13 посилань, 2 додатки.

Мета роботи – дослідження найактуальніших загроз для веб-сервісів а також способів атак на них та захисту.

Використання незахищених веб-сервісів у рази збільшує ризики несанкціонованого доступу до корпоративних ресурсів та порушення працездатності веб-сайтів, що веде до репутаційних, а також грошових втрат компанії. Число компаній, які застосовують веб-технології для підвищення продуктивності роботи і залучення нових клієнтів, зростає з кожним роком. Безсумнівно, інтернет-сервіси несуть з собою безліч переваг, але є й зворотна сторона медалі – з ростом числа додатків збільшується і кількість кіберзагроз. Тому важливо забезпечити надійний та ефективний захист веб-сервісів найякіснішим підходом до інформаційної безпеки що буде розглянуто в моїй роботі.

ВЕБ-СЕРВІС, КІБЕРАТАКА, ВРАЗЛИВІСТЬ, ЗАХИСТ

ABSTRACT

Explanatory note: 93 pp., 17 figs., 13 references, 2 appendix.

The purpose of the work is to study the most current threats to web services, as well as ways to attack and protect them.

The use of unsecured web services increases the risk of unauthorized access to corporate resources and disruption of websites, which leads to reputational and monetary losses of the company. The number of companies that use web technologies to increase productivity and attract new customers is growing every year. Undoubtedly, Internet services have many advantages, but there is a flip side to the coin - as the number of applications increases, so does the number of cyber threats. Therefore, it is important to ensure reliable and effective protection of web services with the highest quality approach to information security, which will be considered in my work.

WEB SERVICE, CYBER ATTACK, VULNERABILITY, PROTECTION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HTML - стандартна мова розмітки документів для перегляду веб-сторінок в браузері.

API - набір визначених підпрограм для створення програмного забезпечення.

POST - один з багатьох методів запиту, що підтримуються протоколом передачі даних HTTP, який використовується у всесвітній мережі Інтернет.

PUT - служити зміни чи вставки ресурсу, у вимогі зміни має бути заданий унікальний ID вказаного ресурсу.

DELETE – потрібен для щоб видалити ресурс структури об'єкта, треба вказати ID головного об'єкта.

JWJ - це відкритий стандарт для створення токенів доступу на основі JSON.

Cookie - невеликий фрагмент даних, надісланий веб-сервером та збережений на комп'ютері користувача.

CORS - технологія сучасних браузерів, яка дозволяє надати веб-сторінкам доступ до ресурсів іншого домену.

SMTP - це широко використовуваний мережевий протокол, призначений передачі електронної пошти.

FTP – протокол для передачі данни в мережі.

TLS – протокол для захисту данних.

ОС – операційна система.

SQL - декларативна мова програмування, що застосовується для створення, модифікації та управління даними в реляційній базі даних, що керується відповідною системою управління базами даних.

XML - стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет.

URL - стандартизована адреса певного ресурсу в мережі.

TCP - один із основних протоколів передачі даних інтернету.

IP - протокол мережевого рівня для передавання датаграм між мережами.

CI/CD - це комбінація безперервної інтеграції та безперервного розгортання програмного забезпечення у процесі розробки.

DNS - комп'ютерна розподільна система для отримання інформації про домени.

VPN - узагальнена назва технологій, що дозволяють забезпечити одне або кілька мережевих з'єднань поверх іншої мережі, наприклад, Інтернет.

Wi-Fi – стандарт для передавання цифрових потоків даних по радіоканалах.

MAC - це унікальний ідентифікатор, що зіставляється з різними типами устаткування для комп'ютерних мереж.

ВСТУП

Сучасні інформаційні послуги стають дедалі зручнішими, інтерактивнішими та клієнтоорієнтованішими і все частіше надаються віддалено за допомогою мережі Інтернет через звичайний веб-інтерфейс. Такі прості сервіси, як електронна пошта, замовлення піци, покупка будь-яких речей або такі складні, як дистанційне банківське обслуговування, надання державних послуг, сьогодні все доступне через мережу. Це дуже зручно, але це дуже небезпечно.

Особливо враховуючи те, що можливість веб-доступу можуть мати не лише фронтофіси інформаційних систем (ІС) роздрібних компаній, а й критичні бізнес-системи, а послуги надаються не лише в сегменті b2c, а й b2b. Тому атака на веб-сервіси – одна з найнебезпечніших і найпоширеніших загроз, здатна викликати фінансові або репутаційні втрати для власника сервісу, але й призвести до фінансових втрат його клієнтів. Щорічні прямі збитки від атак на веб-ресурси, за різними даними, становлять від 1 до 3,5 трильйона доларів США. Непрямі збитки та втрати, пов'язані зі збитком репутації, втраченим прибутком, викраденням даних кредитних карток, персональних даних тощо не піддаються оцінці.

Веб-сервіси часто розробляють студії та люди, які не мають жодного уявлення про інформаційну безпеку, а рішення, які вони використовують, не дозволяють ні забезпечити захист самої системи, ні розмежувати периметри веб- та внутрішніх інформаційних систем. Інша зустрічається крайність - економія фінансів і часу на розробку веб-додатків як не основного продукту при впровадженні ІТ-систем, що сприяє зниженню рівня безпеки навіть спочатку надійних платформ і, часто викликає появу унікальних уразливостей.

1. АНАЛІЗ УРАЗЛИВОСТЕЙ ВЕБ-СЕРВІСІВ

Аналіз проведено за методикою OWASP (розшифровується як,Open Web Application Security Project) – це некомерційний фонд, який працює над підвищенням безпеки програмного забезпечення,веб-застосунків а також документацію,різні інструменти та технології[1].

OWASP Топ-10.2021 – це список із десяти найпоширеніших насьогодні вразливостей веб-застосункам.

Насьогодні найпоширенішими вразливостями є:

A01:2021-Broken Access Control **(**Порушений контроль доступу**);**

A02:2021-Cryptographic Failures (Криптографічні збої);

A03:2021-Injection **(**Ін'єкція**);**

A04:2021-Insecure Design **(**Небезпечний дизайн**);**

A05:2021-Security Misconfiguration (Помилка конфігурації безпеки);

A06:2021-Vulnerable and Outdated Components(Уразливі та застарілі компоненти**);**

A07:2021-Identification and Authentication Failures (Помилки ідентифікації та аутентифікації);

A08:2021-Software and Data Integrity Failures (Помилки програмного забезпечення та цілісності даних);

A09:2021-Security Logging and Monitoring Failures (Регистрація та моніторинг збоїв безпеки);

A10:2021-Server-Side Request Forgery **(**Підробка запиту на стороні сервера**).**

1.1 A0h1:2021-Broken Access Control **(**Порушений контроль доступу**)**

1.1.1 Огляд

Піднявшись з п’ятої позиції, 94% додатків були перевірені на певну форму порушеного контролю доступу із середнім рівнем захворюваності 3,81%, і мають найбільшу кількість випадків у наданому наборі даних – понад 318 тисяч. Серед відомих загальних перерахувань слабких місць (CWE) включені CWE-200: розкриття конфіденційної інформації неавторизованому суб’єкту, CWE-201: вставка конфіденційної інформації в надіслані дані та CWE-352: підробка міжсайтових запитів.

1.1.2 Опис

Контроль доступу запроваджує політику таким чином, що користувачі не можуть діяти за межами передбачених ними дозволів. Збої зазвичай призводять до несанкціонованого розкриття інформації, зміни або знищення всіх даних або виконання бізнес-функції за межами обмежень користувача. Серед поширених уразливостей контролю доступу.

Порушення принципу найменших привілеїв або заборони за замовчуванням, коли доступ має надаватися лише для певних можливостей, ролей або користувачів, але доступний кожному.Обхід перевірок контролю доступу шляхом зміни URL-адреси (змінення параметрів або примусового перегляду), внутрішнього стану програми або сторінки HTML, або за допомогою інструменту атаки, який змінює запити API.Дозвіл перегляду або редагування чужого облікового запису шляхом надання його унікального ідентифікатора (небезпечні прямі посилання на об’єкт).Доступ до API з відсутніми елементами керування доступом для POST, PUT та DELETE.Підвищення привілею.

Дія як користувач без входу в систему або як адміністратор під час входу як користувач.

Маніпуляції з метаданими, як-от повторне відтворення або втручання в маркер доступу JSON Web Token (JWT), файл cookie або приховане поле, яким маніпулюють для підвищення привілеїв або зловживання визнанням JWT недійсним.Неправильна конфігурація CORS дозволяє отримати доступ до API з неавторизованих/ненадійних джерел.Примусово переглядати автентифіковані сторінки як неавтентифікований користувач або привілейовані сторінки як звичайний користувач.Порушений контроль доступу може бути використаний як дуже складними атаками, так і дуже простими. Такі атаки можуть варіюватися від збору облікових даних користувачів за допомогою спеціальних інструментів, таких як Mimikatz (забезпечення бічного переміщення в скомпрометованій мережі), до простого експериментування й маніпулювання URL-адресами. По суті, порушення контролю доступу виникають, коли користувачі, які не мають права доступу до файлів або функцій, все одно можуть отримати до них доступ.

Контроль доступу часто структурований для роботи як «привратник». Неавторизовані користувачі залишаються поза межами, поки вони не отримають належну авторизацію (автентифікацію), за допомогою таких засобів, як ім’я користувача та пароль. Проблема в тому, що дизайн програми рідко враховує, коли зловмисники отримують цю авторизацію[2].

Як тільки хтось може «увійти», часто залишається незначний контроль доступу або його взагалі немає, щоб запобігти збиранню даних, фальсифікації записів або доступу до привілейованих функцій.У сукупності ці фактори створюють дуже складний ризик для безпеки. Порушений контроль доступу важко помітити заздалегідь, його ще важче виявити під час постійного порушення; і може мати надзвичайно далекосяжні та дорогі наслідки (таб.1.1).

Таблиця 1.1 - A0h1:2022-Broken Access Control **(**Порушений контроль доступу**)**

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник який є авторизованим користувачем,просто змінює URL або параметри привілейованої функції.	Додатки незавжди належним чином захищяють свої функції. Інколи захист на функціональному рівні здійснюється за рахунок конфігурації,а система налаштовується невірно.		Такі недоліки дають зловмисникам можливість отримати доступ до незахищеної функції.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

1.2A02:2021-Cryptographic Failures (Криптографічні збої)

1.2.1 Огляд

Зміщення на одну позицію вгору до №2, раніше відомого як «викриття чутливих даних», що є більш широким симптомом, а не першопричиною, акцент робиться на збоях, пов’язаних із криптографією (або її відсутності). Що часто призводить до розкриття конфіденційних даних. Серед відомих загальних перерахувань слабких місць (CWE) включені CWE-259: використання жорстко закодованого пароля, CWE-327: зламаний або ризикований криптоалгоритм і CWE-331 недостатня ентропія.

1.2.2 Опис

Перш за все, необхідно визначити потреби в захисті даних під час передачі та спокою.

