Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи магістра

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: *«Метод аналізу веб-експлойтів на основі JavaScript»*

Виконавець: студент ІІ курсу, групи КБм-21

Денис ПТАШИНСЬКИЙ
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Сергій БУЧИК	
Нормоконтроль	Юрій ЩЕБЛАНІН	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-21 (група) Пташинського Дениса Володимировича (прізвище ім'я по-батькові)

Тема дипломної роботи Метод аналізу веб-експлойтів на основі JavaScript

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 3 від 20.10.2022

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес аналізу веб-експлойтів
Предмет досліджень	Методи аналізу веб-експлойтів на основі JavaScript
Мета	Розробка методу аналізу веб-експлойтів на основі JavaScript
Вихідні дані для проведення роботи	Методи аналізу веб-експлойтів на основі JavaScript

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна Удосконалення методу аналізу веб-експлойтів за рахунок перевірки основних уразливостей та недоліків лише з критичним рівнем ризику

Практична цінність Покращення системи захисту інформації у організаціях різних форм власності.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Розробка плану для досягнення мети роботи	24.10.2022 – 23.01.2023
Аналіз літературних джерел	24.01.2023 – 14.02.2023
Розробка методу аналізу веб-експлойтів на основі JavaScript	15.02.2023 – 24.04.2023
Оформлення і друк пояснювальної записки	25.04.2023 – 19.05.2023

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через викрадення даних

Соціальний ефект Покращення технологій забезпечення захисту інформації в організаціях різних форм власності

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Сергій БУЧИК
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Денис ПТАШИНСЬКИЙ
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 24.10.2022 р.
Термін подання кваліфікаційної роботи до ЕК 19.05.2023 р.

УДК. 004.432.16

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної магістерської роботи «Метод аналізу веб-експлойтів на основі JavaScript»: 71 сторінка, 10 рисунків та 24 літературних джерела.

Об’єкт дослідження – процес аналізу веб-експлойтів

Мета роботи – розробка методу аналізу веб-експлойтів на основі JavaScript.

Методи дослідження – методи аналізу веб-експлойтів на основі JavaScript.

У роботі досліджено сучасні загрози та методи протидії веб-експлойтам. Проведено аналіз існуючих інструментів для аналізу веб-додатків на вразливості.Запропоновано метод аналізу веб-експлойтів.

Наукова новизна: удосконалено метод аналізу веб-експлойтів за рахунок перевірки основних уразливостей та недоліків лише з критичним рівнем ризику.

Актуальність теми: У темі кібербезпеки обговорення аналізу веб-експлойтів є досить важливим. Популярною мовою програмування для створення динамічних та інтерактивних онлайн-додатків є JavaScript. На жаль, ця популярність також робить його головною мішенню для хакерів, які використовують недоліки для здійснення різноманітних атак. Загалом, важливість вивчення веб-експлойтів випливає з необхідності зрозуміти, ідентифікувати та протидіяти зростаючій кількості кіберзагроз, які це роблять. Це може допомогти створити стратегії, безпечні стандарти кодування та механізми захисту, які захистять користувачів і веб-додатки від злочинної діяльності.

Ключові слова: веб-сервер, веб-експлойти, вразливості, JavaScript, система захисту інформації, захист даних.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ІБ – Інформаційна система;

СЗІ – Система захисту інформації;

ПЗ – Програмне забезпечення

АСУ – Автоматизована система управління

HTTP – HyperText Transfer Protocol

CPU – Central processing unit

SSL – Secure Sockets Layer

TLS – Transport layer security

ОС – Операційна система

HTML – Hypertext Markup Language

ID – Identifier

SQL –Structured Query Language

URL – Uniform Resource Locator

XSS – Cross-site Scripting

IP – Internet Protocol

DOM – Document Object Model

SQL – Structured Query Language

CSRF – Cross-site request forgery

ВСТУП

У двадцять першому столітті основним об’єктом усіх галузей людської діяльності є інформація. Існує думка, що інформація стає основною міжнародною валютою. Але, на жаль, мережеві технології досить вразливі до цілеспрямованих атак. Крім того, такі атаки можуть здійснюватися дистанційно, в тому числі з-за меж національних кордонів. Все це створює нові проблеми для розробників. Деякі сучасні форми бізнесу повністю базуються на мережевих технологіях (електронна комерція, IP-телефонія, мережевий провайдер тощо), і з цієї причини вони особливо вразливі.

Сьогодні веб-сайти вже не просто призначені для «присутності в Інтернеті», але також використовуються для комерційних операцій і передачі конфіденційних даних. Таке широке використання допомагає зломщикам отримати більше знань про вразливості та методи експлуатації, ніж раніше. Різноманітні дослідження безпеки показують, що атаки на веб-сайти з метою отримання слави чи грошей безперечно зростають. Також існує велика кількість методів, які можуть бути використані системними адміністраторами для захисту веб-інфраструктури.

Оскільки веб-сайти останнім часом використовуються в широкому спектрі додатків, збитки, які завдаються наборами веб-експлойтів, також збільшується.

Мета магістерської роботи: розробка методу аналізу веб-експлойтів на основі JavaScript.

Об’єкт дослідження: процес аналізу веб-експлойтів.

Предмет долідження: методи аналізу веб-експлойтів на основі JavaScript

Завдання магістерської роботи:

- Дослідити принципи функціонування веб-серверів;
- Дослідити значення веб-експлойтів;
- Дослідити та проаналізувати інструменти для аналізу веб-експлойтів;
- Розробити метод аналізу веб-експлойтів на основі JavaScript

Наукова новизна: удосконалено метод аналізу веб-експлойтів за рахунок перевірки основних уразливостей та недоліків лише з критичним рівнем ризику.

Актуальність теми. У темі кібербезпеки обговорення аналізу веб-експлойтів є досить важливим. Популярною мовою програмування для створення динамічних та інтерактивних онлайн-додатків є JavaScript. На жаль, ця популярність також робить його головною мішенню для хакерів, які використовують недоліки для здійснення різноманітних атак. Загалом, важливість вивчення веб-експлойтів випливає з необхідності зрозуміти, ідентифікувати та протидіяти зростаючій кількості кіберзагроз, які це роблять. Це може допомогти створити стратегії, безпечні стандарти кодування та механізми захисту, які захистять користувачів і веб-додатки від злочинної діяльності.

Апробація результатів роботи:

Бучик С. Розуміння загроз для веб-сайтів: веб-експлойти та захист від них / С. Бучик, Д. Пташинський // Проблеми кібербезпеки ін-формаційно-телекомунікаційних систем: Збірник матеріалів доповідей та тез; м. Київ, 27 квітня 2023 року; Київський національний університет імені Тараса Шевченка / Редкол.: В.В. Ільченко, д.ф-м.н., проф., (голова); та ін. – К.: ВПЦ "Київський університет", 2023. – 166 с. (с.76-78)

РОЗДІЛ 1
ЗАГАЛЬНИЙ ОГЛЯД ВЕБ-СЕРВЕРУ, ВЕБ-ЕКСПЛОЙТІВ ТА МОВИ ПРОГРАМУВАННЯ JAVASCRIPT

1.1 Аналіз принципів функціонування веб-серверів

Веб-сервери розміщують веб-сайти, веб-сервер — це комп'ютер, на якому працює операційна система та зберігає файли сайту (HTML-сторінки, стилі CSS, сценарії JavaScript і зображення) перед доставкою їх у веб-браузер кінцевого користувача.

Веб-сервери призначені для отримання даних із баз даних і надання їх у визначеному форматі HTML, вони також можуть спілкуватися з базами даних. З точки зору програмного забезпечення, веб-сервер складається з різноманітних частин, таких як сервер HTTP, який відповідає за надання користувачам доступу до ресурсів сервера [1]. Частина програмного забезпечення, яка охоплює HTTP та веб-адреси (URL-адреси), називається HTTP-сервером.

Важливу частину загальної архітектури відіграє HTTP, який є протоколом, відповідальним за надсилання даних від клієнта до сервера та навпаки. Сервери зберігають веб-сторінки та подають їх клієнту за запитом, обробленим через HTTP [2]. Хакери, які націлюються на веб-сервери різними способами, обов'язково націляться на мережу. Через це атака на веб-сервер буде результатом будь-якої слабкості програм, бази даних, операційної системи чи мережі. Атака на WEB-сервер передбачає втручання в звичайні операції вузла, видалення або зміну його вмісту або отримання доступу до машини зі спеціальними привілеями [3].

Аналіз поточних мережевих атак на веб-сервер і контрзаходи необхідні, враховуючи різноманітність технологій, що використовуються в компонентах веб-сервера.

1.2 Визначення веб-експлойту

Веб-експлойт, також відомий як експлойт веб-програми - це загальний термін, що описує будь-який тип атаки, спрямованої на веб-програми або веб-сайти, відноситься до техніки або методу, який використовують хакери, для досягнення певної мети, наприклад отримання несанкціонованого доступу, викрадення конфіденційної інформації або завдання шкоди системі [4].

Веб-експлойти стосуються будь-яких зловмисних дій або вразливостей, націлених на веб-програми, веб-сайти та веб-сервери. Ці експлойти можуть варіюватися від простих атак, таких як міжсайтовий сценарій (XSS), до більш складних, таких як SQL-ін’єкція та віддалене виконання коду.

Веб-експлойти можна загалом класифікувати на дві категорії: клієнтські та серверні. Експлойти на стороні клієнта спрямовані на цільові вразливості в браузері користувача або клієнтському коді веб-програми [5]. Ці експлойти часто використовують уразливості в застарілих або невиправлених веб-переглядачах або плагінах, таких як Adobe Flash або Java.

З іншого боку, серверна сторона використовує цільові вразливості в серверному коді веб-додатку або веб-сервера. Ці експлойти можуть призвести до того, що зловмисник отримає несанкціонований доступ до конфіденційних даних або захопить контроль над самим сервером. Поширені експлойти на стороні сервера включають впровадження SQL, віддалене виконання коду та обхід каталогу [6].

Веб-експлойти можуть мати серйозні наслідки, включаючи крадіжку даних, компрометацію системи та фінансові втрати. Вони також можуть завдати шкоди репутації та втрати довіри клієнтів, особливо для компаній, які обробляють конфіденційну інформацію клієнтів.

Для захисту від веб-експлойтів надзвичайно важливо впровадити надійні заходи безпеки, такі як брандмауери веб-додатків, регулярні оновлення програмного забезпечення та тестування на проникнення. Крім того, веб-розробників слід навчити безпечному кодуванню та бути в курсі останніх тенденцій веб-безпеки та вразливостей.

Веб-експлойти стають все більш поширеними, оскільки все більше компаній переводять свою діяльність в Інтернет. Це призвело до зростання кількості кіберзлочинців, які націлюються на веб-додатки та веб-сайти в пошуках уразливостей для використання.

Одним із найпоширеніших типів веб-експлойтів є міжсайтовий сценарій (XSS), який передбачає впровадження шкідливого коду на веб-сторінку, яку переглядають інші користувачі. Це може призвести до викрадення конфіденційної інформації, як-от облікові дані для входу, або перенаправлення користувача на шкідливий веб-сайт.

Іншим поширеним типом веб-експлойту є SQL-ін’єкція, яка передбачає вставку зловмисних команд SQL у вразливу веб-програму. Це може призвести до того, що зловмисник отримає несанкціонований доступ до конфіденційних даних, що зберігаються в базі даних веб-програми.

Віддалене виконання коду (RCE) — ще один тип веб-експлойту, який можна використовувати для контролю над сервером. Експлойти RCE можуть бути особливо руйнівними, оскільки вони можуть дозволити зловмиснику виконати довільний код на сервері, що може призвести до крадіжки даних або знищення критичних файлів.

Веб-експлойти можна пом’якшити за допомогою поєднання технічних і нетехнічних заходів.

Технічні заходи включають впровадження таких функцій безпеки, як брандмауери та системи виявлення вторгнень, а також регулярне оновлення програмного забезпечення та використання методів безпечного кодування.

Нетехнічні заходи включають навчання співробітників найкращим практикам веб-безпеки, наприклад, остерігатися підозрілих електронних листів і не натискати невідомі посилання чи вкладення. Регулярні перевірки безпеки та тестування на проникнення також можуть допомогти виявити вразливі місця у веб-додатках і веб-сайтах, перш ніж ними зможуть скористатися зловмисники.

Веб-експлойти можна класифікувати на основі вектора атаки, який може включати атаки на стороні клієнта, атаки на стороні сервера та атаки на основі мережі [7].

Атаки на стороні клієнта включають експлойти, спрямовані на вразливості у веб-браузерах, плагінах та іншому програмному забезпеченні на стороні клієнта. Ці атаки можуть включати міжсайтовий сценарій (XSS), експлойти браузера та вразливості формату файлу.

