НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання комп'ютерного практикуму

ДОСЛІДЖЕННЯ БЕЗПЕЧНОЇ РЕАЛІЗАЦІЇ ТА ЕКСПЛУАТАЦІЇ ДЕЦЕНТРАЛІЗОВАНИХ ДОДАТКІВ

Виконали студенти
групи ФІ-32мн
Мельник Ілля,
Міснік Аліна

Перевірила:
Селюх П.В.

Київ — 2024

Мета роботи: отримання навичок роботи із децентралізованими додатками та оцінка безпеки інформації при їх функціонуванні

Постановка задачі: Дослідити вимоги OWASP (безпека web-додатків) та скласти аналогічні вимоги для обраної системи децентралізованих додатків.

1 ХІД РОБОТИ

1.1 OWASP та її вимоги

Почнемо з того, що ж таке **OWASP**. **OWASP** – це некомерційна організація, яка спеціалізується на захисті веб-додатків. **OWASP Foundation** працює над підвищенням безпеки програмного забезпечення за допомогою своїх проектів програмного забезпечення з відкритим кодом під керівництвом спільноти, сотень відділень по всьому світу, десятків тисяч членів і проведення локальних і глобальних конференцій. А початок вона бере з недалекого 2001 року, поступово об'єднуючи різних спеціалістів, розробників та тестувальників в єдину спільноту покликану покращити рівень безпеки веб-додатків.

Спільною роботою цієї спільноти також є робота OWASP TOP 10. Це актуальний список із 10 загроз безпеки веб-додатків. Наразі існує декілька версій, що дозволяють навіть проглянути ієрархію, які загрози опускаються у рейтингу, які підіймаються, які з'являються нові, та ті, що зникають. Наразі найновішою версією є OWASP TOP 10 2021. Наразі ці 10 катергорій виглядають наступним чином:

1) Порушений контроль доступу (Broken Access Control)

2) Криптографічні збої (Cryptographic Failures)

3) Ін'єкція (Injection)

4) Ненадійний дизайн (Insecure Design)

5) Неправильна конфігурація безпеки (Security Misconfiguration)

6) Вразливі та застарілі компоненти (Vulnerable and Outdated Components)

7) Помилки ідентифікації та автентифікації (Identification and Authentication Failures)

8) Порушення цілісності програмного забезпечення та даних (Software and Data Integrity Failures)

9) Помилки реєстрації та моніторингу безпеки (Security Logging and Monitoring Failures)

10) Підробка запитів на стороні сервера (Server-Side Request Forgery (SSRF))

Далі пропонуємо коротко розібрати кожну загрозу, опишемо як запобігти їм (вимоги) та які сценарії атак бувають.

A01: Порушений конроль доступу

Контроль доступу забезпечує дотримання політики таким чином, щоб користувачі не могли діяти поза межами призначених дозволів. Збої зазвичай призводять до несанкціонованого розголошення інформації, модифікації чи знищення всіх даних або виконання бізнес-функцій за межами обмежень користувача.

Вимоги запобігання: Контроль доступу ефективний лише в довіреному серверному коді або безсерверному API, де зловмисник не може змінити перевірку контролю доступу або метадані. Повинно забезпечувати право власності на запис, а не приймати, що користувач може створювати, читати, оновлювати або видаляти будь-який запис.Реєструвати збої контролю доступу, сповіщати адміністраторів, коли це необхідно. API обмеження швидкості та доступ до контролера. Ідентифікатори сеансу з підтримкою стану мають бути визнані недійсними на сервері після виходу з системи. Токени JWT без стану повинні бути короткочасними, щоб мінімізувати вікно можливостей для зловмисника.

Атаки: Зловмисник переглядає усі можливі цільові сторінки (якщо будь-який не автентифікований користувач має доступ до всіх сторінок, якщо не адміністратор має доступ до привілейованих даних чи сторінок – це недолік).

A02: Криптографічні збої

Раніше відомий як розкриття конфіденційних даних, яке було скоріше широким симптомом, а не основною причиною, а тут увага зосереджується на збоях, пов'язаних із криптографією (або її відсутністю). Що часто призводить до розкриття конфіденційних даних. До відомих загальних слабких місць (CWE) входять CWE-259: використання жорстко

закодованого пароля , CWE-327: зламаний або ризикований алгоритм шифрування та CWE-331 недостатня ентропія.