Наприклад, паролі, номери кредитних карток, медичні картки, особиста інформація та ділові таємниці потребують додаткового захисту (таблиця 1.2), в основному, якщо ці дані підпадають під дію законів про конфіденційність, наприклад, Загального регламенту ЄС про захист даних (GDPR) або правил, наприклад, про захист фінансових даних. наприклад, PCI Data Security Standard (PCI DSS). Для всіх таких даних.Чи передаються будь-які дані відкритим текстом? Це стосується таких протоколів, як HTTP, SMTP, FTP, які також використовують оновлення TLS.

Зовнішній інтернет-трафік є небезпечним. Перевірти весь внутрішній трафік, наприклад, між балансувальниками навантаження, веб-серверами або внутрішніми системами.Чи використовуються якісь старі чи слабкі криптографічні алгоритми чи протоколи за замовчуванням або в старішому коді.Чи використовуються криптоключі за замовчуванням, слабкі криптоключі генеруються або використовуються повторно, чи відсутнє належне керування ключами чи ротація? Чи перевіряються криптоключі в сховищах вихідного коду?Чи не застосовується шифрування, наприклад, чи відсутні директиви безпеки заголовків HTTP (браузера) або заголовки?.

Чи правильно перевірено отриманий сертифікат сервера та ланцюжок довіри?. Чи вектори ініціалізації ігноруються, використовуються повторно чи не створюються достатньо безпечними для криптографічного режиму роботи?. Чи використовується небезпечний режим роботи?. Чи використовується шифрування, коли автентифіковане шифрування є більш доречним?Чи використовуються паролі як криптографічні ключі за відсутності функції отримання базового ключа пароля?. Чи використовується випадковість для криптографічних цілей, яка не була розроблена для задоволення криптографічних вимог?.Навіть якщо вибрано правильну функцію, чи потрібно її розширювати розробнику, а якщо ні, чи переписав розробник вбудовану в неї функціональність сильного посіву, яка не має достатньої ентропії/непередбачуваності?

Чи використовуються застарілі хеш-функції, такі як MD5 або SHA1, чи використовуються некриптографічні хеш-функції, коли потрібні криптографічні хеш-функції ?

Чи використовуються застарілі методи криптографічного заповнення, такі як PKCS № 1 v1.5?.Чи можна використовувати повідомлення про криптографічні помилки або інформацію стороннього каналу, наприклад, у формі атак оракула заповнення[3].

Таблиця 1.2 A02:2022-Cryptographic Failures (Криптографічні збої)

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник може використати слабкий або застарілий алгоритм та фреймворків додатку.	Слабкі криптографічні алгоритми, недостатня ентропія,застарілі протоколи, використання власних алгоритмів.		Такі недоліки дають зловмисникам можливість отримати доступ до данних.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

1.3A03:2021-Injection **(**Ін'єкція**)**

1.3.1 Огляд

Ін’єкція ковзає вниз до третьої позиції. 94% додатків були перевірені на певну форму ін’єкції з максимальним рівнем захворюваності 19%, середнім рівнем захворюваності 3% і 274 тисячі випадків. Серед відомих перерахувань загальних слабкостей (CWE) включені CWE-79: міжсайтові сценарії, CWE-89: ін’єкція SQL і CWE-73: зовнішній контроль імені файлу або шляху.

1.3.2 Опис

Програма вразлива для атаки, коли данні, надані користувачем, не перевіряються, не фільтруються та не очищаються програмою.Динамічні запити або непараметризовані виклики без екранування з урахуванням контексту використовуються безпосередньо в інтерпретаторі.Ворожі дані використовуються в параметрах пошуку об’єктно-реляційного відображення (ORM) для вилучення додаткових конфіденційних записів.Ворожі дані використовуються безпосередньо або об’єднуються. SQL або команда містить структуру та шкідливі дані в динамічних запитах, командах або збережених процедурах.Деякі з найпоширеніших ін’єкцій – це SQL, NoSQL, команда ОС, об’єктно-реляційне відображення (ORM), LDAP і мова виразів (EL) або ін’єкція бібліотеки навігації графів об’єктів (OGNL). Концепція ідентична серед усіх тлумачів. Огляд вихідного коду є найкращим методом виявлення, чи є програми вразливими до ін’єкцій[4].

Автоматичне тестування всіх параметрів, заголовків, URL-адрес, файлів cookie, даних JSON, SOAP і XML настійно рекомендується. Організації можуть включити статичні, динамічні та інтерактивні інструменти тестування безпеки додатків у конвеєр, щоб виявити недоліки ін’єкції перед розгортанням у виробництві.

Найчастіше ін’єкція використовується для зловмисного доступу до даних користувача, таких як дані особи або пароль облікового запису; але ін'єкційні атаки також здатні безпосередньо виконувати команди ОС або вводити новий код в систему(таблиця 1.3). Це може надати адміністративний доступ новим неавторизованим сторонам або заборонити доступ легальним користувачам.

Таблиця 1.3-A03:2021-Injection **(**Ін'єкція**)**

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник може впровадити дані в веб-додаток і змусити виконати дії які не були розроблені для програми.	Відсутність перевірки та очищення данних, використання коду через SQl-запит що використовує ненадійні дані.		Такі недоліки дають зловмисникам можливість крадіжки конфіденційної інформації та данних.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

1.4A04:2021-Insecure Design **(**Небезпечний дизайн**)**

1.4.1 Огляд

Нова категорія на 2021 рік зосереджена на ризиках, пов’язаних із недоліками дизайну та архітектури, із закликом до більшого використання моделювання загроз, безпечних шаблонів проектування та еталонних архітектур. Як спільнота, ми повинні вийти за межі «Shift-Left» у просторі кодування до попереднього кодування діяльності, яка є критичною для принципів Secure by Design. Помітні загальні перерахування слабких місць (CWE) включають CWE-209: генерування повідомлення про помилку, що містить конфіденційну інформацію, CWE-256: незахищене зберігання облікових даних, CWE-501: порушення межі довіри та CWE-522: недостатньо захищені облікові дані.

1.4.2 Опис

Небезпечний дизайн — це широка категорія, яка представляє різні недоліки, виражені як «відсутній або неефективний дизайн контролю». Небезпечний дизайн не є джерелом для всіх інших 10 категорій ризику.

Існує різниця між небезпечним дизайном і небезпечною реалізацією. Ми неспроста розрізняємо недоліки дизайну та дефекти реалізації, вони мають різні першопричини та їх усунення. Захищений дизайн все ще може мати дефекти реалізації, які призводять до вразливостей, які можуть бути використані. Небезпечний дизайн неможливо виправити ідеальною реалізацією, оскільки за визначенням необхідні засоби контролю безпеки ніколи не створювались для захисту від конкретних атак. Одним із факторів, що сприяють небезпечному дизайну, є відсутність профілю бізнес-ризиків, властивого програмному забезпеченню або системі, що розробляється, і, таким чином, неможливість визначити, який рівень проектування безпеки необхідний.

1.4.3 Вимоги та управління ресурсами

Збирайте та обговорюйте бізнес-вимоги до програми з бізнесом, включаючи вимоги захисту щодо конфіденційності, цілісності, доступності та автентичності всіх активів даних та очікуваної бізнес-логіки. Візьміть до уваги, наскільки відкритою буде ваша програма і чи потрібна вам сегрегація орендарів (додатково для контролю доступу). Скласти технічні вимоги, включаючи функціональні та нефункціональні вимоги безпеки.

Плануйте та обговорюйте бюджет, який охоплює все проектування, будівництво, тестування та експлуатацію, включаючи заходи безпеки.

1.4.4Безпечний дизайн

Захищений дизайн — це культура та методологія, які постійно оцінюють загрози та гарантують, що код надійно розроблений та перевірений для запобігання відомим методам атак. Моделювання загроз має бути інтегровано в сеанси уточнення (або подібні заходи); шукайте зміни в потоках даних і контролю доступу або інших засобів контролю безпеки.

Під час розробки історії користувача визначте правильні стани потоку та збою, переконайтеся, що вони добре зрозумілі та погоджені відповідальними сторонами. Проаналізуйте припущення та умови для очікуваних потоків і потоків невдач, переконайтеся, що вони все ще точні та бажані. Визначте, як підтвердити припущення та забезпечити умови, необхідні для належної поведінки. Переконайтеся, що результати задокументовані в історії користувача. Вчіться на помилках і запропонуйте позитивні стимули для покращення. Захищений дизайн не є ані доповненням, ані інструментом, який можна додати до програмного забезпечення[5].

1.4.6 Життєвий цикл безпечної розробки

Захищеному програмному забезпеченню потрібен безпечний життєвий цикл розробки, певна форма безпечного шаблону проектування, методологія з твердим покриттям, бібліотека захищених компонентів, інструменти та моделювання загроз. Зверніться до своїх спеціалістів із безпеки на початку проекту програмного забезпечення протягом усього проекту та обслуговування вашого програмного забезпечення. Розгляньте можливість використання моделі зрілості програмного забезпечення OWASP Software Assurance (SAMM), щоб допомогти структурувати ваші зусилля з розробки безпечного програмного забезпечення.

Небезпечний дизайн програми може мати серйозні наслідки для бізнесу, оскільки це може дозволити зловмисникам втручатися в логіку програми і призвести до розкриття конфіденційної інформації або компрометації веб-додатків (таблиця 1.4).

Таблиця 1.4 - A04:2021-Insecure Design **(**Небезпечний дизайн**)**

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу СЕРЕДНЯ	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Використовуючи підроблений дизайн для обману користувача.	Слабкі шаблони безпеки, ,використання власних алгоритмів.		Такі недоліки дають зловмисникам можливість отримати доступ до данних.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

1.5A05:2021-Security Misconfiguration (Помилка конфігурації безпеки)

1.5.1 Огляд

З більшою кількістю зрушень у програмне забезпечення, яке можна налаштувати, не дивно, що ця категорія рухається вгору. До відомих CWE включені конфігурація CWE-16 і неправильне обмеження посилання на зовнішній об’єкт XML CWE-611. Погано налаштований захист додатків може мати багато різних форм. Неправильні конфігурації можуть виникати у власному коді розробника, у коді готових функцій і функцій або через API.

Вони можуть з’являтися в самій програмі, на серверах і базах даних, які використовуються програмою, або в ресурсах, які використовуються під час процесу розробки. Будь-який рівень стека програм організації може виявляти недолік конфігурації, і чим більше шарів, тим більше шансів на помилку, що призведе до вразливості.Наприклад, користувачі часто неправильно налаштовують брандмауери.

Політики можна залишити занадто широкими, фактично залишивши мережу постійно відкритою. Тестові системи можуть не бути належним чином захищені від виробничих систем; і основний принцип найменших привілеїв не завжди виконується. Зі збільшенням обсягу програми стає все важче підтримувати жорсткі та ефективні конфігурації безпеки. Це стає більшою проблемою, якщо програма містить непотрібні або невикористані функції.

Це може бути викликано такими речами, як непотрібні порти, увімкнені під час циклу розробки, неправильне видалення облікових записів за замовчуванням тощо[6]. Якщо залишити ці невикористані функції, вони, ймовірно, будуть проігноровані або, принаймні, погано обслуговуються, що дасть зловмисникам більший потенціал для виявлення вразливостей (таблиця 1.5)

1.5.2 Опис

Програма може бути вразливою, якщо вона:

- Відсутнє відповідне посилення безпеки в будь-якій частині стеку програми або неправильно налаштовані дозволи на хмарні служби.
- Увімкнено або встановлено непотрібні функції (наприклад, непотрібні порти, служби, сторінки, облікові записи чи привілеї).
- Облікові записи за замовчуванням та їхні паролі залишаються активними та незмінними.