Атаки на стороні сервера включають експлойти, спрямовані на вразливості веб-серверів, веб-додатків і баз даних. Ці атаки можуть включати SQL-ін’єкцію, віддалене виконання коду (RCE) і вразливості включення файлів. Мережеві атаки включають експлойти, спрямовані на саму мережеву інфраструктуру, наприклад атаки на відмову в обслуговуванні (DoS), атаки типу 'людина посередині' (MitM) і перехоплення пакетів.

Веб-експлойт може мати серйозні наслідки для компаній і організацій, зокрема втрату прибутку, шкоду репутації та юридичну відповідальність. У зв’язку з цим важливо запровадити комплексну стратегію безпеки, яка включає регулярні оцінки вразливостей, тестування на проникнення та постійне навчання співробітників найкращим практикам веб-безпеки.

Щоб захиститися від веб-експлойтів, компанії повинні переконатися, що все програмне забезпечення та системи оновлені з останніми виправленнями безпеки та що будь-які відомі вразливості негайно усуваються. Це включає регулярні перевірки безпеки веб-додатків, баз даних і веб-серверів [8].

Крім того, компанії повинні використовувати брандмауери веб-додатків (WAF) для захисту від поширених веб-експлойтів, таких як впровадження SQL і міжсайтовий сценарій.

WAF також можна використовувати для виявлення та блокування підозрілого трафіку, запобігаючи використанню зловмисниками вразливостей у веб-додатках і серверах.

1.3 Роль JavaScript у веб-експлойтах

JavaScript - це універсальна мова сценаріїв, яка широко використовується у веб-розробці для додавання інтерактивності та динамічної функціональності веб-сторінок [9]. Однак його гнучкість і повсюдність також роблять його популярним

інструментом для зловмисників для здійснення веб-експлойтів. JavaScript може відігравати значну роль у веб-експлойтах як інструмент, який використовують зловмисники, так і як ціль для атак.

Як інструмент, який використовують зловмисники, JavaScript можна використовувати для здійснення таких атак, як міжсайтовий сценарій (XSS) і Clickjacking.

XSS-атаки включають введення шкідливого коду JavaScript на веб-сторінку, який потім виконується браузером жертви, коли вона відвідує сторінку. Цей код можна використовувати для викрадення конфіденційної інформації, наприклад облікових даних або номерів кредитних карток, або для викрадення сеансу користувача.

Clickjacking - ще один тип веб-експлойту на основі JavaScript. Атаки клікджекінгу передбачають накладання зловмисної веб-сторінки на законну сторінку, обманом змушуючи користувача натискати кнопки чи посилання, які вони не мали на меті. Ці атаки можуть використовуватися для викрадення конфіденційної інформації, здійснення шахрайських транзакцій або виконання шкідливого коду в системі жертви.

Код JavaScript також може містити вразливості, якими можуть скористатися зловмисники. Наприклад, якщо перевірка вхідних даних реалізована неправильно, код JavaScript може бути вразливим до атак переповнення буфера або атак впровадження коду. Ці атаки можуть використовуватися для виконання довільного коду в системі жертви, що призводить до крадіжки даних або компрометації системи [10].

Щоб зменшити ризик веб-експлойтів на основі JavaScript, важливо дотримуватися правил безпечного кодування під час розробки веб-додатків, які використовують JavaScript. Це включає впровадження перевірки введених даних і санітарної обробки, використання безпечних бібліотек і фреймворків, а також підтримку програмного забезпечення веб-додатків в актуальному стані за допомогою останніх виправлень безпеки.

Брандмауери веб-додатків та інші заходи безпеки також можна використовувати для виявлення та запобігання веб-експлойтам на основі JavaScript. Наприклад, брандмауери веб-додатків можна налаштувати для блокування запитів, які містять потенційно шкідливий код JavaScript.

JavaScript є потужним інструментом для створення веб-програм на стороні клієнта, але він також представив кілька проблем безпеки, які привертають багато уваги. Зловмисники можуть вводити сценарії через інтерфейс веб-додатків і виконувати їх на комп'ютері клієнта завдяки тому як JavaScript взаємодіє з об'єктною моделлю документа (DOM) [11].

Цього можна уникнути за допомогою однієї з двох стратегій. Перший із них призводить до розробки пісочниць (пісочниці) або окремого виконання сценаріїв, що дозволяє отримати доступ лише до визначених ресурсів і дозволяє виконувати лише визначені завдання. Друга стратегія полягає в тому, щоб запровадити політику обмеження домену, яка забороняє сценаріям з одного сайту отримувати доступ до даних, які використовуються та обробляються сценаріями з іншого сайту.

Через погано продуманий дизайн веб-браузерів і відсутність рішень для пом'якшення загроз безпеці, спричинених об'єктною моделлю документа, JavaScript має низку недоліків безпеки. Відсутність відповідної перевірки вхідних і вихідних даних часто призводить до вразливостей у клієнтській частині веб-програми, які потім можуть бути використані для зміни вихідного коду програми або отримання несанкціонованого доступу.

Крім того, використання сучасних бібліотек і фреймворків у розробці програмного забезпечення пов'язане з появою нових уразливостей, які, окрім простоти структури, відкривають нові можливості для маніпулювання веб-додатками та даними, які вони обробляють.

Прикладом можуть бути атаки, які використовують слабкі місця в прив'язках даних і шаблонах, які надають фреймворки, а також відомі проблеми в бібліотеках і модулях, які складають ці продукти.

1.4 Аналіз спільної системи оцінювання вразливостей CVSS 3.0

Відкритою системою для опису характеристик і серйозності вразливостей програмного забезпечення є Загальна система оцінки вразливостей (CVSS). Три групи показників складають CVSS: Base, Temporal і Environmental [12].

Базові метрики представляють ознаки вразливості, які не залежать від середовища виконання та не змінюються з часом. Ці показники пояснюють, наскільки важко використати вразливість і можливу шкоду для конфіденційності, цілісності та доступності інформації.

Тимчасові метрики, як випливає з їх назви, змінюють загальну оцінку для врахування зрілості експлуатаційного коду (якщо такий є), доступності засобів правового захисту та повноти інформації, яка наразі доступна про вразливість.

За допомогою контекстних метрик експерти з безпеки можуть внести до результуючої оцінки виправлення з урахуванням характеристик інформаційного середовища.

Тимчасові та контекстні метрики є опціональними і застосовуються для більш точної оцінки небезпеки, яку представляє дана вразливість для більш-менш конкретної інфраструктури.

Значення метрики прийнято публікувати як пари з вектора (конкретні значення окремих показників) і числового значення, розрахованого з урахуванням всіх показників з допомогою формули, визначеної стандарті [13].

Оцінка від 0 до 10 забезпечується базовими показниками, які потім можна змінити, дивлячись на часові маркери та маркери екологічної оцінки. Векторний рядок, стисле текстове представлення значень, що використовуються для обчислення оцінок, є ще одним способом представлення оцінки CVSS.

1.5 Основні типи атак на веб-додатки та приклади сценаріїв атак

1.5.1 SQL-ін’єкції

База даних є дуже зручною структурою для зберігання інформації. У більшості випадків доступ до бази даних здійснюється за допомогою спеціальної мови запитів - SQL, яка реалізована за допомогою інтерпретатора. Інтерпретована мова — це мова,

реалізація якої включає компонент часу виконання, який інтерпретує код мови та виконує інструкції, що містяться в ньому [14]. Через спосіб інтерпретації мови існує група вразливостей, відома як впровадження коду. Можливість впровадження виникає, коли недостовірні дані надсилаються інтерпретатору як частина команди чи запиту.

Процес доступу програми до сховища даних однаковий незалежно від того, чи ініціює доступ непривілейований користувач чи адміністратор програми. Веб-програма діє як дискреційний контроль доступу до сховища даних, надаючи запит на отримання, додавання або зміну даних у базі даних на основі облікового запису та типу користувача.

Успішна ін’єкційна атака, яка змінює запит (а не лише дані в запиті), може обійти дискреційний контроль доступу програми та отримати неавторизований доступ. І якщо логіка програми контролюється результатами запиту, зловмисник може змінити логіку програми, змінивши запит.

Поля введення веб-програми можуть бути використані зловмисником для впровадження зловмисного коду SQL у запит до бази даних, що є вразливістю безпеки. У результаті зловмисник може викрасти конфіденційні дані, отримати доступ або змінити дані бази даних, а також вчинити інші нечесні дії.

Коли онлайн-додатки не перевіряють належним чином або не очищають дані користувача, виникає вразливість. Наприклад, веб-програма може дозволити користувачеві заповнити форму своєю реєстраційною інформацією або пошуковим запитом. Зловмисник може додати зловмисний код SQL як частину вхідних даних і змусити програму виконати введений код як частину запиту, якщо програма наївно вставить введені користувачем дані в запит до бази даних без відповідної перевірки [15].

Введений код може приймати різні форми, наприклад оператори SELECT для вилучення конфіденційних даних, оператори DELETE для знищення даних або навіть оператори DROP TABLE для видалення цілих таблиць із бази даних.

Приклад сценарію атаки:

Уявіть, що є веб-сайт, на якому користувачі можуть використовувати панель пошуку для пошуку продуктів у базі даних. Спосіб написання пошукового запиту дозволяє негайно вводити введені користувачем дані в оператор SQL без будь-якої перевірки чи очищення. Надсилаючи зловмисний ввід, який містить команди SQL, зловмисник може скористатися цією вразливістю.

Наприклад, припустимо, що користувач вводить такий рядок у рядок пошуку:

' OR 1=1; --

Цей напис може виглядати нешкідливим на перший погляд, але зловмисник може використати його для створення SQL-запиту, який виглядає так:

SELECT * FROM products WHERE name='' OR 1=1; --'

Подвійне тире (--) позначає початок коментаря SQL, який, по суті, ігнорує решту тексту запиту. У цьому сценарії остаточна цитата та будь-які додаткові символи, які інакше призвели б до помилки, не врахували б.

Цей запит повертає всі рядки з таблиці продуктів, оскільки умова АБО 1=1 завжди має значення істини. Використовуючи цей метод, хакер може редагувати або знищувати існуючі дані, витягувати конфіденційну інформацію з бази даних або навіть отримати доступ до сервера без авторизації.

1.5.2 CSRF-атака

Підробка міжсайтових запитів (CSRF) – це атака, яка дозволяє зловмиснику змусити необережного користувача виконувати небажані дії на веб-сайті без його відома чи згоди. CSRF-атака полягає в тому, що ціль обманом змушує її натиснути посилання або перейти на шкідливий веб-сайт, який надсилає неавторизовані запити на вразливий веб-сайт.

Використовуючи довіру, яка існує між браузером користувача та передбачуваним веб-сайтом, атака є успішною. Зазвичай веб-сайт встановлює сеанс для користувача, коли він входить, і зберігає токен сеансу в браузері користувача. Подальші запити користувача перевіряються за допомогою цього токену [16].

CSRF-атака – це коли зловмисник створює веб-сторінку зі зловмисним кодом, який містить запит до вразливого веб-сайту, а також токен сеансу, зібраний із браузера жертви. Шкідливий код автоматично надсилає запит до вразливого веб-сайту кожного разу, коли жертва відвідує веб-сайт зловмисника, діючи так, ніби він походить із браузера жертви.

У результаті слабкий веб-сайт неправильно приймає запит як дійсний і виконує запитану дію, наприклад змінює пароль жертви, переміщує гроші з рахунку жертви або видаляє дані жертви.

Приклад сценарію атаки:

1. Сеанс Боба транслюється на веб-сайті, де він здійснює онлайн-банкінг.

2. Боб отримує електронний лист із посиланням на веб-сайт від ворожого зловмисника, який видавав себе за надійне джерело.

3. Коли Боб натискає посилання, відвідується веб-сайт зловмисника. Цей веб-сайт містить зловмисний код, який без відома або схвалення Боба автоматично надсилає форму на веб-сайт онлайн-банкінгу від імені Боба.

4. У формі міститься запит на переказ грошей з рахунку Боба на рахунок зловмисника.

5. Запит перевірено та оброблено без подальших перевірок автентифікації, оскільки Боб уже ввійшов на веб-сайт банку.

6. Боб ніколи не дізнається про крадіжку з його облікового запису зловмисником, який досяг успіху.

У цьому випадку зловмисник обманом змусив веб-браузер Боба надіслати запит на веб-сайт онлайн-банкінгу від його імені, обійшовши процедури автентифікації веб-сайту.

1.5.3 XSS-атака

Міжсайтовий сценарій, або XSS, є різновидом уразливості безпеки, який присутній у веб-додатках. Зловмисний фрагмент коду, як правило, у формі JavaScript, вставляється на веб-сторінку, яку переглядають інші користувачі під час атаки XSS [17]. Вставлений код запускається в браузерах відвідувачів які нічого не підозрюють,

коли вони відвідують веб-сайт, надаючи зловмиснику можливість викрасти їхні особисті дані або здійснити злі дії від їх імені.