Вимоги запобігання: Класифікуйте дані, які обробляються, зберігаються або передаються програмою. Визначте, які дані є конфіденційними відповідно до законів про конфіденційність, нормативних вимог або потреб бізнесу. Не зберігайте конфіденційні дані без потреби (дані які не зберігаються не можуть бути вкрадені). Забезпечте наявність актуальних і надійних стандартних алгоритмів, протоколів і ключів; використовувати правильне керування ключами. Шифруйте всі дані, що передаються, за допомогою безпечних протоколів, таких як TLS, пріоритезацією шифру сервером і безпечними параметрами. Забезпечте шифрування за допомогою таких директив, як HTTP Strict Transport Security (HSTS). Вимкніть кешування для відповідей, які містять конфіденційні дані. Зберігайте паролі, використовуючи надійні адаптивні та засолені функції хешування з робочим фактором (фактором затримки), наприклад 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). Завжди використовуйте автентифіковане шифрування замість простого шифрування. Уникайте застарілих криптографічних функцій і схем заповнення.

Атаки: 1. Сайт не використовує TLS або не підтримує TLS для всіх сторінок або підтримує слабке шифрування. Зловмисник відстежує мережевий трафік (наприклад, у незахищеній бездротовій мережі), знижує рівень підключень з HTTPS до HTTP, перехоплює запити та викрадає файл cookie сеансу користувача. Потім зловмисник повторно відтворює цей файл cookie та захоплює (автентифікований) сеанс користувача. 2. Програма шифрує номери кредитних карток у базі даних за допомогою автоматичного шифрування бази даних. Однак ці дані автоматично розшифровуються під час отримання, дозволяючи помилці впровадження SQL отримати номери кредитних карток у відкритому вигляді. 3. База

даних паролів використовує несолоні або прості хеші для зберігання паролів кожного. Помилка завантаження файлу дозволяє зловмиснику отримати базу даних паролів. Усі несолоні хеші можна розкрити за допомогою райдужної таблиці попередньо обчислених хешів. Хеші, згенеровані простими або швидкими хеш-функціями, можуть бути зламані графічним процесором, навіть якщо вони були засолені.

A03: Ін'єкція

Програма вразлива до атак, коли дані, надані користувачем, не перевіряються, не фільтруються та не очищаються програмою. Динамічні запити або непараметризовані виклики без контекстно-залежного екранування використовуються безпосередньо в інтерпретаторі. Ворожі дані використовуються в параметрах пошуку об'єктно-реляційного відображення (ORM) для отримання 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.

Вимоги запобігання: Бажаним варіантом є використання безпечного API, який повністю уникає використання інтерпретатора, надає параметризований інтерфейс або переходить на інструменти відображення об'єктів (ORM). Використовуйте позитивну перевірку введення на стороні сервера. Для будь-яких залишкових динамічних запитів екрануйте спеціальні символи, використовуючи спеціальний синтаксис екранування для цього інтерпретатора. Використовуйте LIMIT та інші елементи керування SQL у запитах, щоб запобігти масовому розкриттю записів у разі впровадження SQL.

Атаки: Сліпа довіра програми до фреймворків може призвести до запитів, які залишаються вразливими (наприклад, Hibernate Query Language (HQL).

A04: Ненадійний дизайн

Незахищений дизайн – це широка категорія, що представляє різні слабкі сторони, виражені як «відсутній або неефективний дизайн

контролю». Небезпечний дизайн не є джерелом для всіх інших 10 категорій ризику. Існує різниця між незахищеним дизайном і незахищеним впровадженням. Ми недаремно розрізняємо недоліки дизайну та дефекти впровадження, оскільки вони мають різні першопричини та способи їх усунення. Захищений дизайн все ще може мати дефекти реалізації, що призводять до вразливостей, якими можна скористатися. Незахищений дизайн не можна виправити ідеальною реалізацією, оскільки за визначенням необхідні засоби контролю безпеки ніколи не створювалися для захисту від конкретних атак. Одним із факторів, які сприяють незахищеному дизайну, є відсутність профілювання бізнес-ризиків, властивого програмному забезпеченню чи системі, що розробляється, і, отже, нездатність визначити, який рівень безпеки дизайну потрібен.