- Обробка помилок виявляє користувачам сліди стека або інші надто інформативні повідомлення про помилки.
- Для оновлених систем найновіші функції безпеки вимкнено або не налаштовано надійно.
- Параметри безпеки на серверах додатків, фреймворках програм бібліотеках, базах даних тощо, не налаштовані на безпечні значення.
- Сервер не надсилає заголовки чи директиви безпеки, або вони не налаштовані на захищені значення.
- Програмне забезпечення застаріле або вразливе (див. A06:2021-Уразливі та застарілі компоненти).
- Без узгодженого, повторюваного процесу налаштування безпеки додатків системи піддаються підвищеному ризику.

Таблиця 1.5 - A05:2021-Security Misconfiguration (Помилка конфігурації безпеки)

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки тяжкі	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник може використати застаріле або вразливе програмне забезпечення в системі.	Без параметру безпеки в серверах,програмах, базах данних що встановленні в безпечні значення.		Такі недоліки дають зловмисникам можливість використати для доступу до сервера,крадіжку данних, знаходження вразливостей контролю доступу.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

1.6A06:2021-Vulnerable and Outdated Components(Уразливі та застарілі компоненти**)**

1.6.1 Огляд

Уразливі компоненти – це відома проблема, яку ми намагаємося випробувати та оцінити ризики, і це єдина категорія, у якій немає жодних загальних вразливостей та ризиків (CVE), зіставлених із включеними CWE, тому використовується вага експлойтів/впливу за замовчуванням 5,0. Серед відомих CWE включені CWE-1104. Використання компонентів сторонніх розробників, які не обслуговуються.Проблема посилюється через тенденцію розробників програмного забезпечення використовувати компоненти з відкритим кодом, які часто розробляються програмістами в країнах, що розвиваються.

Під тиском для швидкої доставки ці компоненти недостатньо перевіряються перед використанням. Результатом можуть бути нові веб-сайти та програми з глибоко вбудованими уразливими місцями, невідомими оператору програми. Але як тільки ця вразливість буде виявлена кіберзлочинцями, програми, що використовують уразливий компонент, можна знайти та використати. Це може бути простий недолік у невеликому компоненті, але такий, який в кінцевому підсумку робить всю систему зламаною.

Це серйозна проблема для невеликих (а іноді не дуже) компаній або окремих осіб, які використовують бібліотеки або щоб отримати безкоштовні попередньо написані підпрограми, які використовуються для того, щоб зробити їхні веб-сайти більш привабливими та/або інтерактивними. Оскільки бібліотеки існують, сценарії часто використовуються без перевірки та без усвідомлення того, що вони можуть створити вразливість у програмі.

Звичайно, немає гарантії, що будь-який компонент програми – з відкритим кодом, запатентований чи ліцензований – буде повністю захищеним. Розробники та/або дослідники безпеки часто виявляють нові вразливості після публікації та видають виправлення безпеки, щоб їх виправити.

Не всі компоненти отримають необхідні виправлення, але навіть якщо вони отримають, якщо користувач не зможе їх застосувати, уразливість залишається[7].

Невиправлені відомі вразливості становлять серйозний ризик. Як тільки вразливість виправляється розробником, її існування стає загальнодоступним (таблиця 1.6).

Хакери поспішають розробляти та використовувати експлойти, перш ніж користувачі встигають виправити програми. Це зростає проблема: Gartner прогнозує, що до 2024 року 99% використаних недоліків безпеки будуть відомі принаймні протягом року.

1.6.2 Опис

Ймовірність, вразливості якщо:

- Якщо не відомо версії всіх компонентів, які використовуєте (як на стороні клієнта, так і на стороні сервера). Це включає компоненти, які ви використовуєте безпосередньо, а також вкладені залежності.
- Якщо програмне забезпечення є вразливим, не підтримується або застаріло. Це включає ОС, веб-сервер/сервер додатків, систему керування базами даних (СУБД), програми, API та всі компоненти, середовища виконання та бібліотеки.
- Якщо ви регулярно не скануєте на наявність уразливостей і не підписуєтесь на бюлетені з безпеки, пов’язані з компонентами, які ви використовуєте.

- Якщо ви не виправляєте чи не оновлюєте базову платформу, фреймворки та залежності вчасно, залежно від ризику. Це зазвичай трапляється в середовищах, коли виправлення є щомісячним або щоквартальним завданням під контролем змін, залишаючи організації відкритими на дні або місяці непотрібного впливу фіксованих уразливостей.
- Якщо розробники програмного забезпечення не перевіряють сумісність оновлених, оновлених або виправлених бібліотек.
- Якщо ви не захищаєте конфігурації компонентів (див. A05:2022-Помилка конфігурації безпеки).

Таблиця 1.6 - A06:2021-Vulnerable and Outdated Components(Уразливі та застарілі компоненти**)**

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник може використати не обновлені або застарілі компоненти.	Не обновленні компоненти що використовуються,програмне забезпечення що є вразливим і не підтримується розробником,не виконується сканування на пошук вразливостей.		Такі недоліки дають зловмисникам Скористуватися вразливостями для доступу до програми.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

1.7A09:2021-Security Logging and Monitoring Failures (Помилки ідентифікації та аутентифікації)

1.7.1 Огляд

Раніше відома як зламана аутентифікація, ця категорія опустилася з другої позиції і тепер включає перерахування загальних слабкостей (CWE), пов’язані з помилками ідентифікації. Серед відомих CWE включені CWE-297: неправильна перевірка сертифіката з невідповідністю хостів, CWE-287: неправильна аутентифікація та CWE-384: фіксація сеансу.

1.7.2 Опис

Підтвердження ідентичності користувача, аутентифікація та керування сеансами є критичними для захисту від атак, пов’язаних із автентифікацією. Можуть бути недоліки автентифікації, якщо програма дозволяє автоматизовані атаки, такі як заповнення облікових даних, коли зловмисник має список дійсних імен користувачів і паролів.

Дозволяє грубу силу або інші автоматизовані атаки.Дозволяє стандартні, слабкі або добре відомі паролі, такі як «Пароль1» або «адміністратор/адміністратор».

Використовує слабкі або неефективні процеси відновлення облікових даних і забуття пароля, такі як «відповіді на основі знань», які неможливо зробити безпечними.Використовує звичайний текст, зашифровані або слабо хешовані сховища паролів (див. A02:2022-Криптографічні збої).Відсутня або неефективна багатофакторна автентифікація.Видає ідентифікатор сеансу в URL-адресі.Повторне використання ідентифікатора сеансу після успішного входу.Неправильно скасовує ідентифікатори сеансів.

Сеанси користувача або маркери автентифікації (переважно маркери єдиного входу (SSO)) належним чином не анулюються під час виходу або періоду бездіяльності[8].

Таблиця 1.7 - A09:2021-Security Logging and Monitoring Failures (Помилки ідентифікації та аутентифікації)

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку/ діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник має список список дійсних імен користувачів і паролів які дійсні.	Використання слабких процесів відновлення данних,слабкі паролі доступу,		Такі недоліки дають зловмисникам можливість отримати доступ до данних користувача.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

Хоча недостатнє ведення журналів і моніторинг є занадто абстрактними, щоб бути вектором прямої атаки, вони впливають на виявлення та реагування на кожне порушення. Якщо інциденти веб-додатків і серверів не відстежуються належним чином, підозрілу активність можна легко пропустити. Якщо ризики безпеки не реєструються належним чином – або журнали погано зберігаються або важкодоступні – ці недоліки залишаться без уваги (таблиця 1.7)

1.8A08:2021-Software and Data Integrity Failures (Помилки програмного забезпечення та цілісності даних)

1.8.1 Огляд

Нова категорія на 2021 рік зосереджена на створенні припущень, пов’язаних з оновленнями програмного забезпечення, критичними даними та конвеєрами CI/CD без перевірки цілісності.

Один із найбільш зважених впливів даних про загальні вразливості та ризики/систему оцінки загальної вразливості (CVE/CVSS). Помітні загальні перерахування слабких місць (CWE) включають CWE-829: включення функціональності з ненадійної сфери управління, CWE-494: завантаження коду без перевірки цілісності та CWE-502: десеріалізація недовірених даних.

1.8.2 Опис

Порушення цілісності програмного забезпечення та даних стосуються коду та інфраструктури, які не захищають від порушення цілісності. Прикладом цього є те, коли програма покладається на плагіни, бібліотеки або модулі з ненадійних джерел, репозиторіїв і мереж доставки вмісту (CDN). Небезпечний конвеєр CI/CD може створити потенційну можливість несанкціонованого доступу, шкідливого коду або компрометації системи. Нарешті, багато програм тепер мають функцію автоматичного оновлення, де оновлення завантажуються без достатньої перевірки цілісності та застосовуються до раніше надійної програми.

Зловмисники потенційно можуть завантажувати власні оновлення для розповсюдження та запуску на всіх установках. Іншим прикладом є те, коли об’єкти або дані кодуються або серіалізуються в структуру, яку зловмисник може побачити та змінити, є вразливими для небезпечної десеріалізації.

Якщо критичні дані, що використовуються програмою, не перевіряються, зловмисники можуть втрутитися в них, що може призвести до досить серйозних проблем, таких як впровадження шкідливого коду в програмне забезпечення.

Багато програм тепер мають функцію автоматичного оновлення програмного забезпечення, що викликає занепокоєння щодо цілісності даних під час процесу оновлення.

Якщо зловмисники можуть здійснити атаку MitM і надіслати шкідливий код до програми під час процесу оновлення, дуже важливо, щоб такі оновлення ніколи не встановлювалися, інакше програма буде скомпрометована.Порушення цілісності програмного забезпечення та даних стосуються коду та інфраструктури, які не захищають від порушення цілісності[9].

Таблиця 1.8 - A08:2021-Software and Data Integrity Failures(Помилки програмного забезпечення та цілісності даних)

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник може використати недоліки програмного забезпечення для зміни.	Порушення цілісності програмного забезпечення які не мають захист, слабке кодування данних.		Зловмисник потенційно може завантажувати власні оновлення для розповсюдження та запуску на всіх установках	Зважайте на значення функції та даних, що ними обробляються для діяльності.

Це може статися, коли ви використовуєте програмне забезпечення з ненадійних джерел і репозиторіїв або навіть програмне забезпечення, яке було підроблено в джерелі, під час передачі або навіть у кеші кінцевої точки.

Зловмисники можуть використати це для потенційного несанкціонованого доступу, шкідливого коду або компрометації системи (таблиця 1.8)

1.9A09:2021-Security Logging and Monitoring Failures(Регистрація та моніторинг збоїв безпеки)

1.9.1 Огляд

Тестування журналу та моніторингу може бути складним, часто передбачаючи інтерв'ю або запит, чи були атаки виявлені під час проникнення. випробування. Для цієї категорії не так багато даних CVE/CVSS, але виявлення порушень і реагування на них є критичними. Тим не менш, це може дуже вплинути на підзвітність, видимість, оповіщення про інциденти та криміналістичну експертизу. Ця категорія виходить за рамки CWE-778.