XSS-атаки можна класифікувати на три типи:

Reflected XSS: у цьому типі атаки зловмисник впроваджує сценарій, який повертається користувачеві у відповідь веб-програми. Браузер користувача виконує сценарій, який можна використовувати для викрадення конфіденційних даних.

Stored XSS: у цьому типі атаки зловмисник впроваджує сценарій, який зберігається в базі даних або на сервері веб-програми. Щоразу, коли користувач відвідує заражену веб-сторінку, виконується сценарій, який можна який використовується для зловмисних дій.

XSS на основі DOM: у цьому типі атаки шкідливий код виконується у браузері жертви, а не на веб-сервері. Зловмисник впроваджує сценарій, який змінює об’єктну модель документа (DOM) веб-сторінки, яка може бути використана для викрадення конфіденційних даних або виконання інших зловмисних дій.

Зловмисники можуть вставити зловмисний код на веб-сайт за допомогою різних методів, включаючи поля введення, рядки запиту, файли cookie або заголовки HTTP. Після введення шкідливого коду його можна запускати щоразу, коли завантажується заражена сторінка, навіть ті користувачі, які не були цілями атаки.

XSS-атаки можуть використовуватися для викрадення конфіденційних даних (таких як номери кредитних карток або паролів для входу), зміни веб-сайтів, розповсюдження зловмисного програмного забезпечення або викрадення сеансів користувачів, серед іншого. Крім того, зловмисники можуть використовувати недоліки XSS як стартовий майданчик для більш складних атак, таких як впровадження SQL, віддалене виконання коду або підвищення привілеїв [18].

Оскільки XSS-атаки покладаються на людську діяльність, виявити та запобігти їм може бути складно. Наприклад, хакер може створити переконливий фішинговий електронний лист, щоб обманом змусити жертву відвідати ворожий веб-сайт, який імітує надійний. Користувачі повинні стежити за дивними посиланнями, спливаючими вікнами та попередженнями, а також використовувати розширення веб-

переглядача або інструменти безпеки, які блокують або сповіщають їх про потенційні атаки XSS, щоб не стати жертвою цих атак.

У 2020 році обсяг XSS-атак зріс на 10% порівняно з аналогічним періодом минулого року, а 39% усіх атак на веб-додатки становили XSS-атаки, згідно зі звітом Akamai про стан Інтернету/безпеку за 2021 рік. Тому веб-розробники, експерти з безпеки та користувачі повинні знати про XSS-атаки та вживати необхідних запобіжних заходів для їх запобігання.

Приклад сценарію атак:

Скажімо, «example.com» – це веб-сайт, який дозволяє людям залишати коментарі до публікацій блогу.

Оскільки розділ коментарів веб-сайту відкритий для атак XSS, зловмисник вирішує замаскувати шкідливий сценарій під законний коментар. Наприклад, хтось може написати як коментар:

“Привіт, чудова публікація! Перегляньте цей чудовий веб-сайт: <script>тут розміщено шкідливий код</script>.”

Розділ сценарію «тут розміщено шкідливий код» може містити будь-що: від простого попередження до більш складного нападу, який викрадає дані користувача або надсилає їх на фішинговий веб-сайт.

Інший користувач бачить коментар зловмисника, коли відвідує статтю блогу та читає коментарі, після чого сценарій запускається в його браузері.

Наприклад, зловмисний код може створювати спливаюче вікно, що імітує справжнє повідомлення з веб-сайту, і запитувати інформацію для входу користувача. Зловмисник може отримати доступ до облікового запису жертви та потенційно викрасти важливу інформацію, якщо користувач піддасться на хитрість і введе свої облікові дані для входу.

1.5.4 Broken authentication

Коли зловмиснику вдається використати слабкі місця в процесах автентифікації програми, паролі, керування сеансом або засоби контролю доступу, це називається «атакою зі зламаною автентифікацією», що є свого роду вразливістю безпеки. Ці

недоліки можуть надати зловмиснику доступ до конфіденційних даних без авторизації або дати йому можливість діяти від імені авторизованих користувачів [19].

Вгадування паролю – це найтиповіший тип атаки зі зламаною автентифікацією, під час якої зловмисник пробує численні комбінації імені користувача та пароля, поки не знайде той, який працює. Для виконання цього завдання вручну або автоматично можна використовувати відому інформацію цільового користувача, списки часто використовуваних паролів або автоматизовані інструменти.

Викрадення сеансу, яке відбувається, коли зловмисник викрадає токен сеансу або файл cookie користувача та використовує його для припущення особи користувача, є ще одним типовим видом атаки з порушенням автентифікації. Це може статися, коли токен сеансу неправильно перевірено програмою або коли токен доставлено через незахищений канал [20].

Інші типи атак зі зламаною автентифікацією включають введення облікових даних і атаки грубої сили, під час яких зловмисник пробує десятки тисяч або мільйони потенційних паролів за допомогою автоматизованих інструментів і списків викрадених імен користувачів і паролів з інших веб-сайтів.

Приклад сценарію атак:

Розглянемо систему онлайн-банкінгу, де користувачі повинні входити, використовуючи ім’я користувача та пароль для доступу до своїх рахунків. Блокуючи користувачів після попередньо визначеної кількості невдалих спроб входу, метод призначений для запобігання нападам грубої сили. Проте процедура автентифікації системи має недолік, який дає змогу зловмиснику обійти ці заходи безпеки.

Зловмисник спочатку використовує список часто використовуваних паролів, щоб спробувати отримати доступ до системи онлайн-банкінгу. Система блокує їх після численних невдалих спроб. Однак зловмисник знає, що файли cookie використовуються системою для збереження сеансів користувачів після успішної автентифікації.

Файл cookie системи, який надсилається під час дійсного сеансу входу користувача, потім перехоплюється зловмисником за допомогою інструменту. Щоб

почати новий сеанс на власному пристрої, вони копіюють файл cookie. Система приймає зловмисника як законного користувача та надає йому доступ до облікового запису, оскільки файл cookie містить облікові дані користувача.

Потрапивши в обліковий запис, зловмисник може отримати доступ до даних облікового запису користувача, відправити гроші та здійснити інші мерзенні дії. Поки користувач не вийде з системи або не закінчиться термін дії файлу cookie, він може отримати доступ до облікового запису за допомогою вкраденого файлу cookie.

1.5.5 Витік конфіденційних даних

Деякі вразливості можна класифікувати як порушення конфіденційних даних, і всі вони мають спільне ненавмисне розкриття особистих даних, які повинні бути зашифровані. Перш за все, конфіденційні дані включають інформацію, як-от ідентифікаційні податкові номери та облікові дані, а також номери банківських карток, які можуть бути використані або змінені для негідних цілей. Користувач вводить будь-яку інформацію у веб-програму, знаючи, що сервер захистить цю конфіденційну інформацію [21]. Однак незахищені системи часто зберігають конфіденційні дані в базах даних, чутливих до SQL-ін’єкцій та інших форм атак, і взагалі не шифрують їх.

Навіть якщо використовується шифрування, його не завжди може бути достатньо, оскільки численні веб-програми продовжують використовувати прості хеші або слабкі криптографічні алгоритми для захисту конфіденційних даних.

Хоча хакерам зазвичай надзвичайно важко скористатися цими недоліками, вплив є дуже серйозним, тому вкрай важливо їх зрозуміти та зробити правильний вибір архітектури програми. Оскільки більшість, якщо не всі, компанії працюють за допомогою онлайн-додатків, їхні дані постійно знаходяться під загрозою як внутрішніх, так і зовнішніх загроз. Цей ризик може призвести до значних витрат для бізнесу, таких як витрати на відновлення безпеки, витрати на сповіщення жертв і надання їм підтримки, а також витрати на регулятивних штрафів [22].

Приклад сценарію атаки:

1. Отримання прямого доступу до інших записів у базі даних, якими вони не володіють, використовуючи значення ідентифікатора(ID).

Приклад URL в уразливому додатку:

www.example.com/profile/1234

Запис бази даних для профілю ідентифікується в цій URL-адресі за ідентифікатором 1234. Зловмисник може легко змінити його на інше значення та отримати доступ до обмежених профілів інших користувачів, оскільки він відображається на URL-адресу та є передбачуваним.

Інший приклад використання URL-адреси для отримання інформації з файлової системи:

www.example.com/files?name=examplefile.pdf

Фактична назва файлу, який потрібно видобути, визначається аргументом name в цій URL-адресі. Зловмисники можуть змінити це очікуване налаштування, щоб отримати доступ до додаткових файлів, доступу до яких у них немає.

2. Виконання операцій в системі завдяки значенням параметра.

www.example.com/changepassword?user=name

У цьому випадку програмі вказується, який користувач має оновити пароль за значенням аргументу користувача. Цей етап часто буде компонентом багатоетапного процесу. Програма спочатку отримає запит на зміну пароля конкретного користувача, а на наступному етапі користувач введе новий пароль (без запиту старого).

Пряме посилання на об'єкт користувача, для якого буде виконана операція зміни пароля, здійснюється за допомогою аргументу user. Зловмисник може спробувати ввести ім’я користувача, відмінне від того, яке зараз зареєстровано, і може змінити пароль іншого користувача. Що поведе за собою негативні наслідки.

3. Вилучення інформації з файлової системи використовуючи значення параметра.

www.example.com/showImage?img=img1234.jpg

У цьому випадку програмі повідомляється файл, який користувач хоче отримати, за значенням параметра img (може бути будь-який файл, не тільки

картинка). Зловмисник може отримати елементи, які належать іншим користувачам, надавши назву або ідентифікатор іншого файлу (наприклад, img= image4321.jpg).

4. Доступ до фунціональності програми використувуючи значення параметра.

www.example.com/accessPage?menuitem=10

У цьому випадку програма отримує інформацію про те, до якого пункту меню (і, отже, до якої функції програми) користувач намагається отримати доступ, за допомогою значення аргументу menuitem. Припустимо, користувач обмежений і може отримати доступ лише до пунктів меню 1, 2 і 3 завдяки доступним йому посиланням. Значення параметра пункту меню можна змінити, щоб обійти автентифікацію та отримати доступ до додаткових функцій програмного забезпечення. Зловмисник знаходить місце, де пункт меню визначає функціональність програми за допомогою посилання, відображає значення пунктів меню, до яких він отримує доступ, а потім пробує виконати інші пункти меню.

1.6 Основні методи аналізу веб-експлойтів

Веб-експлойти стосуються будь-якої зловмисної діяльності, яка використовує вразливі місця у веб-додатках або веб-серверах. Основні методи аналізу веб-експлойтів можна загалом розділити на два типи: статичний аналіз і динамічний аналіз [23].

Статичний аналіз: цей метод передбачає аналіз вихідного коду веб-програми або веб-сервера для виявлення вразливостей. У статичному аналізі використовуються такі основні методи:

-перевірка коду вручну: це передбачає ручну перевірку вихідного коду на наявність уразливостей у безпеці. Цей метод трудомісткий і вимагає високого рівня кваліфікації;

-автоматизовані інструменти аналізу коду: автоматизовані інструменти можуть допомогти виявити вразливі місця безпеки в коді. Ці інструменти використовують

різні методи, такі як зіставлення шаблонів, аналіз потоку даних і аналіз потоку керування для виявлення вразливостей;

-Fuzz-тестування: Fuzz-тестування передбачає надсилання випадкових вхідних даних до веб-додатку або сервера для виявлення будь-якої неочікуваної поведінки чи вразливостей.

Динамічний аналіз: цей метод передбачає аналіз поведінки веб-програми або сервера під час виконання. У динамічному аналізі використовуються такі основні методи:

-тестування на проникнення: Тестування на проникнення передбачає імітацію атаки на веб-програму або сервер для виявлення вразливостей. Цей метод можна виконати вручну або за допомогою автоматизованих засобів.

-сканери веб-додатків: ці інструменти автоматизують процес тестування вразливостей шляхом сканування веб-додатків на відомі вразливості та типові моделі атак.

-аналіз трафіку: включає в себе аналіз мережевого трафіку між клієнтом і сервером для виявлення будь-яких вразливостей або неочікуваної поведінки.

Загалом, комбінація методів статичного та динамічного аналізу зазвичай використовується для ретельного аналізу веб-експлойтів і виявлення будь-яких вразливостей безпеки [24].

Висновок до першого розділу

В першому розділі розглядались основні ризики для безпеки веб-додатків. Вони здебільшого залежать від того, наскільки добре реалізовано захист від них як на стороні клієнта, так і на стороні сервера. Клієнтська сторона веб-додатків покладається на використання такого інструменту, як JavaScript, і вразливості, які виникають у ньому, часто походять від неправильної перевірки вхідних та вихідних даних. З боку клієнта, деяким із них можна повністю або частково запобігти.

Технології та компоненти, які використовуються під час розробки онлайн-програм, а також будь-які потенційні недоліки в цих компонентах впливають на те, наскільки успішно веб-програми захищені від шкідливих атак. Існують різні категорії

вразливостей, і кожна атака вразливості має свої особливості. Однак помилки в дизайні, реалізації та використанні компонентів веб-додатків є причиною вразливості, тому необхідно шукати вразливості та реагувати на повідомлення про їх виявлення.