Вимоги запобігання: Встановіть і використовуйте безпечний життєвий цикл розробки разом із професіоналами AppSec, щоб допомогти оцінити та розробити засоби контролю безпеки та конфіденційності. Створіть і використовуйте бібліотеку безпечних шаблонів проектування. Використовуйте моделювання загроз для критичної автентифікації, контролю доступу, бізнес-логіки та потоків ключів. Напишіть модульні та інтеграційні тести, щоб підтвердити, що всі критичні потоки стійкі до моделі загроз. Розділіть рівні рівня на системному та мережевому рівнях залежно від потреб впливу та захисту. Обмежте споживання ресурсів користувачем або службою.

Атаки: 1. Робочий процес відновлення облікових даних може включати «запитання та відповіді», що заборонено NIST 800-63b, OWASP ASVS і OWASP Top 10. Запитання та відповіді не можна вважати доказом ідентичності кількох осіб. можуть знати відповіді, тому вони заборонені. Такий код слід видалити та замінити більш безпечним дизайном. 2. Мережа кінотеатрів надає знижки на групове бронювання та має максимум п'ятнадцять відвідувачів, перш ніж вимагати депозиту. Зловмисники можуть загрозливо змоделювати цей потік і перевірити, чи зможуть вони забронювати шістсот місць і всі кінотеатри одночасно за кілька запитів, що

призведе до величезної втрати доходу.

A05: Неправильна конфігурація безпеки

Програма може бути вразливою, якщо відсутнє посилення безпеки в будь-якій частині стеку програм або неправильно налаштовані дозволи для хмарних служб. Увімкнено або встановлено непотрібні функції (наприклад, непотрібні порти, служби). Облікові записи за замовчуванням і їхні паролі залишаються активними та не змінюються. Параметри безпеки на серверах додатків, платформах додатків не мають безпечних значень.

Вимоги запобігання: Переглянути та оновити конфігурації, що відповідають усім приміткам щодо безпеки, оновленням і виправленням у рамках процесу керування виправленнями (див. A06:2021-Уразливі та застарілі компоненти). Перегляньте дозволи на хмарне сховище (наприклад, дозволи на відро S3). Сегментована архітектура програми забезпечує ефективне та безпечне розділення між компонентами або орендарями за допомогою сегментації, контейнеризації або хмарних груп безпеки (ACL). Автоматизований процес перевірки ефективності конфігурацій і налаштувань у всіх середовищах.

Атаки: Список каталогу не вимкнено на сервері. Зловмисник виявляє, що може просто перерахувати каталоги. Зловмисник знаходить і завантажує скомпільовані класи Java, декомпілює їх і виконує зворотне проектування для перегляду коду. Потім зловмисник знаходить серйозну ваду контролю доступу в програмі.

A06: Вразливі та застарілі компоненти

Уразливі компоненти — це відома проблема, яку ми намагаємося протестувати та оцінити ризик, і це єдина категорія, яка не має загальних уразливостей і вразливостей (CVE), зіставлених із включеними CWE, тому використовується вага експлойтів/впливу за замовчуванням 5,0. До відомих CWE включено CWE-1104: використання необслуговуваних сторонніх компонентів і два CWE з Топ-10 2013 і 2017 років.

Вимоги запобігання: Видаляйте невикористовувані залежності, непотрібні функції, компоненти, файли та документацію. Постійно

інвентаризуйте версії як клієнтських, так і серверних компонентів (наприклад, фреймворків, бібліотек) і їхніх залежностей, використовуючи такі інструменти, як версії, OWASP Dependency Check, retire.js тощо. Постійно відстежуйте джерела, як-от загальні вразливості та експозиції (CVE) і національну базу даних про вразливості (NVD) щодо вразливостей у компонентах. Отримуйте компоненти лише з офіційних джерел через безпечні посилання. Слідкуйте за бібліотеками та компонентами, які не обслуговуються або не створюють виправлення безпеки для старіших версій. Якщо виправлення неможливо, подумайте про розгортання віртуального патча для моніторингу, виявлення або захисту від виявленої проблеми.

Атаки: Компоненти зазвичай працюють із тими ж привілеями, що й сама програма, тому недоліки будь-якого компонента можуть призвести до серйозних наслідків. Такі недоліки можуть бути випадковими (наприклад, помилка кодування) або навмисними (наприклад, бекдор у компоненті). Ось деякі приклади виявлених уразливостей компонентів, які можна використовувати: CVE-2017-5638, уразливість віддаленого виконання коду Struts 2, яка дозволяє виконувати довільний код на сервері, звинувачують у значних порушеннях; хоча Інтернет речей (IoT) часто важко або неможливо виправити, важливість їх виправлення може бути великою (наприклад, біомедичні пристрої).