Недостатнє ведення журналу, щоб включити CWE-117 Неправильну нейтралізацію виводу для журналів, CWE-223 Пропуск інформації, що стосується безпеки, і CWE-532 Вставка конфіденційної інформації до файлу журналу.

1.9.2 Опис

Повертаючись до топ-10 OWASP за 2021 рік, ця категорія має допомогти виявляти, ескалувати та реагувати на активні порушення. Без реєстрації та моніторингу порушення неможливо виявити. Недостатня реєстрація, виявлення, моніторинг та активна відповідь відбувається будь-коли події, які підлягають аудиту, такі як вхід у систему, невдалий вхід та транзакції високої вартості, не реєструються.

Попередження та помилки не генерують, неадекватні або нечіткі повідомлення журналу.Журнали програм і API не відстежуються на предмет підозрілої активності.Журнали зберігаються лише локально.Відповідні пороги попередження та процеси ескалації реагування не діють або не ефективні.

Тестування на проникнення та сканування за допомогою інструментів динамічного тестування безпеки додатків (DAST) (наприклад, OWASP ZAP) не ініціюють попередження[10].

Програма не може виявляти, посилювати або сповіщати про активні атаки в режимі реального часу або майже в реальному часі.Ви вразливі до витоку інформації, роблячи журнали та події видимими для користувача або зловмисника (таблиця 1.9).

Таблиця 1.9 - A08:2021-Software and Data Integrity Failures(Помилки програмного забезпечення та цілісності даних)

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу ЛЕГКА	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку/діяльно
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник може використати порушення через відсутність моніторингу та реєстрації	Порушення через відсутність моніторингу та реєстрації,порушення данних.		Зловмисник може отримати доступ до інформації або крадіжку конференційної інформації користувача.	Зважайте на значення функції та даних, що ними оброблюються для діяльності.

1.10A10:2021-Server-Side Request Forgery(Підробка запиту на стороні сервера)

1.10.1 Огляд

Ця категорія додається з опитування 10 найкращих спільнот (№1). Дані показують відносно низький рівень захворюваності з вищим за середнє охоплення тестуванням і вищими за середні рейтинги потенційних можливостей використання та впливу. Оскільки нові записи, ймовірно, будуть єдиним або невеликим кластером Common Weakness Enumerations (CWE) для привернення уваги та обізнаності.

1.10.2 Опис

Помилки SSRF виникають щоразу, коли веб-додаток отримує віддалений ресурс без перевірки URL-адреси, наданої користувачем. Це дозволяє зловмиснику змусити програму надіслати створений запит до несподіваного місця призначення, навіть якщо він захищений брандмауером, VPN або іншим типом списку контролю доступу до мережі (ACL).Оскільки сучасні веб-додатки надають кінцевим користувачам зручні функції, отримання URL-адреси стає звичайним сценарієм. У результаті зростає захворюваність на ССР. Крім того, серйозність SSRF стає вищою через хмарні сервіси та складність архітектур.Зазвичай зловмисник може прочитати відповіді, надіслані з початкового запиту.

Це можна використовувати для сканування портів внутрішньої мережі або локального хосту, надсилання запитів до серверної частини бази даних, доступу до зовнішніх постачальників хмарних послуг для отримання маркерів або читання локальних файлів[11].

Таблиця 1.10 - A10:2021-Server-Side Request Forgery **(**Підробка запиту на стороні сервера**)**

Чинники Загрози	Вектори атаки	Слабкі місця безпеки		Технічні наслідки	Наслідки для діяльності
Специфічні для додатка	Можливість зламу СЕРЕДНЯ	Поширеність Звичайна	Можливість виявлення СЕРЕДНЯ	Наслідки помірні	Специфічні для додатку /діяльності
Будь-хто з доступом до мережі може відправити запит до вашого додатку.	Зловмисник може використати порушення через відсутність моніторингу та реєстрації	Порушення через відсутність URL-адреси.		Зловмисник може отримати доступ до інформації або крадіжку конференційної інформації користувача.	Зважайте на значення функції та даних, що ними обробляються для діяльності.

2. КІБЕРАТАКИ НА ВЕБ-СЕРВІСИ

Кібератака – це набір дій, які здійснюються суб’єктами загрози, які намагаються отримати несанкціонований доступ, викрасти дані або завдати шкоди комп’ютерам, комп’ютерним мережам чи іншим комп’ютерним системам. Кібератаку можна запустити з будь-якого місця. Атака може бути здійснена окремою особою або групою з використанням однієї або кількох тактик, прийомів і процедур (TTP).

Особи, які здійснюють кібератаки, зазвичай називаються кіберзлочинцями, акторами загроз, поганими акторами або хакерами. Вони можуть працювати поодинці, у співпраці з іншими нападниками або в складі організованої злочинної групи. Вони намагаються виявити вразливі місця — проблеми чи слабкі місця в комп’ютерних системах — і використовувати їх для досягнення своїх цілей.

Кіберзлочинці можуть мати різні мотиви під час здійснення кібератак. Деякі здійснюють напади з метою особистої або фінансової вигоди. Інші є «хактивістами», які діють в ім'я соціальних або політичних причин. Деякі атаки є частиною операцій кібервійни, які проводяться національними державами проти своїх супротивників, або діють у складі відомих терористичних груп.

2.1 Введення SQL-коду

Введення SQL-коду – це атака, під час якої шкідливий код вставляється в рядки, які пізніше будуть передані екземпляру SQL Server для аналізу та виконання.

Будь-яку процедуру, яка генерує оператори SQL, слід розглядати на предмет вразливостей ін’єкції коду, оскільки SQL Server виконує будь-які синтаксично правильні запити, які він отримує. Навіть параметризовані дані можуть маніпулювати кваліфікованим зловмисником.

Основною формою атаки є пряме введення коду в змінні, які вводяться користувачами, які поєднуються з командами SQL і виконуються (рис. 2.1).

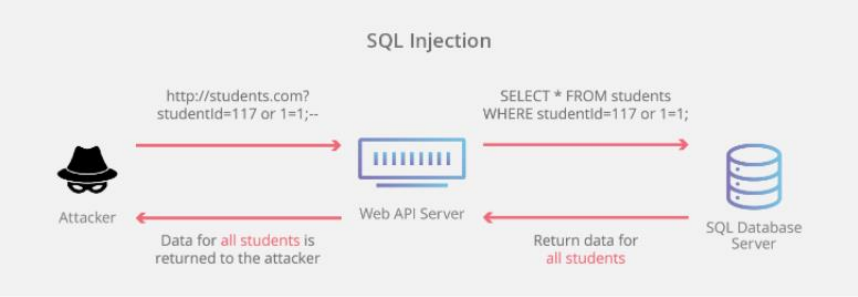

Рисунок 2.1– Введення SQL-коду

Менш очевидна атака вводить небезпечний код у рядки, призначені для зберігання в таблиці або як метадані. Коли збережені рядки згодом об’єднуються з динамічною командою SQL, виконується небезпечний код. Атака здійснюється шляхом передчасного припинення текстового рядка та додавання до нього нової команди. Оскільки додаткові рядки можуть бути додані до вставленої команди перед виконанням, зловмисник закінчує введений рядок знаком коментаря «--». Весь наступний текст ігнорується під час виконання.

2.2 PHP-ін’єкція

PHP- ін’єкція - це вразливість на рівні програми, яка може дозволити зловмиснику здійснювати різні види шкідливих атак, такі як ін’єкція коду, ін’єкція SQL.

Обхід шляху та відмова в обслуговуванні програми, залежно від контексту. Уразливість виникає, коли введені користувачем дані не продезінфіковані належним чином перед тим, як їх передати у функцію unserialize PHP.

Оскільки PHP дозволяє серіалізацію об’єктів, зловмисники можуть передавати шкідливий код до веб-сервера для отримання контролю над веб-застосунком (рис.2.2).

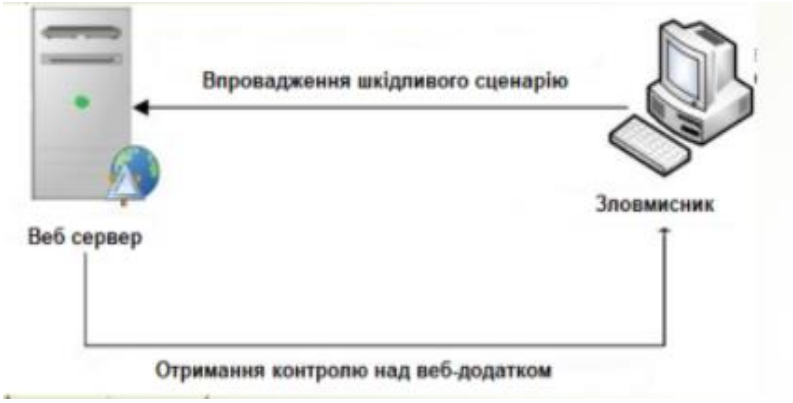

Рисунок 2.2 – PNP - ін’єкція

Для успішного використання вразливості PHP Object Injection повинні бути виконані дві умови програма повинна мати клас, який реалізує магічний метод PHP (наприклад, __wakeup або __destruct), який можна використовувати для здійснення шкідливих атак або для запуску «ланцюжка POP». Усі класи, використані під час атаки, мають бути оголошені під час виклику уразливого unserialize(), інакше для таких класів має підтримуватися автозавантаження об’єктів.

2.3 Спуфінг-IP

Спуфінг IP — це заміна адреси відправника, одного з полів заголовка IP, записом іншого значення. Складність полягає в тому, що машина, отримавши заголовок з такою адресою, надішле відповідь на цю адресу, а не на адресу зломисник. У разі TCP-з'єднання необхідно отримати відповідь від адресата, щоб встановити з ним з'єднання.

При встановленні TCP-з'єднання важливий так званий ISN (Initial Sequence Number) - початковий порядковий номер. Під час встановлення з'єднання між машинами передається порядковий номер клієнта, позначений як ISNc, і ISN сервера, як ISN, також передається.

Підробка IP-адреси відбувається, коли хакер, як усередині корпорації, так і за її межами, видає себе за авторизованого користувача. Це можна зробити двома способами.

По-перше, хакер може використовувати IP-адресу, яка знаходиться в межах дозволених IP-адрес, або авторизовану зовнішню адресу, якій дозволено доступ до певних мережевих ресурсів. Атаки підробки IP-адресів часто є відправною точкою для інших атак.

Класичним прикладом є DoS-атака, яка починається з того, що чужа адреса приховує справжню особу хакера (рис.2.3).Як правило, підробка IP обмежується вставленням неправдивої інформації або шкідливих команд у звичайний потік даних, що передається між клієнтським і серверним додатком або через канал зв’язку між одноранговими партнерами. Для двостороннього зв’язку хакер повинен змінити всі таблиці маршрутизації, щоб спрямовувати трафік на підроблену IP-адресу.