Існують численні програми заохочення за виявлення вразливостей, такі як Bug Bounty, а також організовані групи екстреного реагування в кібербезпеці як в Україні, так і в інших країнах світу. Окремі міжнародні стандарти регулюють процедуру розкриття вразливостей, а спеціалізовані бази даних уразливостей містять статистичні дані про виявлені вразливості та їх характеристики.

Знайти вразливості та виправити їх можна за допомогою різноманітних технологій, але ефективність їх використання залежить від алгоритму дій. Заходи захисту вимагають спеціальних знань, дорогого обладнання або багато часу для застосування. Таким чином, необхідно створити техніку захисту, яка використовує найменшу кількість ресурсів і при цьому захищає веб-додатки належним чином.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ІСНУЮЧИХ ІНСТРУМЕНТІВ ДЛЯ АНАЛІЗУ ВЕБ-ЕКСПЛОЙТІВ ТА ЗАХИСТ ВІД АТАК НА ВЕБ-ДОДАТКИ

2.1 Інструменти для аналізу веб-експлойтів

JavaScript є важливою мовою програмування для Інтернету та широко використовується веб-розробниками. Однак він також часто використовується зловмисниками для виконання шкідливого коду в цільовій системі. В останні роки спостерігається збільшення кількості веб-експлойтів на основі JavaScript. У результаті було розроблено багато інструментів для аналізу цих веб-експлойтів.

Експлойти на основі JavaScript - це шкідливі сценарії, які використовують JavaScript для використання вразливостей у веб-додатках. Ці експлойти призначені для викрадення конфіденційної інформації або отримання несанкціонованого доступу до системи. Декілька прикладів експлойтів на основі JavaScript включають міжсайтовий сценарій (XSS), миттєві завантаження та зловмисне перенаправлення.

Існує кілька інструментів, доступних для аналізу веб-експлойтів на основі JavaScript. Ці інструменти можна розділити на дві основні групи: інструменти статичного аналізу та інструменти динамічного аналізу.

Інструменти статичного аналізу:

Інструменти статичного аналізу аналізують вихідний код веб-програми, щоб виявити потенційні вразливості. Ці інструменти можуть виявити такі проблеми, як впровадження коду, незахищена конфігурація та неправильне керування доступом. Деякі приклади інструментів статичного аналізу включають:

ESLint - це популярний інструмент для розпізнавання JavaScript, який можна використовувати для виявлення потенційних проблем безпеки у вихідному коді веб-програми. Він може виявляти такі проблеми, як впровадження коду, міжсайтовий сценарій та інші вразливості, які можуть виникнути через небезпечні практики кодування. ESLint має широкі можливості налаштування, що дозволяє розробникам адаптувати інструмент до своїх конкретних потреб і вподобань.

Retire.js - це інструмент, який можна використовувати для виявлення вразливих бібліотек JavaScript, які використовуються у веб-програмі. Він сканує вихідний код веб-програми, щоб виявити застарілі або вразливі бібліотеки, якими можуть скористатися зловмисники. Retire.js також надає рекомендації щодо оновлення бібліотек до останніх, безпечніших версій.

SonarQube - це інструмент аналізу якості коду, який може виявити вразливі місця у вихідному коді веб-програми. Він забезпечує автоматизований перегляд коду та допомагає виявляти такі проблеми, як впровадження SQL, міжсайтовий сценарій та інші поширені вразливості. SonarQube можна інтегрувати в процес розробки, щоб забезпечити безперервний зворотний зв’язок і забезпечити безпеку коду перед його розгортанням.

Інструменти динамічного аналізу:

Інструменти динамічного аналізу аналізують поведінку веб-програми під час виконання, щоб виявити потенційні проблеми безпеки. Ці інструменти можуть виявляти такі проблеми, як міжсайтовий сценарій, впровадження SQL і зловмисне перенаправлення. Деякі приклади інструментів динамічного аналізу включають:

Burp Suite - це популярний інструмент тестування безпеки веб-додатків, який містить проксі-сервер для перехоплення та модифікації HTTP-запитів і відповідей. Він також містить сканер, який можна використовувати для виявлення вразливостей, таких як міжсайтовий сценарій, впровадження SQL та інших поширених уразливостей веб-додатків. Burp Suite можна використовувати для перевірки безпеки веб-додатків під час розробки або після розгортання.

ZAP (Zed Attack Proxy) - ще один популярний інструмент тестування безпеки веб-додатків, який містить функції для виявлення вразливостей у веб-додатках. Він містить проксі-сервер-перехоплювач, який дозволяє користувачам переглядати та змінювати HTTP-запити та відповіді, а також сканер, який може ідентифікувати такі вразливості, як міжсайтовий сценарій та впровадження SQL. ZAP можна використовувати для перевірки безпеки веб-додатків під час розробки або після розгортання.

OWASP Amass - це інструмент із відкритим вихідним кодом, який можна використовувати для виявлення веб-програм і виявлення в них потенційних уразливостей. Він містить функції для сканування веб-сайтів і ідентифікації субдоменів, а також сканер, який може виявляти вразливості, такі як впровадження SQL і міжсайтовий сценарій. OWASP Amass можна використовувати для перевірки безпеки веб-додатків під час розробки або після розгортання.

Загалом ці інструменти розроблені, щоб допомогти розробникам і фахівцям із безпеки виявляти та пом’якшувати вразливі місця у веб-додатках. Однак важливо зазначити, що ці інструменти слід використовувати разом з іншими заходами безпеки, такими як безпечне кодування та регулярні оцінки безпеки, щоб забезпечити безпеку веб-додатків.

2.2 Порівняльний аналіз інструментів для аналізу веб-експлойтів

Порівняння різних інструментів разом із їхніми відповідними плюсами та мінусами:

ESLint:

Плюси:

-Можливість налаштування за допомогою різноманітних плагінів і правил

-Можна інтегрувати з популярними текстовими редакторами та інструментами для створення

-Може ідентифікувати потенційні вразливості безпеки та помилки кодування

Мінуси:

-Конфігурація та налаштування може зайняти багато часу

-Може давати хибнопозитивні або хибнонегативні результати

Retire.js:

Плюси:

-Може ідентифікувати застарілі та вразливі бібліотеки JavaScript

-Можна інтегрувати з інструментами створення для автоматизації процесу сканування

-Допомагає підтримувати програми в актуальному стані за допомогою останніх виправлень безпеки

Мінуси:

-Може виявити не всі вразливості

-Може давати хибнопозитивні або хибнонегативні результати

-Може знадобитися ручне оновлення або заміна бібліотек

SonarQube:

Плюси:

-Може визначати вразливі місця безпеки, запахи коду та інші проблеми

-Можливість налаштування за допомогою різноманітних плагінів і правил

-Можна інтегрувати з популярними інструментами збірки

Мінуси:

-Конфігурація та налаштування може зайняти багато часу

-Може давати хибнопозитивні або хибнонегативні результати

Burp Suite:

Плюси:

-Може визначити поширені вразливості веб-додатків

-Включає різноманітні функції для виявлення вразливостей

-Використовується дослідниками безпеки та тестувальниками проникнення

Мінуси:

-Може бути складним і трудомістким у використанні

-Може давати хибнопозитивні або хибнонегативні результати

-Може бути дорогим для комерційного використання

ZAP:

Плюси:

-Може визначити поширені вразливості веб-додатків

-Включає різноманітні функції для виявлення вразливостей

-Безкоштовний і з відкритим кодом

Мінуси:

-Може бути складним і трудомістким у використанні

-Може давати хибнопозитивні або хибнонегативні результати

Node.js:

Плюси:

-Можна використовувати для створення безпечних веб-додатків із використанням найкращих практик безпечного кодування

-Включає вбудовані модулі для обробки HTTP-запитів і обробки даних

-Безкоштовний і з відкритим кодом

Мінуси:

-Потрібні знання JavaScript і веб-розробки

-Створення безпечних програм може зайняти багато часу

Firebug:

Плюси:

-Інструмент із відкритим кодом для аналізу та налагодження веб-додатків

-Включає різноманітні функції для аналізу веб-додатків

-Безкоштовний і простий у використанні

Мінуси:

-Більше не розробляється та не підтримується

-Обмежується аналізом веб-додатків у веб-браузері Mozilla Firefox

JSHint:

Плюси:

-Може визначити потенційні проблеми безпеки в коді JavaScript

-Безкоштовний і з відкритим кодом

-Простий у використанні та інтеграції з існуючими робочими процесами

Мінуси:

-Може давати хибнопозитивні або хибнонегативні результати

-Обмежується статичним аналізом коду JavaScript.

2.3 Порівняльний аналіз існуючих методологій для ІБ

На сьогоднішній день найбільш розповсюдженими методологіями які можна використовувати для аналізу веб-додатків на уразливості є:

- The Open Source Security Testing Methodology Manual (OSSTMM);
- The National Institute of Standards and Technology (NIST) Special Publication 800-115;
- OWASP Testing Guide;
- Penetration Testing Execution Standard (PTES);
- Information Systems Security Assessment Framework (ISSAF);
- BSI – Study A Penetration Testing Model.

1. Аналіз методології The Open Source Security Testing Methodology Manual (OSSTMM).

Добре відомою та загальновизнаною структурою для виконання тестування безпеки та оцінки інформаційних систем є OSSTMM. OSSTMM, створений Пітом Герцогом і підтримується Інститутом безпеки та відкритих методологій (ISECOM), пропонує ретельний і організований метод виявлення вразливостей і оцінки загального стану безпеки системи.

OSSTMM призначений як ресурс з відкритим вихідним кодом, що означає, що будь-хто може використовувати та змінювати його без обмежень. Завдяки відкритому характеру, який сприяє співпраці та постійному розвитку, його можна налаштувати відповідно до вимог окремих експертів із безпеки та організацій.

Науковий підхід до тестування безпеки підкреслюється технікою, описаною в OSSTMM, яка ґрунтується на неупереджених вимірюваннях і статистичному аналізі. Він зосереджений на оцінці безпеки з точки зору операційної, людської та фізичної безпеки. OSSTMM пропонує комплексну оцінку загальної безпеки системи, розглядаючи ці компоненти в цілому.

Сама техніка, інструменти, політики та вимірювання є одними з основних частин OSSTMM. Методологія описує покрокову процедуру проведення тестування безпеки, що включає такі завдання, як збір даних, виявлення вразливостей, оцінка ризиків і звітування. Він пропонує поради щодо різноманітних методів тестування, таких як оцінка соціальної інженерії, перевірка безпеки та тестування на проникнення.

OSSTMM надає різноманітні інструменти та інструкції для допомоги в техніці. Ці технології полегшують процес тестування, автоматизуючи деякі операції та спрощуючи збір і аналіз даних. Рекомендації надають тестувальникам більше деталей і передових практик для тестування безпеки, допомагаючи їм дотримуватися галузевих норм і гарантуючи глибину й точність їхніх оцінок.

Зосередженість OSSTMM на метриках є однією з його видатних характеристик. Посібник пропонує набір уніфікованих заходів, які дозволяють підприємствам оцінювати та оцінювати рівень безпеки. Ці вимірювання дозволяють проводити надійний порівняльний аналіз, аналіз тенденцій і порівняння різних заходів безпеки. Організації можуть використовувати показники, щоб відстежувати свій розвиток з часом і з упевненістю приймати рішення про покращення безпеки.

Природа OSSTMM з відкритим вихідним кодом сприяла його широкому прийняттю та визнанню в спільноті безпеки. Він став корисним інструментом для наукових установ, компаній і експертів із безпеки, яким потрібен ретельний і організований метод тестування безпеки. Процес спільної розробки гарантує, що він тримає руку на пульсі нових загроз і найкращих практик, надаючи йому надійну та надійну техніку для оцінки безпеки.

Переваги методики:

- Комплексний підхід: OSSTMM пропонує методичний і всеохоплюючий підхід до тестування безпеки, який стосується операційної, людської та фізичної безпеки. Це гарантує, що під час оцінювання враховуються всі потенційні вразливості та слабкі сторони.
- Науковий підхід: OSSTMM робить сильний акцент на неупередженому науковому методі тестування безпеки. Цей метод більш надійний і послідовний, оскільки він базується на вимірюваннях, показниках і статистичному аналізі.
- Співпраця та відкритий вихідний код: OSSTMM сприяє співпраці та внеску з боку спільноти безпеки, оскільки це техніка з відкритим кодом. Це призводить до постійного вдосконалення, обміну знаннями та засвоєння багатьох точок зору та спеціалізацій.

- Стандартизовані показники: OSSTMM пропонує стандартні показники, які дозволяють підприємствам оцінювати рівень безпеки. Це дає змогу оцінювати покращення безпеки, аналізувати тенденції та приймати мудрі рішення.