A07: Помилки ідентифікації та автентифікації

Ця категорія , яка раніше була відома як «Порушена автентифікація». Включає перерахування загальних недоліків (CWE), пов'язаних із помилками ідентифікації. До відомих CWE включено CWE-297: неправильна перевірка сертифіката з невідповідністю хосту , CWE-287: неправильна автентифікація та CWE-384: фіксація сеансу.

Вимоги запобігання: Якщо це можливо, застосуйте багатофакторну автентифікацію, щоб запобігти автоматизованому підкиданню облікових даних, грубій форсації та атакам повторного використання вкрадених облікових даних. Не надсилайте та не розгортайте облікові дані за

замовчуванням, особливо для адміністраторів. Застосовуйте слабкі перевірки паролів, наприклад перевіряйте нові або змінені паролі зі списком 10 000 найгірших паролів. Узгодьте політику щодо довжини, складності та ротації пароля з інструкціями Національного інституту стандартів і технологій (NIST) 800-63b у розділі 5.1.1 щодо секретів, що запам'ятовуються, або іншими сучасними політиками паролів, заснованими на фактах. Обмежуйте або частіше відкладайте невдалі спроби входу, але будьте обережні, щоб не створити сценарій відмови в обслуговуванні. Реєструйте всі збої та сповіщайте адміністраторів, коли виявлено перекидання облікових даних, грубу силу чи інші атаки.

Атаки: 1. Підкидання облікових даних, тобто використання списків відомих паролів, є поширеною атакою. Припустімо, що програма не реалізує автоматичний захист від загроз або внесення облікових даних. У цьому випадку програму можна використовувати як оракул пароля, щоб визначити, чи облікові дані дійсні. 2. Час очікування сеансу програми встановлено неправильно. Користувач використовує загальнодоступний комп'ютер для доступу до програми. Замість вибору «вийти» користувач просто закриває вкладку браузера та йде геть. Зловмисник використовує той самий браузер через годину, а користувач усе ще аутентифікований.

A08: Порушення цілісності програмного забезпечення та даних

Нова категорія для 2021 року зосереджена на припущеннях, пов'язаних з оновленнями програмного забезпечення, критичними даними та конвеєрами CI/CD без перевірки цілісності. Один із найвищих зважених показників даних про загальну вразливість і ризики/загальну систему оцінки вразливостей (CVE/CVSS). Відомі переліки загальних недоліків (CWE) включають CWE-829: включення функціональності з ненадійної сфери управління , CWE-494: завантаження коду без перевірки цілісності та CWE-502: десеріалізація ненадійних даних.

Вимоги запобігання: Використовуйте цифрові підписи або подібні механізми, щоб переконатися, що програмне забезпечення або дані походять

з очікуваного джерела та не були змінені. Переконайтеся, що бібліотеки та залежності, такі як npm або Maven, використовують надійні репозиторії. Якщо у вас вищий профіль ризику, подумайте про розміщення перевіреного внутрішнього завідомо справного сховища. Переконайтеся, що інструмент безпеки ланцюга поставок програмного забезпечення, наприклад OWASP Dependency Check або OWASP CycloneDX, використовується для перевірки того, що компоненти не містять відомих уразливостей. Переконайтеся, що є процес перевірки коду та змін конфігурації, щоб мінімізувати ймовірність того, що зловмисний код або конфігурація можуть бути введені у ваш конвеєр програмного забезпечення. Переконайтеся, що непідписані або незашифровані серіалізовані дані не надсилаються ненадійним клієнтам без певної форми перевірки цілісності або цифрового підпису для виявлення підробки або повторного відтворення серіалізованих даних.

Атаки: 1. Оновлення без підпису: багато домашніх маршрутизаторів, приставок, мікропрограми пристроїв тощо не перевіряють оновлення за допомогою підписаної мікропрограми. Непідписане мікропрограмне забезпечення стає все більшою мішенню для зловмисників і, як очікується, буде тільки погіршуватися. Це викликає серйозне занепокоєння, оскільки багато разів немає іншого механізму для виправлення, крім виправлення в майбутній версії та чекання, поки попередні версії застаріють. 2. Небезпечна десеріалізація: програма React викликає набір мікросервісів Spring Boot. Будучи функціональними програмістами, вони намагалися забезпечити незмінність свого коду. Рішення, яке вони придумали, — це серіалізація стану користувача та передача його вперед і назад з кожним запитом. Зловмисник помічає підпис об'єкта Java «rO0» (у base64) і використовує інструмент Java Serial Killer, щоб отримати віддалене виконання коду на сервері додатків.