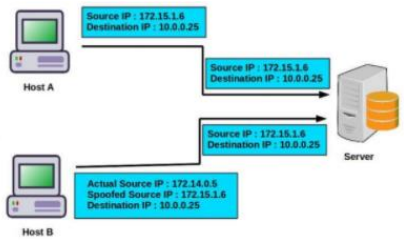

Рисунок 2.3 – Атака з заміною IP-адреси

Однак деякі хакери навіть не намагаються отримати відповідь від додатків. Якщо основним завданням є отримання важливого файлу від системи, то відповіді додатків не мають значення.

Коли зломщик намагається встановити TCP-з'єднання з підробленою IP-адресою, сервер надсилає комп'ютеру А пакет SYN-ACK, що містить його ISN. Оскільки комп’ютер А не надіслав SYN-пакет на сервер, він відповість пакетом RST, щоб припинити невідоме з’єднання. Хакер повинен почекати, поки комп'ютер А не буде вимкнено або перезавантажено.

Зловмисник не зможе побачити ISN, надісланий з однієї машини на іншу. Йому знадобиться цей ISN на третьому кроці, коли він повинен буде збільшити його на 1 і надіслати. Хакер повинен вгадати ISN. У старих операційних системах (ОС) було дуже легко вгадати ISN - він збільшувався на один з кожним підключенням. Сучасні ОС використовують механізм, який запобігає вгадуванням ISN. Сучасні сервіси використовують ім’я користувача та пароль для аутентифікації та передачі даних у зашифрованому вигляді, тому в наш час немає необхідності в IP-спуфінгу.

2.4 Атака із заміною ARP

ARP(англ. Address Resolution Protocol) — це протокол у комп’ютерних мережах, призначений для визначення MAC-адреси іншого комп’ютера за відомою IP-адресою[12].

Заміна ARP полягає в незаконній заяві про те, що ви є MAC-адресою певної IP-адреси, внаслідок чого запитувач помилково оновлює таблицю кешу ARP, щоб дані, надіслані підробленим хостом, надсилалися атакуючому хосту замість ідеального IP-хосту призначення. ARP-атаки можнgta виконувати лише в мережі Ethernet, наприклад, комп’ютерна кімната, інтранет, корпоративна мережа тощо (рис.2.4).

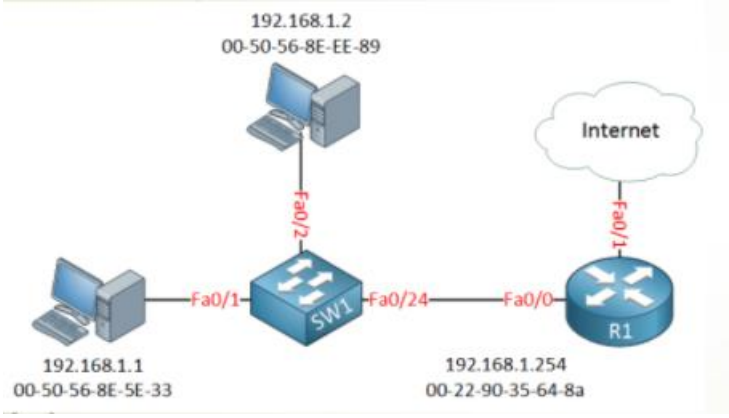

Рисунок – 2.4 Заміна ARP

Неможливо атакувати зовнішні мережі (Інтернет, локальні мережі за межами локальної мережі).Якщо зловмисник обманює цільовий хост, передачу даних між двома хостами можна контролювати.

2.5 Спуфінг DNC-серверів.

DNS-сервер - це спеціальний комп'ютер, на якому зберігаються IP-адреси веб-сайтів. Основними функціями DNS-сервера є надання браузеру адреси сайту за доменним іменем і кешування записів DNS домену.

Спуфінг DNS - це тип комп’ютерної атаки, при якій користувача змушують перейти на підроблений веб-сайт, замаскований так, щоб він виглядав як справжній, з наміром перенаправити трафік або вкрасти облікові дані користувачів. Атаки підробки можуть тривати тривалий період часу, не будучи виявлені, і можуть спричинитиь серйозні проблеми з безпекою.

Сервер доменних імен (DNS) розв’язує алфавітні доменні імена, такі як www.example.com, у відповідні IP-адреси, які використовуються для визначення місцезнаходження та зв’язку між вузлами в Інтернеті.

Спуфінг DNS здійснюється шляхом заміни IP-адрес, що зберігаються на DNS-сервері, на ті, які контролюються зловмисником.

Коли це буде зроблено, кожен раз, коли користувачі намагаються перейти на певний веб-сайт, вони перенаправляються на фальшиві веб-сайти, розміщені зловмисником на підробленому DNS-сервері. Існують в основному два способи, за допомогою яких здійснюється спуфінг DNS – отруєння кешу DNS і спуфінг DNS ID. При отруєнні кешу DNS локальний DNS-сервер замінюється зламаним DNS-сервером, що містить налаштовані записи справжніх імен веб-сайтів з власними IP-адресами зловмисника.

Таким чином, коли на локальний DNS-сервер надсилається запит на дозвіл IP, він зв’язується з скомпрометованим DNS-сервером, в результаті чого користувач перенаправляється на помилковий веб-сайт, створений зловмисником.

Під час спуфінгу ідентифікатора DNS, ідентифікатор пакета та інформація IP, згенерована для запиту на розв’язання, надісланого клієнтом, дублюється з неправдивою інформацією (рис.2.5).

Оскільки ідентифікатор відповіді відповідає ідентифікатору запиту, клієнт приймає відповідь, що містить інформацію, яка не очікується.

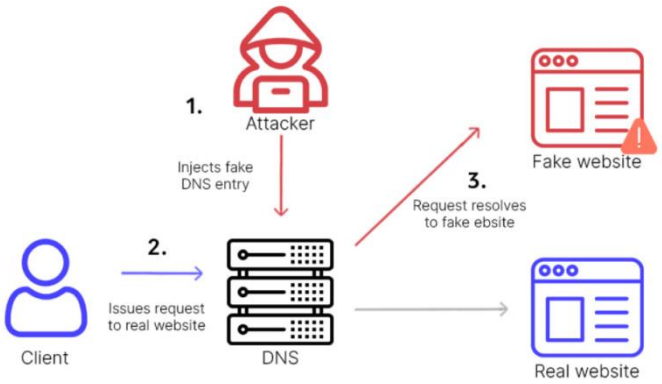

Рисунок 2.5 – Спуфінг DNS-сервера

Загальні поради щодо запобігання спуфінгу DNS включають підтримання актуального програмного забезпечення DNS, підтримку окремих серверів для загальнодоступних і внутрішніх служб тавикористання безпечних ключів для підписання оновлень, отриманих з інших серверів DNS, щоб уникнути оновлень із ненадійних джерел.

2.6 Спуфінг MAC-адрес

MAC-адрес— унікальний ідентифікатор, присваиваемый кожній одиниці активного обладнання або некоторим їх інтерфейсом у комп’ютерних мережах Ethernet. При проектуванні стандарту Ethernet було передбачено, що будь-яка сетева карта (рівно як і вбудований сетевий інтерфейс) повинна мати унікальний шестибайтний номер (MAC-адрес), «прошитий» в ній при виготовленні[13].

Цей номер використовується для ідентифікації відправника та одержувача фрейма; і передбачається, що при появі в мережі нового комп'ютера (або іншого пристрою, здатного працювати в мережі) сетевому адміністратору не приходить налаштувати цей MAC-адрес комп'ютера вручну.

Спуфінг MAC-адрес - атака канального рівня, що полягає в тому, що на мережевій карті змінюється MAC-адресу, що змушує комутатор відправляти на порт, до якого підключений зловмисник, пакети, які до цього він бачити не міг.В основному його використовують хакери для атак передавання і збору інформації, отримання паролів і для тестування мережі (рис.2.6)

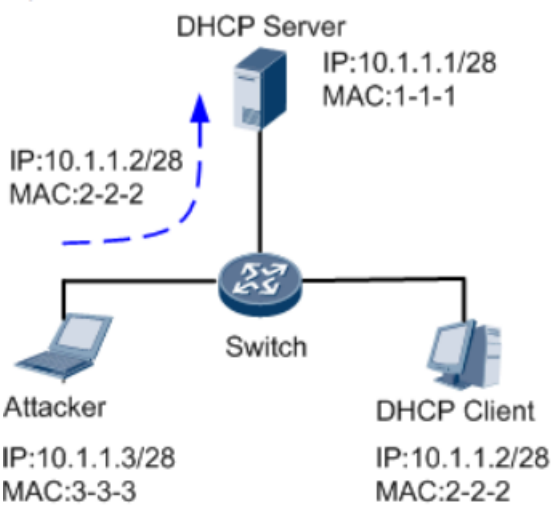

Рисунок 2.6 – Спуфінг MAC-адрес

Цей метод часто використовується в мережах Wi-Fi у два етапи: прослуховується мережевий трафік і виділяються MAC-адреси, після чого MAC-адреса безпосередньо змінюється.Однією з найважливіших стратегій, що використовуються в атаках підміну MAC, є маскування MAC-адреси, також відоме як спуфінг MAC ID. Підробка загалом означає різноманітні методи, доступні для керування та роботи основної адресної системи в різних комп’ютерних мережах.

3.ЗАХИСТ ВЕБ-СЕРВІСІВ

3.1 Способи захисту від Broken Access Control **(**Порушений контроль доступу**)**

Структура програми може пом’якшити проблеми контролю доступу, реалізуючи додаткові рівні безпеки для захисту конфіденційних даних. Таким чином, навіть якщо зловмисник отримує доступ до певного рівня привілейованих функцій, дані користувача або адміністративні команди можуть залишатися захищеними.

Пишучи про порушення IRS у 2015 році, блог Gartner зазначив: ви повинні припустити, що злочинці можуть пройти принаймні через один рівень, тому чим більше шарів і заходів у вас буде, тим краще для вас.

Забезпечення контролю доступу завжди має здійснюватися на стороні сервера. Навіть якщо елементи керування на стороні клієнта реалізовані, вони ніколи не повинні бути єдиним засобом аутентифікації, особливо коли йдеться про конфіденційні дані.

За допомогою правильних інструментів елементи керування на стороні клієнта завжди можна обійти або змінити.Поширеним кроком у веб-додатках є включення авторизованих IP-адрес та ідентифікаторів пристроїв в автентифікацію.

Користувач входить з нового пристрою або місця, і не може ввійти, доки не введе код MFA, надісланий програмою. Це само по собі не є водонепроникним захистом, але воно запобігає порушенню контролю доступу зловмисником, який отримав доступ до основних облікових даних для входу.

Як запобігти. Контроль доступу ефективний лише у надійному серверному коді або API без сервера, де зловмисник не може змінити перевірку контролю доступу або метадані.