- Гнучкість і адаптивність: оскільки OSSTMM відкритий, підприємства можуть налаштувати його відповідно до власних вимог. Архітектуру та інструменти можна змінювати та налаштовувати відповідно до їхніх конкретних налаштувань та потреб.

Недоліки методики:

- Складність: OSSTMM може бути складним для розгортання та потребує певного рівня знань. Організаціям з обмеженими ресурсами або групам безпеки з меншим досвідом може бути важко прийняти та запровадити методологію.

- Крива навчання: для людей або організацій, які тільки починають тестувати безпеку, OSSTMM може мати високу криву навчання через його комплексний характер. Може знадобитися деякий час і зусилля, щоб повністю зрозуміти методологію та успішно її використовувати.

- Відсутність офіційної сертифікації: OSSTMM не має офіційної програми сертифікації, пов’язаної з ним, на відміну від кількох інших підходів до безпеки чи сертифікацій. Для компаній, які цінують авторитетні сертифікати як доказ своїх процедур безпеки, це може бути недоліком.

- Обмежена підтримка постачальників: порівняно з власними фреймворками OSSTMM може мати меншу підтримку постачальників, оскільки це техніка з відкритим кодом. Компанії, які значною мірою покладаються на певних постачальників або продукти безпеки, можуть мати проблеми з ефективною інтеграцією цих рішень із OSSTMM.

- Потенційно застаріла інформація: OSSTMM чутливий до наявності потенційно застарілої інформації, як і будь-який інший ресурс. Користувачі техніки повинні бути в курсі останніх оновлень, оскільки загрози безпеці та технології швидко змінюються.

2. «Технічний посібник з тестування та оцінки інформаційної безпеки», документ Національного інституту стандартів і технологій (NIST), пропонує

інструкції та пропозиції щодо виконання завдань тестування та оцінки інформаційної безпеки. Для компаній і окремих осіб, зацікавлених в оцінці та тестуванні безпеки інформаційних систем, він надає всеосяжний ресурс.

Випуск NIST SP 800-115 від 2008 року зазнав оновлень, щоб відповідати новим загрозам і розвиватися технологіям. У методології наведено різноманітні ідеї, методи та найкращі практики для тестування та оцінки безпеки. Він спрямований на підтримку підприємств у виявленні ризиків безпеки, недоліків і вразливостей у їхніх інформаційних системах. Тестування та оцінка інформаційної безпеки дуже докладно розглядаються в цьому документі.

NIST SP 800-115 розглядає ряд важливих питань, зокрема:

У статті розглядаються методології тестування та оцінювання безпеки інформаційних систем, включаючи тестування на проникнення, сканування вразливостей та оцінки контролю безпеки. Він пропонує пропозиції щодо вибору відповідних методів оцінювання відповідно до вимог і цілей організації.

NIST SP 800-115 наголошує на важливості ретельного планування та підготовки перед проведенням оцінки безпеки. Він пояснює процедури, задіяні у створенні стратегії оцінювання, такі як визначення обсягу, розподіл ресурсів і встановлення цілей оцінювання.

Видання пропонує вичерпні інструкції щодо проведення оцінки безпеки, включаючи використання різних інструментів і підходів. Він вирішує такі проблеми, як відображення мережі, використання вразливостей і звітування про результати.

Звітування та подальші дії: NIST SP 800-115 підкреслює необхідність точного запису результатів оцінювання. Він пропонує поради щодо того, як писати ретельні звіти, які передають зацікавленим сторонам висновки, пропозиції та коригувальні заходи. Це також підкреслює важливість моніторингу виявлених вразливостей і підтвердження успішності коригувальних дій.

Організації та експерти з безпеки загалом визнають і використовують NIST SP 800-115 як корисний інструмент для проведення ефективного тестування та оцінки безпеки інформації. Виявляючи та усуваючи системні недоліки та вразливі місця, це допомагає компаніям підвищити рівень безпеки.

Важливо пам’ятати, що хоча NIST SP 800-115 пропонує багато порад, це не універсальна відповідь. Впроваджуючи пропозиції, викладені в документі, організації повинні враховувати свої особливі вимоги, конкретну технологію, яку вони використовують, і будь-яке галузеве законодавство чи стандарти.

Переваги:

- Комплексне керівництво: NIST SP 800-115 надає комплексну структуру для аналізу та оцінки інформаційної безпеки. Він охоплює різні методології, техніки та найкращі практики, надаючи організаціям міцну основу для проведення ретельного оцінювання.
- Визнаний стандарт: NIST є високоповажною організацією, і її публікації, включаючи SP 800-115, широко визнані галузевими стандартами. Дотримання вказівок NIST може підвищити довіру до організації та продемонструвати відданість найкращим практикам інформаційної безпеки.
- Адаптація до різних середовищ: публікація пропонує гнучкість і може бути застосована до різних середовищ, включаючи різні типи інформаційних систем і технологій. Він містить вказівки щодо вибору відповідних методів оцінювання на основі конкретних потреб і цілей організації.
- Усвідомлює нові небезпеки: NIST SP 800-115 регулярно оновлюється з урахуванням нових загроз і технологій, що розвиваються. Дотримуючись рекомендацій, організації можуть стежити за найновішими проблемами безпеки та за потреби змінювати свої процедури тестування та оцінювання.

Недоліки:

- Складність: для людей або компаній без серйозного досвіду в інформаційній безпеці публікація може бути надто технічною та складною. Для ефективного застосування деяких порцій можуть знадобитися знання і досвід.
- Відсутність кастомізації: NIST SP 800-115 пропонує надійну структуру, але вона може не охоплювати всіх конкретних вимог організації чи галузевих правил. Організації повинні брати до уваги свої унікальні обставини та змінювати правила за необхідності.

- Середовище, яке швидко розвивається: інформаційна безпека є динамічною сферою, у якій постійно з’являються нові загрози та технології. Незважаючи на те, що NIST SP 800-115 оновлюється, може виникнути затримка у вирішенні останніх тенденцій і вразливостей, що вимагає від компаній доповнити публікацію додатковими ресурсами.

3. Open Web Application Security Project (OWASP) — некомерційна організація, яка займається підвищенням безпеки веб-додатків. Для розробників, експертів із безпеки та компаній, щоб зрозуміти та зменшити ризики, пов’язані з уразливістю веб-додатків, OWASP пропонує безліч знань та інструментів. Посібник з тестування OWASP є одним із найвідоміших інструментів, які пропонує OWASP.

Посібник OWASP — це вичерпний посібник, який описує найкращі практики та процедури тестування для виявлення недоліків безпеки в онлайн-додатках. Він діє як корисний посібник для тестувальників безпеки, програмістів і всіх, хто залучений до створення або підтримки веб-додатків. Спільнота OWASP постійно оновлює посібник, щоб відобразити мінливий ландшафт загроз і новітні методи атак.

Посібник OWASP охоплює широкий спектр тем, зокрема:

Вступ: у цьому розділі наведено огляд посібника, пояснено важливість тестування безпеки веб-додатків, а також описано методології та методи тестування.

Збір інформації: ця фаза передбачає збір інформації про цільову веб-програму, таку як визначення її компонентів, використовуваних технологій і потенційних точок входу для зловмисників.

Тестування конфігурації та керування розгортанням: цей розділ присвячено тестуванню безпеки налаштувань конфігурації програми та механізмів розгортання, гарантуючи, що вони дотримуються правил безпеки.

Тестування керування ідентифікацією: воно охоплює перевірку механізмів автентифікації та авторизації веб-додатків, гарантуючи їх надійність і стійкість до поширених атак, таких як підкидання облікових даних, підбір і викрадення сеансу.

Тестування перевірки вхідних даних: ця фаза передбачає перевірку того, як програма обробляє введені користувачем дані, включаючи перевірку даних,

фільтрацію та очищення, щоб запобігти поширеним уразливостям, таким як впровадження SQL, міжсайтовий сценарій (XSS) і впровадження команд.

Тестування на наявність неправильних конфігурацій безпеки: воно зосереджено на виявленні та тестуванні на наявність неправильних конфігурацій у веб-серверах, базах даних, фреймворках та інших компонентах, які можуть призвести до вразливості безпеки.

Тестування на вразливості: у цьому розділі розглядаються різні типи вразливостей, такі як помилки ін’єкцій, порушення автентифікації та керування сеансами, атаки на зовнішні об’єкти XML (XXE), обхід системи безпеки тощо.

Тестування веб-сервісів: містить вказівки щодо тестування веб-сервісів, включаючи API, SOAP і RESTful-сервіси, щоб переконатися, що вони безпечні та не схильні до поширених уразливостей.

Інструменти тестування безпеки: у цьому розділі представлено низку інструментів тестування безпеки та пояснюється, як їх ефективно використовувати в процесі тестування.

Звітування: підкреслюється важливість документування та звітування про вразливості, виявлені під час процесу тестування, разом із рекомендованими кроками для усунення.

Важливо пам’ятати, що OWASP – це проект, керований спільнотою, і що знання та досвід усієї спільноти OWASP використовуються для постійного оновлення та вдосконалення його вмісту. Він пропонує корисний підхід і структуру для тестування веб-додатків, допомагаючи підприємствам у зміцненні їх безпеки та створенні більш довговічних і безпечних додатків.

Переваги:

- Охоплення всіх аспектів: OWASP забезпечує ретельний метод оцінки безпеки веб-додатків, включаючи охоплення збору інформації, автентифікації, перевірки введених даних, керування конфігурацією тощо. Це гарантує, що тестери дотримуються визначеної методичної техніки, знижуючи ймовірність того, що вони можуть пропустити важливі вразливості.

- Найкращі практики та вказівки: ґрунтуючись на фактичному досвіді та галузевих нормах, посібник пропонує найкращі практики та вказівки. Він надає корисні поради щодо захисту онлайн-додатків і допомагає тестувальникам і розробникам зрозуміти й успішно усунути поширені вразливості.

- Керований спільнотою та оновлений: OWASP — це проект, який базується на сукупних знаннях і внесках фахівців із безпеки з усього світу. Його часто оновлюють, щоб включити нові стратегії атак, діри в безпеці та процедури тестування, щоб підтримувати його актуальним і корисним.

- Навчальний ресурс: Посібник є корисним освітнім ресурсом для людей і компаній, зацікавлених дізнатися більше про безпеку веб-додатків. Це допомагає експертам із безпеки дізнатися про різні шляхи атак і типи вразливостей, дозволяючи їм створювати ефективні заходи пом’якшення.

- Підтримка відповідності та нормативних актів: використання Посібника з тестування OWASP може допомогти компаніям дотримуватися законодавчих і нормативних вимог щодо безпеки онлайн-додатків. Це допомагає підтвердити, що програми були захищені, а конфіденційні дані були ретельно захищені.

Недоліки:

- Складність навчання: Посібник з тестування OWASP охоплює широкий спектр тем і методологій, які можуть налякати новачків або тих, хто не має досвіду тестування безпеки веб-додатків. Щоб правильно зрозуміти і використати посібник, вам може знадобитися багато часу та зусиль.

- Контекстно-специфічні міркування: хоча посібник пропонує детальну структуру, він може не охоплювати всі конкретні потреби та складності будь-якої програми чи стеку технологій. Виконуючи рекомендації з керівництва, тестувальники та розробники повинні враховувати унікальний контекст і вимоги своїх додатків.

- Обмеження інструментів: хоча посібник пропонує ряд інструментів тестування безпеки, він не вдається в подробиці щодо обмежень або недоліків цих інструментів. Щоб забезпечити ретельне тестування, випробувачі повинні знати про переваги та недоліки інструментів, які вони використовують.

- Загрози, що швидко змінюються: Безпека веб-додатків — це сфера, яка постійно змінюється, постійно з’являються нові методи атак і недоліки. Хоча Посібник з тестування OWASP часто оновлюється, він не завжди може містити найновіші загрози. Окрім цього документа, тестувальники та розробники повинні бути в курсі нових досягнень і найкращих практик.

- Обмеження ресурсів: для повного виконання Посібника з тестування OWASP можуть знадобитися спеціальні ресурси, наприклад досвідчені тестувальники, час та інструменти. Меншим підприємствам або підприємствам з меншими ресурсами може бути важко повністю реалізувати рекомендації посібника.

4. Стандарт виконання тестування на проникнення (PTES) — це всеохоплююча основа та рекомендації щодо проведення ефективного та систематичного тестування на проникнення.

Щоб забезпечити повну оцінку стану безпеки систем і мереж організації, він пропонує організований спосіб планування, проведення та документування заходів тестування на проникнення.

PTES був створений командою досвідчених тестувальників проникнення та експертів із безпеки з метою стандартизації дисципліни та підвищення її надійності та ефективності. Він допомагає тестувальникам проникнення, компаніям і клієнтам ефективніше спілкуватися та розуміти процес тестування, надаючи узгоджену мову та точку відліку.