A09: Помилки реєстрації та моніторингу безпеки

Реєстрація і моніторинг можуть бути складними для тестування, що часто передбачає проведення інтерв'ю або запитання про те, чи були виявлені атаки під час тесту на проникнення. Даних CVE/CVSS для цієї

категорії небагато, але виявлення порушень і реагування на них є критично важливими. Тим не менш, це може бути дуже впливовим для підзвітності, видимості, оповіщення про інциденти та криміналістики. Ця категорія розширюється за межі CWE-778 Insufficient Logging і включає CWE-117 Неналежна нейтралізація вихідних даних для журналів , CWE-223 Пропущення інформації, що стосується безпеки , і CWE-532 Вставлення конфіденційної інформації до файлу журналу.

Вимоги запобігання: Переконайтеся, що всі помилки входу, контролю доступу та перевірки введення на стороні сервера можна реєструвати з достатнім контекстом користувача для ідентифікації підозрілих або зловмисних облікових записів і зберігати протягом достатнього часу, щоб забезпечити відкладений криміналістичний аналіз. Переконайтеся, що журнали генеруються у форматі, який легко можуть використовувати рішення для керування журналами. Переконайтеся, що дані журналу закодовані правильно, щоб запобігти ін'єкціям або атакам на системи реєстрації чи моніторингу. Переконайтеся, що транзакції з великою вартістю мають контрольний журнал із засобами контролю цілісності, щоб запобігти підробці або видаленню, наприклад, таблиці бази даних, які лише додаються, тощо. Команди DevSecOps повинні налагодити ефективний моніторинг і попередження, щоб підозрілі дії виявлялися та швидко реагували на них. Створіть або прийміть план реагування на інциденти та відновлення, наприклад Національний інститут стандартів і технологій (NIST) 800-61r2 або новіший.

Атаки: 1. Оператор веб-сайту постачальника дитячого плану охорони здоров'я не зміг виявити порушення через відсутність моніторингу та реєстрації. Зовнішня сторона повідомила постачальника плану медичного обслуговування, що зловмисник отримав доступ і змінив тисячі конфіденційних медичних записів понад 3,5 мільйонів дітей. Огляд після інциденту виявив, що розробники веб-сайту не усунули значні вразливості. Оскільки система не реєструвалася та не контролювалася, витік даних міг тривати з 2013 року, тобто протягом більше семи років. 2. У великої

індійської авіакомпанії було викрадено персональні дані мільйонів пасажирів за понад десять років, зокрема дані паспортів і кредитних карток. Витік даних стався стороннім провайдером хмарного хостингу, який через деякий час повідомив авіакомпанію про злам.

A10: Підробка запитів на стороні сервера

Помилки SSRF виникають щоразу, коли веб-програма отримує віддалений ресурс без перевірки наданої користувачем URL-адреси. Це дозволяє зловмиснику змусити програму надіслати створений запит до неочікуваного адресата, 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. Як наслідок, захворюваність на SSRF зростає. Крім того, серйозність SSRF стає вищою через хмарні сервіси та складність архітектур.

Вимоги запобігання: Розділіть функції віддаленого доступу до ресурсів у окремі мережі, щоб зменшити вплив SSRF. Застосуйте політику брандмауера «відмовити за замовчуванням» або правила контролю доступу до мережі, щоб блокувати весь інтрамережевий трафік, крім основного.

Атаки: 1. Сканування портів внутрішніх серверів – якщо архітектура мережі не сегментована, зловмисники можуть нанести на карту внутрішні мережі та визначити, чи відкриті чи закриті порти на внутрішніх серверах, виходячи з результатів підключення або часу, що минув для підключення чи відхилення з'єднань корисного навантаження SSRF. 2. Викриття конфіденційних даних – зловмисники можуть отримати доступ до локальних файлів або внутрішніх служб, щоб отримати конфіденційну інформацію, таку як file:///etc/passwdi http://localhost:28017/. 3. Доступ до сховища метаданих хмарних служб – більшість хмарних постачальників мають сховище метаданих, наприклад http://169.254.169.254/. Зловмисник може прочитати метадані, щоб отримати конфіденційну інформацію. 4. Компрометація внутрішніх служб – зловмисник може зловживати внутрішніми службами для подальших атак, таких як Remote Code

Execution (RCE) або Denial of Service (DoS).