- Крім загальнодоступних ресурсів, заборонити за замовчуванням.
- Впроваджувати механізми контролю доступу один раз і використовуйте їх повторно в усій програмі, включаючи мінімізацію використання між джерелами спільного використання ресурсів (CORS).
- Модель контролю доступу має забезпечувати право власності на записи, а не визнавати, що користувач може створювати, читати, оновлювати або видаляти будь-який запис.
- Унікальні вимоги до бізнес-лімітів додатків мають забезпечуватися моделями домену.
- Вимкнути список каталогів веб-сервера та переконайтеся, що метадані файлів (наприклад, .git) і файли резервної копії відсутні в кореневих веб-сайтах.
- Реєструвати помилки контролю доступу, сповіщайте адміністраторів, коли це необхідно (наприклад, повторні збої).
- Обмежити швидкість доступу до API та контролера, щоб мінімізувати шкоду від автоматизованих інструментів атак.
- Ідентифікатори сеансу, що зберігають стан, повинні бути визнані недійсними на сервері після виходу. Токени JWT без стану мають бути короткочасними, щоб звести до мінімуму можливості для зловмисника. Для довготривалих JWT настійно рекомендується дотримуватися стандартів OAuth, щоб скасувати доступ.

3.2 Способи захисту від Cryptographic Failures (Криптографічні збої)

Конфіденційні дані, що зберігаються, включають збережені бази даних паролів, конфіденційну інформацію користувача, необхідну для програми, і конфіденційну інтелектуальну власність, що належить компанії (для наших цілей останні дві можуть розглядатися аналогічно).

Якщо перший не належним чином хешується, а другий не зашифрований належним чином, він підпадає під ризик розкриття конфіденційних даних. Будь-який успішний хакер може вкрасти дані та використати їх.

Особливо важливо, щоб у базах даних паролів використовувався сучасний, повільний алгоритм хешування з випадковим додаванням солі – і навіть кілька операцій хешування – через схильність до злому вкрадених хешованих паролів райдужними таблицями.

Не менш важливий вибір алгоритму шифрування для інших даних у стані спокою. Загальна порада полягає в тому, щоб уникати власного шифрування – завжди використовуйте один із загальнодоступних алгоритмів, який перевірено експертами та перевірений часом.

Але так само важливо, як алгоритм, його конфігурація, реалізація та бізнес-процеси навколо нього. Хороше шифрування неможливо зламати. Натомість хакери часто намагаються використовувати бізнес-процеси. Наприклад, номери банківських карток можуть зберігатися в зашифрованому вигляді, але для використання їх потрібно розшифрувати. Саме в цей момент зловмисне програмне забезпечення спробує зірвати дані та надіслати їх хакеру.

Запобігти розголошенню конфіденційних даних складно і складно, і це не просто випадок шифрування всіх даних. По-перше, ви повинні знати, які дані потрібно зашифрувати, а потім ви повинні знати, де вони знаходяться. У першому може допомогти класифікація даних; але без дорогого програмного забезпечення для моніторингу даних це стомлююче та схильне до помилок. І небагато компаній у наші дні гнучких хмарних і мобільних обчислень можуть чесно стверджувати, що знають, де знаходяться всі їхні конфіденційні дані. Як тільки дані, які підлягають шифруванню, відомі, слід використовувати ефективне шифрування. Але бізнес-процеси навколо конфіденційних даних також мають бути безпечними.

Якщо зашифровані дані автоматично розшифровуються програмою, якій потрібно використовувати ці дані, то потрібно зняти занепокоєння щодо програми.

Простіше кажучи, запобігти розкриттю конфіденційних даних настільки складно, що багато компаній не приділяють цьому належної уваги, вважаючи за краще захистити інфраструктуру, а не дані. Історія показує, що це неправильний підхід. І, нарешті, примітка на майбутнє… Найкращі криптографічні алгоритми сьогодні не піддаються зламу. Це не триватиме.

Ймовірно, протягом наступного десятиліття нове покоління квантових комп'ютерів стане комерційно доступним.

Ці комп'ютери матимуть можливість переробити більшість існуючого шифрування за години, а не за тисячі років, які зараз потрібні. Звичайно, поточне шифрування, яке використовується для передачі даних, впаде. У той же час квантовий процес, відомий як квантовий розподіл ключів (вже стає доступним і в даний час використовується на виборах у Швейцарії), застосовуватиме фізичні правила квантової механіки для захисту даних при передачі. У якийсь момент незабаром поточна найслабша область передачі даних стане найбезпечнішою частиною запобігання розкриттю конфіденційних даних.

Виконайте принаймні наступне:

- Класифікувати дані, які обробляються, зберігаються або передаються програмою. Визначити, які дані є конфіденційними відповідно до законів про конфіденційність, нормативних вимог або потреб бізнесу.
- Не зберігати конфіденційні дані без потреби. Відмовтися від нього якомога швидше або скористайтеся PCI DSS-сумісною токенізацією або навіть скороченням. Дані, які не зберігаються, не можуть бути вкрадені.
- Обов'язково зашифруйте всі конфіденційні дані.

- Забезпечте наявність актуальних і надійних стандартних алгоритмів, протоколів і ключів; використовувати правильне керування ключами.
- Шифрувати всі дані, що передаються, за допомогою безпечних протоколів, таких як TLS з шифрами прямої секретності (FS), пріоритетом шифрування сервером і безпечними параметрами. Застосовуйте шифрування за допомогою таких директив, як HTTP Strict Transport Security (HSTS).
- Вимкнути кешування відповідей, які містять конфіденційні дані.
- Застосовувати необхідні заходи безпеки відповідно до класифікації даних.
- Не використовувати застарілі протоколи, такі як FTP і SMTP, для транспортування конфіденційних даних.
- Зберігайте паролі, використовуючи потужні адаптивні та сольові функції хешування з робочим коефіцієнтом (коефіцієнт затримки), таким як Argon2, scrypt, bcrypt або PBKDF2.
- Вектори ініціалізації повинні бути обрані відповідно до режиму роботи. Для багатьох режимів це означає використання CSPRNG (криптографічно захищений генератор псевдовипадкових чисел).
- Для режимів, які вимагають одноразового значення, вектор ініціалізації не потребує CSPRNG. У всіх випадках IV ніколи не слід використовувати двічі для фіксованого ключа.
- Завжди використовувати автентифіковане шифрування замість простого шифрування.
- Ключі повинні генеруватися криптографічно випадковим чином і зберігатися в пам'яті у вигляді байтових масивів. Якщо використовується пароль, його потрібно перетворити на ключ за допомогою відповідної функції отримання базового ключа пароля.

- Переконайтися, що криптографічна випадковість використовується там, де це доречно, і що вона не була засіяна передбачуваним способом або з низькою ентропією. Більшість сучасних API не вимагає від розробника завантажувати CSPRNG для забезпечення безпеки.
- Уникати застарілих криптографічних функцій і схем заповнення, таких як MD5, SHA1, PKCS номер 1 v1.5 .
- Перевірте самостійно ефективність конфігурації та налаштувань.

3.3 Способи захисту від Injection **(**Ін'єкція**)**

Ін’єкційні атаки не вимагають великої хакерської майстерності. Маючи лише браузер і вільний час, хакери можуть знайти вразливості, просто погравши з синтаксисом URL-адреси.

Люди з невеликими знаннями кодування або зовсім без них можуть знайти і слідувати онлайн-підручникам, а також є багато автоматизованих інструментів для сліпого впровадження SQL.

На щастя, низький бар’єр проникнення для ін’єкційних атак у більшості випадків має відповідне просте рішення: належне тестування та захист будь-якої бази даних і те, як вона обробляє запити даних. Використання якісного API має важливе значення, і необхідно встановити засоби контролю, щоб захистити від масового розкриття даних і використання спеціальних символів у синтаксисі коду.

Звучить складно, але всього цього можна досягти за умови належної модернізації баз даних і дотримання вже існуючих рекомендацій.Однак найважливішою частиною захисту від недоліків ін'єкції є старанність; OWASP рекомендує провести ретельний огляд вихідного коду вручну та автоматичне тестування «всіх параметрів, заголовків, URL-адрес, файлів cookie, JSON, SOAP та введених даних XML».

Рекомендується використовувати як статичні, так і динамічні інструменти тестування безпеки додатків (SAST і DAST), наприклад ті, що вбудовані в рішення ImmuniWeb High-Tech Bridge і використовуються для захисту від будь-яких нових вразливостей у майбутньому.

Щоб запобігти ін’єкції, потрібно зберігати дані окремо від команд і запитів:

- Кращим варіантом є використання безпечного API, який повністю уникає використання інтерпретатора, забезпечує параметризований
- інтерфейс або переходить до інструментів об’єктно-реляційного відображення (ORM) (Навіть якщо параметризовані, збережені процедури все ще можуть вводити SQL-ін’єкцію, якщо PL/SQL або T-SQL об’єднує запити та дані або виконує ворожі дані за допомогою EXECUTE IMMEDIATE або exec()).
- Використовуйте позитивну перевірку введення на стороні сервера. Це не повний захист, оскільки багато програм вимагають спеціальних символів, таких як текстові області або API для мобільних додатків.
- Для будь-яких залишкових динамічних запитів екрануйте спеціальні символи, використовуючи спеціальний escape-синтаксис для цього інтерпретатора. (Структури SQL, такі як імена таблиць, імена стовпців тощо, не можна екранувати, і, отже, надані користувачем імена структур є небезпечними. Це поширена проблема в програмному забезпеченні для написання звітів).
- Використовувати LIMIT та інші елементи керування SQL у запитах, щоб запобігти масовому розголошенню записів у разі впровадження SQL.

3.4 Способи захисту від Insecure Design **(**Небезпечний дизайн**)**

Категорія небезпечного дизайну відображає 40 CWE, які пов’язані з помилками оцінки введення, правильним використанням API та функцій програми, проблемами керування привілеями та вразливими місцями, безпекою зв’язку з програмою, безпечним зберіганням конфіденційної інформації.

Як запобігти:

- Створити і використовувати бібліотеку безпечних шаблонів проектування або готових до використання компонентів з твердим покриттям.
- Використовувати моделювання загроз для критичної аутентифікації, контролю доступу, бізнес-логіки та ключових потоків
- Інтегрувати мову безпеки та елементи керування в історії користувачів.
- Інтегрувати перевірки правдоподібності на кожному рівні вашої програми (від інтерфейсу до бекенда).
- Написати модульні та інтеграційні тести, щоб підтвердити, що всі критичні потоки стійкі до моделі загроз.
- Скомпілювати варіанти використання та варіанти неправильного використання для кожного рівня вашої програми.
- Розділити рівні рівня на системному та мережевому рівнях залежно від потреб у відкритті та захисті.
- Надійно розділити орендарів за проектом на всіх рівнях.
- Обмежувати споживання ресурсів користувачем або службою.

Сценарій № 1. Робочий процес відновлення облікових даних може включати «запитання та відповіді», що заборонено NIST 800-63b, OWASP ASVS та OWASP Top 10.

Питання та відповіді не можуть вважатися доказом особи як кількох осіб. можуть знати відповіді, тому вони заборонені. Такий код слід видалити та замінити на більш безпечний дизайн.

Сценарій № 2. Мережа кінотеатрів надає знижки на групове бронювання та має максимум п’ятнадцять відвідувачів, перш ніж вимагати депозиту. Зловмисники можуть змоделювати цей потік і перевірити, чи можуть вони забронювати шістсот секунд.