Структура PTES розділена на сім етапів, які разом охоплюють повний життєвий цикл:

Взаємодія перед залученням: мета цього етапу полягає в тому, щоб створити чітку лінію зв’язку та порозуміння між організацією та командою тестування на проникнення. Він містить окреслення цілей тесту, правил ведення бою та інших технічних характеристик.

Збір розвідувальної інформації: на цьому етапі отримують подробиці щодо діяльності цільової компанії. Це передбачає збір загальнодоступних даних, проведення розвідки мережі та виявлення потенційних слабких чи вразливих місць.

Моделювання загроз: Тестери проникнення перевіряють зібрані дані, щоб визначити потенційні загрози та небезпеки, з якими може зіткнутися цільова компанія. Під час цієї фази визначаються цілі та розставляються пріоритети, наносяться шляхи атаки та оцінюється потенційний вплив компромісів.

Аналіз вразливостей: щоб знайти вразливості, тестери проникнення ретельно досліджують цільові мережі та системи. Використання як ручних, так і автоматизованих технологій сканування може використовуватися для пошуку вразливостей, якими можуть скористатися зловмисники.

Експлуатація: після того, як уразливості знайдено, ця фаза передбачає спроби скористатися ними, щоб скомпрометувати цільові системи або отримати до них неавторизований доступ. Тестувальники відтворюють реальні ситуації атаки, обмежуючи потенційну шкоду, використовуючи різні підходи, інструменти та методології.

Після успішної експлуатації тестувальники зберігають доступ до скомпрометованих систем і збільшити там свої привілеї. Вивчення масштабів компрометації, запис виконаних дій і пошук будь-яких інших уразливостей або потенційно відкритих даних є основними цілями цього етапу.

Звітування: останній крок передбачає узагальнення результатів і надання звіту організації. Резюме, технічні описи знайдених уразливостей, запропоновані дії для виправлення та інші відповідні дані включені до цього звіту, щоб допомогти організації зміцнити рівень безпеки.

PTES наголошує на важливості чіткої комунікації та співпраці з організацією, що тестується. Він сприяє етичній та відповідальній практиці тестування, гарантуючи, що тестувальники працюють у визначених межах і не викликають непотрібних збоїв або пошкоджень цільових систем.

Структура також визнає необхідність постійного вдосконалення та навчання. Він заохочує постійний професійний розвиток, обмін знаннями та дотримання етичних стандартів, сприяючи створенню спільноти кваліфікованих і відповідальних практиків.

Переваги методології:

- Визначена техніка: PTES пропонує структуру та визначену техніку для проведення тестування на проникнення. Це робить процедури тестування послідовними та повторюваними, полегшуючи порівняння результатів, відстеження прогресу та розповсюдження результатів між багатьма проектами та організаціями.

- Всебічне покриття: від раннього планування та збору розвідувальних даних до аналізу вразливостей, використання та звітування, архітектура PTES розглядає всі важливі аспекти тестування на проникнення. Він гарантує, що під час тестування не буде упущено жодної важливої ділянки, що веде до більш ретельної оцінки стану безпеки бізнесу.

- Чіткість у спілкуванні: PTES приділяє значну увагу тому, щоб команда тестування на проникнення та організація, що тестується, спілкувалися чітко та ефективно. Як наслідок, є менше помилкових уявлень і краща співпраця протягом усього тестування, оскільки є спільні знання про цілі, обсяг і обмеження.

- Етичне та відповідальне тестування: PTES заохочує етичне та відповідальне тестування, підкреслюючи важливість дотримання умов участі та моральних принципів. Це зменшує ймовірність ненавмисної шкоди або переривання систем і мереж, гарантуючи, що процес тестування залишається в межах встановлених параметрів.

- Постійний розвиток: PTES розуміє цінність безперервної освіти та професійного зростання в галузі тестування на проникнення. Він сприяє культурі безперервного вдосконалення та обміну знаннями в усьому бізнесі, заохочуючи практиків йти в ногу з найновішими техніками, інструментами та вразливими місцями.

Недоліки методології:

- Жорсткість: у деяких випадках, коли потрібна гнучкість і налаштування, стандартизований характер PTES може сприйматися як обмеження. Деякі підприємства можуть мати особливі потреби в тестуванні або певні системи, які не вписуються в установлені етапи та процедури PTES.

- Складність: PTES пропонує ретельний і організований підхід до тестування на проникнення, що може бути обтяжливим для практиків з меншим

досвідом або компаній з меншими ресурсами. Щоб належним чином зрозуміти та ефективно реалізувати структуру, може знадобитися отримати додаткове навчання та отримати навички.

- Затратність часу та ресурсів: дотримання структури PTES вимагає ретельного планування, проведення та записування кожного етапу процесу тестування. Це може зайняти багато часу та ресурсів, особливо для підприємств із обмеженими ресурсами чи кількістю персоналу.
- Обмежений обсяг: хоча PTES охоплює широкий спектр операцій тестування, він може не повністю задовольнити всі потреби безпеки організації. Деяким організаціям можуть знадобитися додаткові процедури тестування або спеціальні оцінки, які не входять до компетенції PTES.
- Відсутність гнучкості для Agile середовищ: традиційна методологія водоспаду була взята до уваги при розробці PTES. Жорстка структура PTES може не інтегруватися ефективно в налаштування гнучкої розробки, де швидкі ітерації та часті розгортання є нормою, що вимагає налаштувань і змін, щоб відповідати життєвому циклу Agile розробки.

5. Структура оцінки безпеки інформаційних систем (ISSAF) — це комплексна методологія та набір інструкцій, призначених для оцінки стану безпеки інформаційних систем. Вона пропонує методичний спосіб оцінити ефективність заходів безпеки та знайти недоліки та вразливі місця інформаційної системи в бізнесі.

Широко використовувана структура ISSAF допомагає підприємствам оцінювати безпеку їхніх інформаційних систем і інформувати про їхні стратегії управління ризиками та пом’якшення. Він пропонує методичну процедуру, яка може бути змінена відповідно до конкретних потреб і специфікацій різних підприємств.

Ключові елементи ISSAF:

Визначення обсягу: на цьому етапі системи, ресурси та процеси, які будуть оцінюватися, чітко визначаються як частина обсягу оцінювання. Це допомагає визначити ресурси, необхідні для оцінки, і створити реалістичні цілі.

Оцінка загроз: на цьому етапі визначаються потенційні загрози та вразливі місця. Це передбачає розуміння середовища загроз організації, яке включає внутрішні

та зовнішні небезпеки, такі як незаконний доступ і зловживання ресурсами, а також зовнішні загрози, такі як хакери, зловмисне програмне забезпечення та соціальна інженерія.

Оцінка вразливості: Метою цього кроку є виявлення слабких місць в інформаційних системах. Це передбачає виконання технічних оцінок, таких як сканування вразливостей і тестування на проникнення, щоб знайти недоліки, якими можуть скористатися зловмисники.

Аналіз ризиків: після виявлення вразливостей виконується аналіз ризиків, щоб визначити ймовірність і потенційні наслідки використання. Виходячи з їх важливості та потенційного впливу на організацію, це допомагає визначити пріоритетність виявлених ризиків.

Оцінка контролю: на цьому етапі оцінюється ефективність поточних засобів контролю безпеки та запобіжних заходів щодо зменшення виявлених ризиків. Це включає оцінку достатності та ефективності політики, практики, технічного контролю та програм підвищення обізнаності щодо безпеки.

Звітування та рекомендації: після завершення оцінювання складається ретельний звіт, у якому підсумовуються висновки, включаючи вразливості, ризики та пропозиції щодо покращення. Цей звіт є дорожньою картою для вдосконалення інформаційних систем безпеки організації.

Переваги методології:

- Комплексний підхід: для оцінки безпеки інформаційних систем ISSAF пропонує структурований та всеохоплюючий підхід. Він охоплює широкий спектр тем, забезпечуючи комплексну оцінку, включаючи оцінку загроз, оцінку вразливості, аналіз ризиків і оцінку контролю.
- Пріоритезація ризиків. Проводячи аналіз ризиків, ISSAF допомагає організаціям вирішити, на чому зосередити свої зусилля з безпеки. Це дає їм змогу зосередитися на ризиках і вразливостях, які мають найбільший потенціал для впливу на їх діяльність, і розумно розподіляти ресурси.
- Можливість налаштування: ISSAF можна змінювати відповідно до конкретних вимог і потреб різних організацій. Він пропонує універсальність з точки

зору використовуваної методології, форматів звітності та масштабу оцінки, що дозволяє компаніям налаштовувати його відповідно до власної ситуації.

- Поради щодо виправлення: для покращення стану безпеки інформаційних систем ISSAF пропонує практичні поради. Він містить корисні поради щодо встановлення засобів контролю безпеки, усунення вразливостей і підвищення безпеки інформаційної системи в цілому.

- Довіра в галузі: Експерти з інформаційної безпеки, консультанти та аудитори широко приймають і використовують ISSAF. Його прийняття може підвищити довіру до організації та продемонструвати відданість справі захисту інформаційних активів.

Недоліки методології:

- Складність: ISSAF може бути складним і вимагати великих витрат часу та ресурсів. Це передбачає низку процесів, процедур і оцінок, які можуть бути складними для фірм, які мають невеликі знання безпеки або фінансування.

- Вимоги до навичок і досвіду: для успішного впровадження ISSAF потрібні особи, які володіють спеціальними знаннями та досвідом оцінки інформаційної безпеки. Щоб досягти точних і ретельних оцінок, організаціям може знадобитися витратити гроші на навчання або найняти зовнішніх консультантів.

- Вимоги до ресурсів: Оцінювання ISSAF може бути досить ресурсомістким, вимагаючи багато часу, енергії та грошей. Малим підприємствам з обмеженими ресурсами може бути важко провести оцінку з необхідною глибиною та широтою.

- Потенційні збої: якщо життєво важливі системи піддаються поглибленому скануванню, тестуванню чи оцінюванню, сам процес оцінювання може погіршити існуючі операції. Оцінки повинні ретельно плануватися та координуватися організаціями, щоб зменшити будь-який потенційний шкідливий вплив на їх робочі процедури.

- Постійний моніторинг і технічне обслуговування: головною метою ISSAF є оцінка середовища безпеки в будь-який момент. Це може не дати повної відповіді щодо постійного моніторингу та підтримки інформаційних систем. Щоб

підтримувати довгострокову безпеку, організації повинні додати до ISSAF постійні процедури безпеки та системи моніторингу.

6. Федеральне відомство з інформаційної безпеки (BSI), також відоме як Bundesamt für Sicherheit in der Informationstechnik (BSIS), створило ретельну методологію тестування на проникнення, відому як Study A. Тестування на проникнення, яке часто називають етичним хакерством, є важливою процедурою для гарантування безпеки комп'ютерних мереж та інформаційних систем. Модель Study A від BSI пропонує структурований метод для проведення тестів на проникнення, що дозволяє компаніям виявляти слабкі місця та оцінювати ефективність своїх заходів безпеки. Модель тестування на проникнення BSI Study A складається з кількох етапів, які керують усім процесом тестування, від планування та підготовки до звітності та подальших дій. Розглянемо кожен етап детальніше:

Визначення обсягу: на цьому попередньому етапі встановлюються цілі. Співпраця між компанією та командою тестування на проникнення використовується для прийняття рішення про те, які системи, програми чи сегменти мережі будуть оцінюватися, а також для визначення точних цілей і обмежень тесту.

Тестери проникнення намагаються дізнатися якомога більше про цільові системи на етапі збору інформації. Ідентифікація цільових IP-адрес, імен доменів та інших відповідних мережевих даних є частиною цього процесу. Мета полягає в тому, щоб зрозуміти архітектуру, інфраструктуру та потенційні точки доступу цілі.

Моделювання загроз: на цьому етапі тестери перевіряють отриману інформацію, щоб знайти потенційні загрози та слабкі місця. Вони оцінюють значущість і ймовірність цих небезпек і ранжують їх за серйозністю. Цей крок допомагає визначити зони фокусування для тестування.

Аналіз вразливостей: на цьому етапі тестувальники активно шукають слабкі місця в цільових системах. Щоб знайти недоліки, якими можуть скористатися зловмисники, вони використовують різноманітні методи, включаючи ручне тестування, сканування мережі та сканування вразливостей. Для досягнення ретельної оцінки випробувачі використовують як ручну перевірку, так і автоматизовані методи.

Експлуатація вразливостей: після виявлення вразливостей тестери проникнення намагаються використати їх для отримання доступу до цільових систем або контролю над ними без авторизації. Щоб визначити ступінь, до якого повідомлені вразливості можуть бути використані зловмисниками, на цьому етапі моделюються реальні сценарії атак. Після успішної експлуатації тестери оцінюють вплив та будь-які наслідки порушення. Це відомо як аналіз після експлуатації. Вони вивчають потенційний доступ до особистих даних і вплив на продуктивність системи. Завдяки цьому дослідженню організації можуть краще зрозуміти ризики, пов’язані з виявленими вразливими місцями.