1.2 Децетралізовані додатки

Як і зрозуміло з назви, децентралізовані додатки працюють у децентралізованій мережі. Так, dApps є схожими на звичайні додатки з вашого мобільного телефону, але вони мають зовсім іншу серверну систему. Функціонування децентралізованих додатків підпорядковується смарт-контрактам не в централізованій системі, а в розподіленій мережі. А ще тут використовується блокчейн, який дозволяє обробляти інформацію та виконувати транзакції через розподілені мережі. Часто dApps створюють на основі платформи Ethereum (як-от відомий децентралізований додаток Uniswap (DEX). dApps мають схожість зі звичними додатками, бо вони користуються однаковим кодом інтерфейсу, потрібним для відтворення сторінки сайту (Front-End). Хоча й тут є відмінність: внутрішні коди dApps відрізняються, тому й вони вільні від контролю людиною чи організацією.

Хоча й інтерфейси традиційних додатків та dApps схожі, останні мають ряд переваг з централізованими.

1) У порівнянні з централізованими, основою dApps є розподілені мережі без центральної влади. Завдяки цьому вони є менш вразливими до атак, тому зловмисникам набагато складніше зламати децентралізовану мережу.

2) Користувачі мають більший контроль над даними, що вони обмінюють, завдяки децентралізованому характеру.

3) Користувачам не потрібно давати реальну особисту інформацію для роботи з dApps, бо компанії не контролюють дані користувачів. Замість цього клієнти часто використовують криптогаманці, щоб підключитися до децентралізованого додатку та контролювати дані, що вони обмінюють.

4) Розробники без проблем інтегрують криптовалюту в основні функції dApp з використанням смарт-контрактів. Для прикладу, децентралізовані додатки на основі Ethereum мають можливість

користуватися ETH у вигляді платежу та не інтегрувати інших постачальників.

Яким вимогам повинні відповідати централізовані додатки?

Розглядаючи це питання, ми повинні зрозуміти, в якій сфері працює наш додаток. Таким чином, ми сформувати що саме він буде в себе включати і на які аспекти варто звернути увагу. Зрозуміло, що є загальні властивості, які притаманні будь-якому додатку в незалежності від його профілю діяльності. Тому варто розглядати увесь перелік можливих і небезпечних вразливостей.

Розглянемо додаток Augur (система прогнозування на основі смарт-контрактів блокчейну Etherium). Перш за все він повинен мати правильний контроль доступу, тобто використовуваати довірений серверний код чи безсерверний API. Конфіденційні дані повинні надійно шифруваатися, а непотрібні дані не зберігатися, щоб убезпечити дані користувачів від розкриття при можливому витоку. Використовувати обмеження та інші елементи керування SQL у запитах, щоб запобігти масовому розкриттю записів у разі впровадження SQL. Також додаток повинен мати надійний дизайн, оскільки це важливо для уникнення багатьох загроз, що можуть виникати через прогалини у вашому проектуванні. До прикладу той самий процес відновлення облікових даних. Необхідно постійно підтримувати конфігурації безпеки та слідкувати за вчасними оновленнями, тут же необхідно проводити і інвентарізацію версій клієнтських та серверних компонентів і їх залежності. За необхідності випускати патчі для оновлення. За можливості чудово було б використовувати двофакторну автентифікацію, встановити політику довжин пароля, контролювати часті невдалі спроби входу в облікові записи. Ще важливим критерієм буде порушення цілісностсі програмного забезпечення та даних, контроль логів та їх генерація, а також використовувати політику брандмауера «відмовити за замовчуванням» або правила контролю доступудо мережі, щоб блокувати весь інтрамережевий трафік, крім основного для зпобігання SSRF.

ВИСНОВКИ

В даній роботі ми розглянули основні вимоги OWASP до вразливостей топ-10, що є наразі актуальним списком найпоширеніших загроз для додатків. Якщо підходити до розробки додатку комплексно і відповідально, то це доволі складна робота, що повинна включати багато тестів. Але, оцінка і запобігання більшості загроз на початку призведе до економії коштів, які необхідно буде витрачати на покриття наслідків витоку інформації чи злому роботи системи.

Також варто відмітити, що більшість властивостей дуже актувальні для децентралізованих додатків, а тому при їх розробці варто звертати особливу увагу на ці властивості для запобігання можливих загроз.