3.5 Способи захисту від Security Misconfiguration(Помилка конфігурації безпеки)

Уразливості безпеки можна виявити з несподіваних місць. Повідомлення про помилки можуть містити підказки для зловмисників, якщо вони обробляються неналежним чином. Залишковий код і зразки додатків від процесу розробки можуть містити відомі вразливості, що дозволяє зловмисникам отримати доступ до сервера додатків. Якщо не налаштовано належним чином, інформація про налагодження, як-от ці повідомлення про помилки та детальні стеки, життєво важливі для розробника, можуть стати зброєю в руках зловмисника.

Однак у міру поширення варіантів хмарного сховища в центрі уваги стала більш проста, але руйнівна неправильна конфігурація безпеки: неможливість заблокувати доступ до даних, що зберігаються на пристроях зберігання даних, що працюють в Інтернеті.

Здається, люди припускають, що будь-яка залучена третя сторона забезпечить безпеку – що зазвичай не відповідає дійсності.

Ця проблема з неправильною конфігурацією також може виникнути на приватних серверах, які використовують стороннє програмне забезпечення. На початку цього року брокер даних s розкрив величезну базу даних особистої інформації про 218 мільйонів осіб, 110 мільйонів домогосподарств і 21 мільйон компаній.

База даних була фронтальною за допомогою Elasticsearch, але контроль доступу не був розгорнутий.

Неправильна конфігурація безпеки виникає через людську помилку, а не через загальні недоліки в протоколах або поширених векторах атак. Це означає, що добре структурований цикл розробки та оновлення, якщо його правильно запровадити, надійно протидіє цьому ризику.

Необхідно запровадити повторюваний процес для захисту та тестування програми під час розробки, розгортання будь-яких нових функцій, а також коли будь-який компонент оновлюється або змінюється.

Як запобігти:

- Необхідно впровадити безпечні процеси встановлення, включаючи:
- Повторюваний процес зміцнення дозволяє швидко та легко розгортати інше середовище, яке належним чином заблоковано.
- Середовища розробки, забезпечення якості та виробництва мають бути налаштовані однаково, з різними обліковими даними, які використовуються в кожному середовищі. Цей процес слід автоматизувати, щоб мінімізувати зусилля, необхідні для створення нового безпечного середовища.
- Мінімальна платформа без будь-яких непотрібних функцій, компонентів, документації та зразків. Видаліть або не встановлюйте невикористані функції та рамки.

- Завдання для перегляду та оновлення конфігурацій, що відповідають усім нотаткам безпеки, оновленням і виправленням, як частина процесу керування виправленнями (див. A06:2021-Уразливі та застарілі компоненти). Перегляньте дозволи хмарного сховища (наприклад, дозволи сегмента S3).
- Сегментована архітектура додатків забезпечує ефективне та безпечне розділення між компонентами або клієнтами за допомогою сегментації, контейнеризації або груп безпеки в хмарі (ACL).
- Надсилання директив безпеки клієнтам, наприклад, заголовків безпеки.
- Автоматизований процес для перевірки ефективності конфігурацій і налаштувань у всіх середовищах.

Приклад ци

Сценарій № 1. Сервер додатків постачається зі зразками програм, не видаленими з робочого сервера. Ці приклади програм мають відомі недоліки безпеки, які зловмисники використовують для компрометації сервера. Припустимо, одна з цих програм є консолью адміністратора, а облікові записи за замовчуванням не змінювалися. У цьому випадку зловмисник увійде в систему з паролями за замовчуванням і перейде на себе.

Сценарій №2. Список каталогу не вимкнено на сервері. Зловмисник виявляє, що може просто перерахувати каталоги. Зловмисник знаходить і завантажує скомпільовані класи Java, які вони декомпілюють і перероблюють, щоб переглянути код. Потім зловмисник знаходить серйозний недолік контролю доступу в програмі.

Сценарій №3: конфігурація сервера додатків дозволяє повертати користувачам детальні повідомлення про помилки, наприклад, трасування стека. Це потенційно розкриває конфіденційну інформацію або основні недоліки, такі як версії компонентів, які, як відомо, є вразливими.

Сценарій №4. Постачальник хмарних послуг (CSP) має дозволи на спільний доступ за замовчуванням, відкриті в Інтернеті іншими користувачами CSP. Це дозволяє отримати доступ до конфіденційних даних, що зберігаються в хмарному сховищі.

3.6 Способи захисту від від Vulnerable and Outdated Components(Уразливі та застарілі компоненти**)**

Обізнаність — це найкращий захист компанії від ризиків, пов’язаних із відомими вразливими місцями. OWASP рекомендує розробникам додатків і користувачам постійно контролювати підтримку авторів і оновлення вразливостей будь-яких компонентів програми сторонніх розробників.

Якщо компонент перестає підтримуватися та оновлюватися його автором, користувачі повинні або шукати альтернативу, або використовувати віртуальне виправлення, доки не буде знайдено більш належним чином захищене рішення.

Усвідомлення будь-якої вразливості компонентів залишається серйозною проблемою, що стимулює зростання компаній з аналізу складу програмного забезпечення (SCA).

Одним із прикладів є BlackDuck, тепер частина Synopsis, яка допомагає компаніям захищати програмне забезпечення з відкритим кодом і керувати ним.

Як завжди, будь-яка програма має бути впорядкованою та має використовувати якомога менше компонентів і файлів. Чим більше компонентів використовується програмою, тим більша ймовірність появи невиправлених уразливостей. Необхідно видалити будь-які непотрібні функції, а також будь-які залежності чи посилання, які все ще можуть містити недолік безпеки. Необхідно використовувати лише компоненти з надійних джерел із захищеними посиланнями та підписаними пакетами, щоб гарантувати, що неавторизована сторона не змінила компонент.

Встановлення чітко визначеного процесу керування виправленнями допоможе інформувати розробників додатків і захистити компоненти. Компанія повинна визначити свої процедури відповідно до потреб безпеки даних, якими обробляє додаток.

Політика повинна враховувати підтримання компонентів в актуальному стані, а також вказувати процедури, коли виявлено вразливість або код більше не можна виправити. Це може включати віртуальне виправлення або переведення програми в автономний режим, поки вразливість не буде виправлена.

Повинен бути запроваджений процес керування виправленнями, щоб:

- Видалити невикористані залежності, непотрібні функції, компоненти, файли та документацію.
- Постійно інвентаризуйте версії компонентів як на стороні клієнта, так і на стороні сервера (наприклад, фреймворки, бібліотеки) та їх залежності, використовуючи такі інструменти, як версії, OWASP Dependency Check, retire.js тощо.
- Постійно відстежуйте такі джерела, як Common Vulnerability and Exposures (CVE) та Національна база даних уразливостей (NVD) для виявлення вразливостей компонентів.

- Використовувати інструменти аналізу складу програмного забезпечення для автоматизації процесу. Підписатися на сповіщення електронною поштою про вразливості безпеки, пов’язані з компонентами, які ви використовуєте.
- Отримувати компоненти лише з офіційних джерел за безпечними посиланнями. Віддавайте перевагу підписаним пакетам, щоб зменшити ймовірність включення модифікованого шкідливого компонента (див. A08:2021-Помилки програмного забезпечення та цілісності даних).
- Відстежуйте бібліотеки та компоненти, які не обслуговуються або не створюють виправлення безпеки для старіших версій. Якщо виправлення неможливо, розгляньте можливість розгортання віртуального виправлення для моніторингу, виявлення або захисту від виявленої проблеми.
- Кожна організація повинна забезпечити постійний план моніторингу, сортування та застосування оновлень або змін конфігурації протягом усього терміну служби програми чи портфоліо.

Сценарій №1: Компоненти зазвичай працюють з тими ж привілеями, що й сама програма, тому недоліки в будь-якому компоненті можуть призвести до серйозних наслідків. Такі недоліки можуть бути випадковими (наприклад, помилка кодування) або навмисними (наприклад, бекдор у компоненті). Деякі приклади виявлених уразливостей компонентів, які можна використовувати:

CVE-2017-5638, уразливість віддаленого виконання коду Struts 2, яка дозволяє виконувати довільний код на сервері, була звинувачена у значних порушеннях.

3.7 Способи захисту від Security Logging and Monitoring Failures (Помилки ідентифікації та аутентифікації)

Хороший спосіб перевірити на неадекватний ризик ведення журналу — це використовувати пентестер, який досліджуватиме і намагатиметься зламати ваші веб-програми.

Якщо ви не можете згодом виявити, що зроблено під час тестування, значить, ваш журнал є недостатнім.

Однак, що хоча неналежне ведення журналів і моніторинг є ризиком, адекватне ведення журналу не є рішенням. Вам все одно потрібно вміти розрізняти доброякісні та шкідливі аномалії в цих журналах.

Багато продуктів допоможуть зробити це, від давно зарекомендованих технологій, таких як IDS і IPS, до нових технологій керування логінами з розширеним штучним інтелектом і виявлення мережевих аномалій. Проблема в тому, що ви покладаєтесь на можливості моніторингу цих технологій. Те, що вони не виявляють проблеми, не означає, що проблеми немає. Але так само, як тестування на проникнення може використовуватися для підтвердження реєстраційної сторони ризику.

Журнали слід зберігати в безпеці, подалі від непотрібних облікових записів користувачів, які можуть їх редагувати, видаляти або пошкодити. Найкраще було б використовувати шифрування для центрального ведення журналів, але це може бути досить дорогим з точки зору продуктивності та персоналу. Найкраще використання журналів – перегляд після події; виявлення скомпрометованих ділянок та інфікованих пристроїв. Після очищення та виявлення вразливостей їх можна діяти та виправляти.

Якщо можливо, запровадьте багатофакторну аутентифікацію, щоб запобігти автоматичному підбору облікових даних, грубій силі та атак повторного використання вкрадених облікових даних.

Як запобігти:

- Не відправляти та не розгортайте з обліковими даними за замовчуванням, особливо для адміністраторів.

- Впроваджувати слабкі перевірки паролів, такі як тестування нових або змінених паролів у списку 10 000 найгірших паролів.
- Узгодьте політику довжини, складності та ротації пароля з рекомендаціями Національного інституту стандартів і технологій (NIST) 800-63b щодо запам’ятованих секретів або іншими сучасними, заснованими на доказах політиками паролів.
- Переконатися, що реєстрація, відновлення облікових даних і шляхи API захищені від атак перерахування облікових записів, використовуючи однакові повідомлення для всіх результатів.
- Обмежувати або все частіше відкладайте невдалі спроби входу, але будьте обережні, щоб не створити сценарій відмови в обслуговуванні. Реєструйте всі збої та сповіщайте адміністраторів у разі виявлення заповнення облікових даних, грубої сили чи інших атак.
- Використовувати безпечний, вбудований менеджер сеансів на стороні сервера, який генерує новий випадковий ідентифікатор сеансу з високою ентропією після входу в систему.
- Ідентифікатор сеансу не повинен бути в URL-адресі, бути надійно збережений та визнаний недійсним після виходу, простою та абсолютних тайм-аутів.

Сценарій №1. Заповнення облікових даних, використання списків відомих паролів, є поширеною атакою. Припустимо, програма не реалізує автоматичний захист від загроз або заповнення облікових даних. У цьому випадку програму можна використовувати як оракул пароля, щоб визначити, чи дійсні облікові дані.

Сценарій №2. Більшість атак аутентифікації відбуваються через постійне використання паролів як єдиного фактора.