Звітування: надається вичерпний звіт із детальним описом результатів тесту на проникнення. Процедура тестування підсумовується у звіті разом із знайденими вразливими місцями, їх рівнями серйозності та пропозиціями щодо виправлення. Звіт є важливим інструментом, який компанії можуть використовувати для усунення виявлених порушень безпеки та покращення загальної безпеки. Після завершення тесту на проникнення підприємства повинні вжити необхідних заходів для усунення виявлених вразливостей і посилити заходи безпеки. Цей етап передбачає введення в дію запропонованих коригувальних дій, як-от виправлення програмного забезпечення, перевпорядкування систем або покращення контролю доступу. Також можна порадити проводити регулярні повторні перевірки, щоб підтвердити ефективність встановлених заходів безпеки.

Переваги методики:

- Підхід зі структурою: модель BSI Study A пропонує чіткий і організований метод проведення тестів на проникнення. Це гарантує, що процедура тестування є вичерпною, структурованою та послідовною, що дозволяє підприємствам успішно виявляти вразливості.
- Комплексна методологія: модель включає визначення обсягу, збір інформації, моделювання загроз, аналіз вразливості, використання, аналіз після використання, звітування та подальші дії. Він також охоплює всі найважливіші етапи тестування на проникнення. Ця ретельна методологія гарантує, що під час процесу тестування не буде упущено жодної важливої деталі.

- Дійсні рекомендації: одель тестування на проникнення пропонує організаціям практичні пропозиції щодо усунення знайдених уразливостей. Завдяки детальному звіту, створеному в кінці тесту, організації можуть зрозуміти основні процедури для ефективного управління ризиками.

- Дослідження BSI щодо відповідності та найкращих практик. Модель відповідає галузевим нормам, законам і рекомендованим методам тестування на проникнення. Прийнявши цю стратегію, організації можуть покращити загальне управління безпекою та продемонструвати відповідність відповідним нормам безпеки.

- Пріоритезація ризиків відповідно до їх серйозності є ключовим компонентом методології. Організації можуть краще керувати всією системою безпеки, розподіляючи ресурси та вирішуючи найсерйозніші ризики, спочатку оцінюючи вплив і ймовірність виявлених вразливостей.

Недоліки методології:

- Ретельне дослідження BSI вимагає багато часу та ресурсів. Оскільки ця методологія є модельною, проведення тестування на проникнення може потребувати часу та ресурсів. Щоб успішно завершити процес тестування, організації повинні виділити достатньо часу, ресурсів і грошей.

- Вимоги до кваліфікації: для успішного впровадження моделі тестування на проникнення потрібні кваліфіковані співробітники. Щоб правильно виявити вразливості та оцінити ризики, організації повинні мати доступ до досвідчених тестувальників проникнення, які знайомі з моделлю.

- Дослідження BSI: обмежена адаптивність. Модель може не покривати всі потреби чи сценарії, з якими може зіткнутися організація. Щоб отримати повне охоплення тестуванням, компанії повинні модифікувати модель відповідно до своїх унікальних вимог і, якщо необхідно, інтегрувати її з іншими методологіями чи підходами.

- Відсутність гнучкості: хоча систематичний підхід моделі забезпечує повноту, він також може накласти обмеження на гнучкість. Організації можуть мати

певні потреби або обмеження, які попередньо встановлені етапи моделі не можуть повністю вирішити.

- Складність дослідження BSI Для фірм, які мають невеликий досвід або фінансування, модель може вважатися складною. Для успішної реалізації кожного етапу моделі потрібен певний рівень технічної компетентності.

2.4 Способи захисту від атак на веб-додатки

2.4.1 Захист від SQL-ін’єкцій

1. Використовуючи параметризовані запити, можна відокремити введені користувачем дані від коду SQL, який виконує база даних. У параметризованому запиті в коді SQL використовуються заповнювачі, а не фактичні дані користувача. Введені користувачем дані вводяться в заповнювачі за допомогою підготовленого оператора під час виконання запиту. Оскільки введені користувачем дані розглядаються як дані, а не код у цьому методі, атаки SQL-ін’єкцій менш імовірні.

2. Перевірка введень користувача: Перевірка введень включає в себе переконання, що вони відповідають бажаним значенням і типам даних. Ви можете зупинити атаки, які покладаються на вставку коду SQL у введені користувачем дані, перевіривши введені користувачем дані. Ви можете перевірити, наприклад, чи введене число знаходиться в заданому діапазоні або що введений рядок не містить жодних неправильних символів. Клієнтська та серверна сторони повинні виконувати перевірку вхідних даних.

3. Найменші привілеї: практика надання користувачам лише мінімальних привілеїв, необхідних для виконання їхніх обов’язків, відповідає ідеї найменших привілеїв. Ви можете зменшити наслідки успішної атаки SQL-ін’єкції, обмеживши привілеї користувачів бази даних. Зловмисник може не мати доступу або змінити конфіденційні дані, наприклад, якщо він може отримати доступ до бази даних за допомогою скомпрометованого облікового запису з низькими привілеями.

4. Повідомлення про помилки. Повідомлення про помилки можуть випадково розкривати подробиці про дизайн вашої бази даних або код SQL, який виконується, що може бути корисним для зловмисників. Щоб запобігти цьому, повідомлення про

помилки мають бути якомога загальнішими. Наприклад, ви можете надіслати загальне повідомлення, яке лише повідомляє про помилку, а не конкретне повідомлення про помилку, яке містить запит SQL, який було виконано.

5. Уникайте динамічного SQL: динамічний SQL - це код SQL, який створюється під час виконання на основі введення користувача або інших обставин. Оскільки динамічний SQL дозволяє модифікувати код SQL, який виконує база даних, зловмисникам може бути простіше вставити шкідливий код. Якщо вам потрібно використовувати динамічний SQL, переконайтеся, що використовуються параметризовані запити та що введення користувача адекватно оцінюється.

6. Безпека бази даних передбачає вживання запобіжних заходів, щоб захистити вашу базу даних від незаконного доступу та маніпуляцій. Це може включати налаштування контролю доступу для обмеження привілеїв користувачів бази даних, використання брандмауерів для обмеження доступу до вашої бази даних і шифрування важливих даних. Щоб захиститися від відомих уразливостей, ви також повинні постійно оновлювати програмне забезпечення своєї бази даних з найновішими виправленнями безпеки.

2.4.2 Захист від CSRF-атак

1. Використовуйте токени CSRF: CSRF токен - це ідентифікатор, який генерується спеціально для кожного сеансу або запиту користувача. Токен надсилається із сервера клієнту, який потім включає його в усі майбутні запити. Коли надходить запит із токеном CSRF, сервер перевіряє, чи справжній токен для цього конкретного користувача, і відхиляє будь-які запити без дійсного токена. Цей метод допомагає запобігти зловмисникам надсилати неавторизовані запити від імені жертви.

2. Обмежити доступ за допомогою файлів cookie SameSite: SameSite - це атрибут cookie, який дозволяє серверам контролювати, як файли cookie надсилаються з міжсайтовими запитами. Обмежуючи файли cookie запитами з однаковим джерелом, жорсткий або слабкий режими атрибута SameSite можуть допомогти уникнути атак CSRF. Лише навігація верхнього рівня та запити POST на тому самому

сайті включатимуть файли cookie з атрибутом SameSite, встановленим у суворий режим, тоді як файли cookie зі слабким режимом не включатимуть жодних міжсайтових запитів.

3. Використовуйте HTTP-only cookie: Файл cookie, який можна отримати лише через HTTP-запити, і до якого не можуть отримати доступ сценарії на стороні клієнта, є HTTP-only cookie. Цей метод допомагає запобігти атакам міжсайтових сценаріїв (XSS) від надання зловмисникам доступу до конфіденційних даних, таких як ідентифікатори сеансів.

4. Перевірте заголовок Referrer: заголовок запиту під назвою Referrer містить URL-адресу веб-сайту, з якого надійшов запит. Переконавшись, що запит надійшов із надійного джерела, перевірка заголовка Referrer може допомогти зупинити атаки CSRF. Заголовок Referrer може бути підроблений, тому важливо пам’ятати, що це не надійний захист.

5. Активувати багатофакторну автентифікацію (MFA): MFA - це метод безпеки, який просить користувачів надати два або більше елементів ідентифікації, перш ніж надати доступ до своїх облікових записів. Користувача можуть попросити ввести як пароль, так і код, наданий на його мобільний пристрій, наприклад. Навіть якщо зловмисник отримав облікові дані користувача за допомогою CSRF-атаки, MFA додає додатковий рівень безпеки, який може допомогти запобігти його доступу до конфіденційних даних.

6.Оновлення програмного забезпечення: найкращий спосіб зупинити атаки CSRF – підтримувати все програмне забезпечення в актуальному стані з найновішими виправленнями та оновленнями безпеки. Веб-сервери, фреймворки та бібліотеки можуть мати недоліки безпеки, які хакери можуть використати проти них. Регулярні оновлення програмного забезпечення забезпечують наявність найновіших патчів безпеки та знижують уразливість програми.

2.4.3 Захист від XSS-атак

1. Перевірка будь-яких вхідних даних від користувача чи будь-якого іншого джерела є одним із найважливіших кроків у запобіганні атакам XSS. Це означає, що

дані мають правильний тип, довжину та формат, а також не містять небезпечного коду чи символів. Очистіть і перевірте введені користувачем дані за допомогою методів перевірки на стороні сервера, таких як регулярні вирази або бібліотеки, такі як OWASP ESAPI.

2. Кодування виводу запобігає обробці спеціальних символів як коду HTML або JavaScript, перекладаючи їх у еквівалентні об’єкти HTML. Перш ніж відтворювати будь-який вивід на веб-сторінці, закодуйте його за допомогою бібліотек для кодування виводу, таких як OWASP Java Encoder, Apache Commons Text або ESAPI. Ви також можете використовувати фреймворки дезінфекції введення та виведення даних, такі як Angular або React.

3. Використовуйте бібліотеки безпеки: для захисту від XSS-атак доступно кілька бібліотек безпеки. Однією з таких бібліотек є OWASP ESAPI, набір API для перевірки вхідних даних, вихідного кодування та інших завдань, пов’язаних із безпекою. Інші бібліотеки, такі як HTML Purifier і AntiXSS, призначені для очищення введених користувачами даних і позбавлення від будь-якого небезпечного матеріалу.

4. Використовуйте HTTPS, щоб переконатися, що дані, що передаються між вашим веб-сайтом і браузерами користувачів, зашифровані та безпечні. HTTPS означає безпечний HTTP. Це запобігає будь-яким зловмисникам від підслуховування розмови або будь-яким чином втручання в неї, зокрема шляхом впровадження шкідливих сценаріїв.

5. Обмежити доступ до файлів cookie: Ви можете заблокувати клієнтським сценаріям доступ до файлів cookie, установивши параметр HttpOnly для кожного з них. Це гарантує, що навіть якщо хакерам вдасться вставити шкідливі сценарії на веб-сторінку, облікові дані користувача та інша важлива інформація не будуть доступні для них.

6. Будьте в курсі подій: фреймворки та бібліотеки для веб-додатків часто оновлюються, щоб виправити будь-які недоліки безпеки або вразливості. Щоб ваш веб-сайт був у безпеці та захищений від нових атак, слідкуйте за найновішими виправленнями безпеки та оновленнями.

7. Користувачів слід навчити: повідомте своїх користувачів про небезпеку XSS-атак і про те, як їм запобігти, порадивши їм не клацати сумнівні посилання та не завантажувати файли з незнайомих їм сайтів. Також слід порадити використовувати розширення браузера, які блокують небезпечні скрипти, і використовувати сучасний браузер, який підтримує захист XSS.

2.4.4 Захист від Broken authentication атаки

1. Впроваджуйте стандарти надійних паролів, вимагаючи від користувачів створювати довгі паролі, які містять комбінацію великих і малих літер, цифр і спеціальних символів і мають довжину принаймні 12 символів. Застосуйте багатофакторну автентифікацію для підвищення безпеки. Це означає, що користувачі повинні генерувати безпечні паролі та часто їх змінювати. Надійні паролі мають бути складними для підбору чи вгадування. Крім того, використання кількох форм автентифікації, таких як пароль і код, який надсилається на телефон користувача, може значно підвищити безпеку.

2. Впровадження процедур, які гарантують безпеку сеансів користувача, є безпечним керуванням сеансами. Щоб зробити це, необхідно суворо дотримуватися тайм-аутів сеансів, щоб обмежити тривалість сеансів користувачів, і сеанси користувачів мають бути належним чином зупинені, коли користувачі виходять із системи. Щоб захистити дані сеансу користувача від перехоплення, також потрібні безпечні файли cookie та примусовий протокол HTTPS.

3. Захист від атак грубою силою: під час атак грубою силою імена користувачів і паролі автоматично підбираються. Програми повинні містити функції обмеження швидкості, які обмежують кількість невдалих спроб входу з однієї IP-адреси для захисту від цих атак. Це може допомогти зупинити зловмисників від спроб численних комбінацій для вгадування паролів. Крім того, автоматичні атаки можна запобігти за допомогою CAPTCHA.