Після розгляду найкращих практик вимоги до ротації паролів і складності спонукають користувачів використовувати та повторно використовувати слабкі паролі. Організаціям рекомендується припинити цю практику відповідно до NIST 800-63 і використовувати багатофакторну аутентифікацію.

Сценарій №3. Час очікування сеансу програми встановлено неправильно. Користувач використовує загальнодоступний комп'ютер для доступу до програми. Замість того, щоб вибрати «вийти», користувач просто закриває вкладку браузера та йде. Через годину зловмисник використовує той самий браузер, а користувач все ще проходить автентифікацію.

3.8 Способи захисту від Software and Data Integrity Failures (Помилки програмного забезпечення та цілісності даних)

Щоб скористатися незахищеною десеріалізацією, зловмисники зазвичай підробляють дані в структурах даних або об'єктах, які змінюють вміст об'єкта або змінюють логіку програми. Коли зловмисники досягають успіху, їхні експлойти можуть призвести до атак відмови в обслуговуванні (DoS), обходу автентифікації та атак віддаленого виконання коду**.**

Як запобігти:

- Використовуйте цифрові підписи або подібні механізми, щоб переконатися, що програмне забезпечення чи дані з очікуваного джерела та не були змінені.
- Переконайтеся, що бібліотеки та залежності, такі як npm або Maven, споживають надійні репозиторії.

- Якщо у вас високий профіль ризику, подумайте про розміщення внутрішнього відомого сховища, яке пройшло перевірку.
- Переконайтеся, що інструмент безпеки ланцюга постачання програмного забезпечення, такий як OWASP Dependency Check або OWASP CycloneDX, використовується для перевірки того, що компоненти не містять відомих уразливостей.
- Переконатися, що існує процес перевірки коду та змін конфігурації, щоб звести до мінімуму ймовірність того, що шкідливий код або конфігурація можуть бути введені в конвеєр програмного забезпечення.
- Переконатися, що ваш конвеєр CI/CD має належну сегрегацію, конфігурацію та контроль доступу, щоб забезпечити цілісність коду, що проходить через процеси збирання та розгортання.
- Переконатися, що непідписані або незашифровані серіалізовані дані не надсилаються ненадійним клієнтам без певної форми перевірки цілісності або цифрового підпису для виявлення фальсифікації або повторного відтворення серійних даних.

Сценарій №1. Оновлення без підпису: багато домашніх маршрутизаторів, телеприставок, мікропрограмного забезпечення пристроїв тощо не перевіряють оновлення за допомогою підписаного мікропрограмного забезпечення. Непідписане мікропрограмне забезпечення стає все більшою метою для зловмисників і, як очікується, буде тільки погіршуватися. Це є серйозним занепокоєнням, оскільки багато разів немає іншого механізму для виправлення, крім як виправлення в майбутній версії та очікування, поки попередні версії застаріють.

Сценарій №2 Шкідливе оновлення SolarWinds: відомо, що національні держави атакують механізми оновлення, причому нещодавньою помітною атакою стала атака SolarWinds Orion.

Компанія, яка розробляє програмне забезпечення, мала безпечні процеси створення та оновлення цілісності. Тим не менш, їх вдалося зруйнувати, і протягом кількох місяців фірма розповсюдила цілеспрямоване шкідливе оновлення серед понад 18 000 організацій, з яких близько 100 постраждали. Це одне з найбільш далекосяжних і найзначніших порушень такого характеру в історії.

Сценарій №3 Небезпечна десеріалізація: програма React викликає набір мікросервісів Spring Boot. Будучи функціональними програмістами, вони намагалися забезпечити незмінність їх коду. Рішення, яке вони придумали, — серіалізувати стан користувача та передавати його туди-сюди з кожним запитом. Зловмисник помічає сигнатуру об’єкта Java «rO0» (у base64) і використовує інструмент Java Serial Killer для віддаленого виконання коду на сервері програм

3.9 Способи захисту від Security Logging and Monitoring Failures (Регистрація та моніторинг збоїв безпеки)

Проблема недостатнього ведення журналів і моніторингу охоплює всю IT-інфраструктуру, а не лише Інтернет-додаток, як і рішення.

З цієї причини ми не будемо обмежувати це обговорення лише реєстрацією та моніторингом веб-програм.Рішення полягає в підвищеній автоматизації процесу. Наприклад, деякі системи контролю доступу можуть мати власні правила моніторингу.

Правила входу можна встановити, щоб дозволити попередньо визначену кількість спроб входу за сеанс. Система реєструє спроби, а потім блокує доступ з цієї IP-адреси на заздалегідь визначений період або на невизначений термін.

Розробники повинні впровадити деякі або всі наведені нижче засоби контролю, залежно від ризику програми:

- Переконатися, що всі помилки входу, контролю доступу та перевірки введення на стороні сервера можуть бути зареєстровані з достатнім контекстом користувача, щоб ідентифікувати підозрілі чи шкідливі облікові записи, і утримуватися протягом достатнього часу, щоб дозволити відкладений криміналістичний аналіз.
- Переконатися, що журнали створюються у форматі, який можуть легко використовувати рішення для керування журналами.
- Переконатися, що дані журналу закодовані правильно, щоб запобігти ін’єкціям або атакам на системи реєстрації чи моніторингу.
- Переконатися, що транзакції з високою вартістю мають контрольний журнал із засобами контролю цілісності, щоб запобігти підробці або видаленню, наприклад, таблиці бази даних лише для додавання тощо.
- Команди DevSecOps повинні налагодити ефективний моніторинг та оповіщення, щоб підозрілі дії виявлялися та швидко реагували на них.
- Створити або прийняти план реагування на інциденти та відновлення, наприклад Національний інститут стандартів і технологій (NIST) 800-61r2 або новішої версії.
- Існують комерційні та відкриті рамки захисту додатків, такі як основний набір правил OWASP ModSecurity, і програмне забезпечення для кореляції журналів з відкритим вихідним кодом, таке як стек Elasticsearch, Logstash, Kibana (ELK), які мають спеціальні інформаційні панелі та оповіщення.

Сценарій №1. Оператор веб-сайту постачальника дитячого плану охорони здоров’я не зміг виявити порушення через відсутність моніторингу та реєстрації.

Зовнішня сторона повідомила постачальника плану охорони здоров’я, що зловмисник отримав доступ та змінив тисячі конфіденційних даних про здоров’я понад 3,5 мільйонів дітей. Перевірка після інциденту показала, що розробники веб-сайту не усунули значні вразливості. Оскільки не було ведення журналу чи моніторингу системи, злом даних міг тривати з 2013 року, тобто протягом більше дев’яти років.

Сценарій № 2: Велика індійська авіакомпанія мала порушення даних, пов’язані з персональними даними мільйонів пасажирів на суму понад десять років, включаючи дані паспортів і кредитних карток. Злом даних стався у стороннього провайдера хмарного хостингу, який через деякий час повідомив авіакомпанію про порушення.

Сценарій №3: Велика європейська авіакомпанія зазнала порушення GDPR. Повідомляється, що порушення було спричинено вразливими місцями безпеки платіжних програм, якими скористалися зловмисники, які зібрали понад 400 000 записів про платежі клієнтів. Регулятор конфіденційності оштрафував авіакомпанію на 20 мільйонів фунтів стерлінгів.

3.10A10:2022-Server-Side Request Forgery**(**Підробка запиту на стороні сервера**)**

Найкращий спосіб захистити веб-застосунок від SSRF – не надсилати запити зовнішнім системам. Однак це не завжди можливо, особливо якщо це функція програми.

Розробники можуть запобігти SSRF, реалізувавши деякі або всі наведені нижче засоби контролю глибини захисту:

З мережевого рівня:

- Сегментуйте функціональні можливості віддаленого доступу до ресурсів в окремих мережах, щоб зменшити вплив SSRF
- Застосуйте політику брандмауера або правила контролю доступу до мережі, щоб блокувати весь трафік внутрішньої мережі, крім необхідного.

Підказки:

Встановити право власності та життєвий цикл для правил брандмауера на основі програм.

Реєструвати всі прийняті та заблоковані мережеві потоки на брандмауерах.

З рівня програми:

- Дезінфікуйте та перевіряйте всі вхідні дані, надані клієнтом
- Застосовувати схему URL-адрес, порт і призначення за допомогою позитивного списку дозволених
- Не надсилати клієнтам необроблені відповіді
- Вимкнути перенаправлення HTTP

Звернути увагу на узгодженість URL-адрес, щоб уникнути таких атак, як переприв’язування DNS та умови змагань «час перевірки, час використання». Не пом'якшуйте SSRF за допомогою списку заборон або регулярного виразу. Зловмисники мають списки корисного навантаження, інструменти та навички для обходу списків заборон.

Додаткові заходи, які слід враховувати:

- Не розгортайте інші служби безпеки на передніх системах (наприклад, OpenID). Контролювати локальний трафік на цих системах (наприклад, localhost)

- Для інтерфейсів із виділеними та керованими групами користувачів використовуйте мережеве шифрування (наприклад, VPN) на незалежних системах, щоб враховувати дуже високі потреби в захисті

Зловмисники можуть використовувати SSRF для атаки на системи, захищені брандмауерами веб-застосунків, брандмауерами або мережевими списками керування доступом, використовуючи такі сценарії, як:

Сценарій № 1. Внутрішні сервери сканування портів – якщо архітектура мережі несегментована, зловмисники можуть намітити внутрішні мережі та визначити, відкриті чи закриті порти на внутрішніх серверах за результатами підключення або за часом підключення чи відхилення підключень корисного навантаження SSRF.

Сценарій № 2: розкриття конфіденційних даних – зловмисники можуть отримати доступ до локальних файлів або внутрішніх служб, щоб отримати конфіденційну інформацію.

Сценарій № 3. Доступ до сховища метаданих хмарних служб. Більшість постачальників хмарних послуг мають сховище метаданих. Зловмисник може прочитати метадані, щоб отримати конфіденційну інформацію.

Сценарій №4: компрометація внутрішніх служб – зловмисник може зловживати внутрішніми службами для подальших атак, таких як віддалене виконання коду (RCE) або відмова в обслуговуванні (DoS).

ВИСНОВКИ

Останнім часом основними цілями для зловмисників стають веб-застосунки,в основному атаки виникають внаслідок вразливостей веб-застосунку і можуть привести то отримання контролю над ним і в наслідок цього може задати для користувача так і для організації шкоди різної степені тяжкості від втрати цінних даних так до грошових коштів.

Розглянуті основні вразливостей для веб-застосунки від введеня шкідливого коду до використання устарілих компонентів, а також представлено способи захисту від них.

ПЕРЕЛІК ПОСИЛАННЬ

[1] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/.

[2] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A01_2021-Broken_Access_Control/.

[3] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

[4] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A03_2021-Injection/

[5] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A04_2021-Insecure_Design/

[6] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

[7]OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

[8] OWASP [Електронний ресурс] – Режим доступу до ресурсу:https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

[9] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

[10] OWASP [Електронний ресурс] – Режим доступу до ресурсу:https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/

[11] OWASP [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/

[12] Wikipedia [Електронний ресур] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ARP

[13] Wikipedia [Елетронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/MAC-%D0%B0%D0%B4%D1%80%D0%B5%D1%81