4. Використовуйте сучасні механізми аутентифікації: це передбачає використання добре встановлених, перевірених і безпечних методів автентифікації, таких як OAuth і OpenID Connect, які є галузевими стандартами. Уникайте

використання індивідуальних методів автентифікації, оскільки вони можуть мати діри в безпеці, якими можуть скористатися хакери.

5. Використовуйте сканери вразливостей і інструменти тестування на проникнення, щоб знаходити та вирішувати проблеми безпеки в процесах автентифікації. Регулярно тестуйте та перевіряйте свої механізми автентифікації. Ви можете виявити спроби несанкціонованого доступу та інші події безпеки, регулярно перевіряючи свої журнали.

2.4.5 Захист від витоку конфідеційних даних

Захист конфіденційних даних від витоку вимагає впровадження надійних заходів захисту. Ось кілька основних стратегій, які слід розглянути:

1. Використовувати надійні алгоритми шифрування, щоб захистити конфіденційні дані, коли вони знаходяться в стані спокою та під час передачі. Дані, які були зашифровані, гарантовано залишаються незрозумілими, навіть якщо їх перехопити або переглянути без дозволу.

2. Застосувати суворий контроль доступу, щоб обмежити доступ, зміну або видалення конфіденційних даних. Використання надійних методів автентифікації, таких як багатофакторна автентифікація (MFA).

3. Мінімізація даних: збирайте та зберігайте лише абсолютний мінімум даних. Якщо ви зберігаєте якомога менше конфіденційної інформації, ваш ризик виявлення у разі порушення є меншим.

4. Безпечне зберігання даних: зберігайте особисту інформацію в безпечних місцях, включаючи зашифровані бази даних або приватне хмарне сховище. Регулярно встановлюйте оновлення безпеки та виправлення, щоб забезпечити швидке усунення вразливостей.

5. Запровадити надійний механізм резервного копіювання, для того щоб забезпечити можливість відновлення даних у разі втрати внаслідок помилки, лиха чи злому. Резервні копії мають бути зашифровані та зберігатися в іншому місці.

6. Навчання та обізнаність співробітників: є обов’язковим інформування співробітників про важливість безпеки даних і надання їм інструкції щодо найкращих

способів обробки конфіденційних даних, важливість захисту паролів, виявлення фішингових шахраїв і дотримання законів про захист даних.

7. Забезпечити передачу даних через мережі з використанням безпечних протоколів, таких як HTTPS, щоб забезпечити безпечну передачу даних. Не надсилайте конфіденційну інформацію через незахищені публічні мережі Wi-Fi.

8. Регулярні перевірки безпеки: щоб знайти та виправити будь-які недоліки у ваших системах, проводите регулярні перевірки безпеки та оцінки вразливостей. Щоб симулювати потенційні атаки та знаходити недоліки, тестування на проникнення є частиною цього.

9. Створення ретельної стратегії реагування на інциденти, яка слугуватиме дорожньою картою для дій у разі порушення конфіденційності даних або їх розкриття. Дії щодо стримування, розслідування, сповіщення та відновлення повинні бути викладені в цій стратегії.

Висновки до другого розділу

В даному розділі було описано існуючі інструменти та методології для аналізу веб-експлойтів, також описано як можна захиститись від різних видів атак на веб-додатки щоб уникнути викрадення конфіденційної інформації, неавторизованого доступу до даних та нанесення шкоди звичайним користувачам системи.

Технології та компоненти, що використовуються під час розробки веб-додатків, а також будь-які потенційні вразливості цих компонентів впливають на те, наскільки добре онлайн-додатки захищені від зловмисників. Існують різні категорії вразливостей, і кожна атака вразливості має свої особливості. Однак помилки в розробці, реалізації та використанні компонентів веб-додатків є причиною вразливостей, що вимагає їх виявлення та виправлення. Як в Україні, так і в інших країнах формуються бригади реагування на надзвичайні ситуації у складі фахівців і дослідників.

Проте процедура розкриття вразливості регулюється кількома міжнародними правилами. Шукати вразливості можна за допомогою різноманітних технологій, але ефективність їх використання залежить від алгоритму завдань, які цей пошук має

виконати. Щоб охопити широкий спектр питань кібербезпеки, таких як тестування безпеки бездротових мереж, операційних систем і фізичного середовища, алгоритми дій можуть бути представлені у вигляді таких спеціалізованих методів; однак це вимагає більше часу для аналізу існуючих методів і вибору таких компонентів.

РОЗДІЛ 3
РОЗРОБКА МЕТОДУ АНАЛІЗУ ВЕБ-ЕКСПЛОЙТІВ

3.1 Вимоги до методу аналізу веб-експлойтів націлених на веб-додатки

Необхідно сформулювати вимоги до методу, створеного з метою аналізу веб-додатку на наявність уразливостей критичного рівня, щоб створити ефективний метод, який б дозволяв аналізувати веб-додатки на основі списку вразливостей:

– дати детальний опис техніки підготовки до аналізу ;

– ознайомитись з підходами до аналізу ;

– опрацювати основні терміни які відносяться до ІБ ;

– описати інструменти які будуть використані для аналізу;

– детально описати техніки перевірки безпеки комп’ютерної системи ;

– посилатись на ПО, яке використовується для аналізу;

– додавати посилання на нормативні документи та методики;

Бажано розділити аналіз на наступні етапи, беручи до уваги специфікації, яким має відповідати техніка, і пропозиції в інших методологіях:

– збір інформації про систему;

– аналіз компонентів веб- додатка, які містять зазначені вразливості;

– аналіз засобів перевірки вхідних даних до веб – додатка;

– аналіз засобів автентифікації;

– аналіз витоку конфіденційних даних;

– аналіз засобів контролю доступу;

– підготовка звіту

3.2 Розробка методу аналізу

3.2.1 Збір інформації про систему

Для етапу збору інформації було обрано вбудований в ОС Kali Linux сканер Nmap. Безкоштовний інструмент із відкритим вихідним кодом для дослідження мережі та тестування безпеки називається Nmap (скорочення від «Network Mapper»). Результати сканування залежать від встановлених параметрів і показують, які вузли доступні в мережі, які служби (назва програми та версія) надаються цими вузлами, які операційні системи (і версії ОС) вони використовують, які типи фільтрів і брандмауерів використовуються, і багато інших деталей. Список параметрів:

-sL: цей параметр повідомляє перерахувати цілі, які будуть скановані, фактично не надсилаючи їм жодних пакетів.

-sn: цей параметр вимикає сканування портів і виконує лише виявлення хоста, надсилаючи exo-запити ICMP і запити ARP.

-sS: цей параметр вмикає приховане сканування TCP SYN.

-sT: цей параметр вмикає сканування підключення TCP.

-sA: цей параметр вмикає сканування TCP ACK. Він надсилає пакети TCP ACK на цільові порти та аналізує відповіді, щоб визначити, чи фільтруються вони брандмауером.

-sW: цей параметр вмикає сканування вікон TCP.

-sM: цей параметр вмикає сканування TCP Maimon

-sU: цей параметр вмикає сканування UDP, яке надсилає пакети UDP для визначення відкритих портів UDP на цільовому пристрої.

-sN: вмикає стелс-сканування TCP Null, яке надсилає TCP-пакети без встановлених прапорів і аналізує відповідь для визначення відкритих портів.

-sF: ця опція вмикає сканування TCP FIN

-sX: ця опція вмикає сканування TCP Xmas

-p : цей параметр визначає діапазон портів для сканування. У цьому випадку конкретний діапазон портів не вказано, тому Nmap скануватиме всі порти.

-sV: ця опція вмикає визначення версії, яка намагається визначити програмне забезпечення та номер версії служб, що працюють на відкритих портах.

-O: цей параметр вмикає виявлення операційної системи, яке намагається визначити операційну систему, запущену на цільовому комп’ютері.

-A: вмикає «агресивне» сканування, яке включає виявлення ОС, визначення версій, сканування сценаріїв і аналіз трасування на додаток до інших типів сканування.

Результат перевірки відкритих портів та сканування запущених сервері служб наведено на рисунку 3.1.

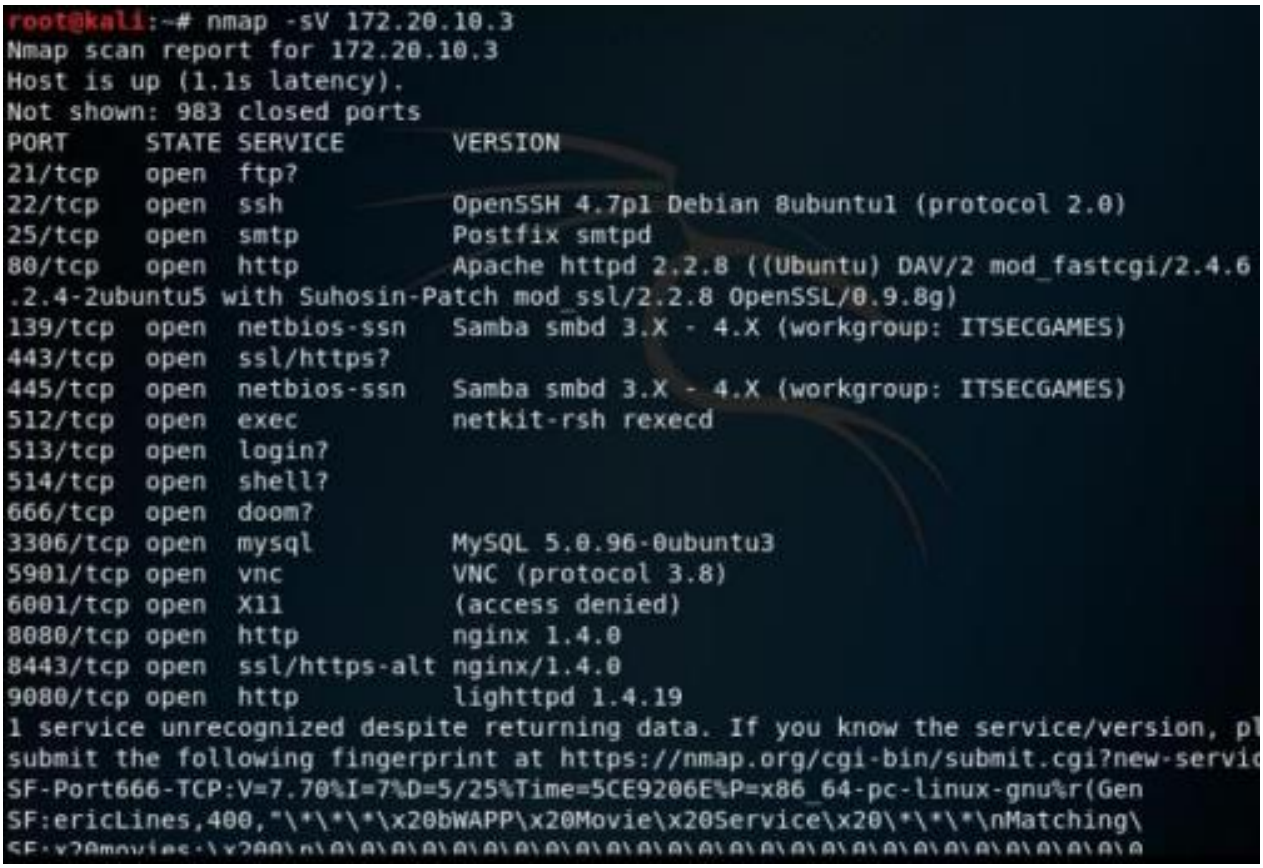

Рисунок 3.1 – Результат перевірки відкритих портів та запущених служб

Результат команди за допомогою якою можемо визначити тип ОС наведено на рисунку 3.2.

Рисунок 3.2 – Результат визначення ОС

Тож завдяки такій універсальній утиліти nmap ми змогли отримати інформацію, що на веб-сервері знаходяться такі сервіси: OpenSSH, Apache, Samba, MySQL.

А також можливі ОС такі як: Actiontec MI424WR-GEN3I WAP (99%), Linux 3.2 (98%), Linux 4.4 (96%), Microsoft Windows XP (96%)

3.2.2 Аналіз компонентів веб-додатку, в яких можливі вразливості

Після того як була одержана інформація про систему є необхідність перевірки наявності веб-експлойтів.

Ми шукаємо відомі вразливості для серверних програм, використовуючи знання, отримані під час першого етапу, і інструмент searchsploit який нам представляє ОС Kali Linux. Використовуючи Exploit Database, інструмент searchsploit шукає потенційні атаки, які можуть скористатися відкритими вразливими місцями.

Щоб досягти поставленої мети потрібно перебрати сервіси які були визначені на першому етапі.

- Аналіз searchsploit apache можна побачити на рисунку 3.3

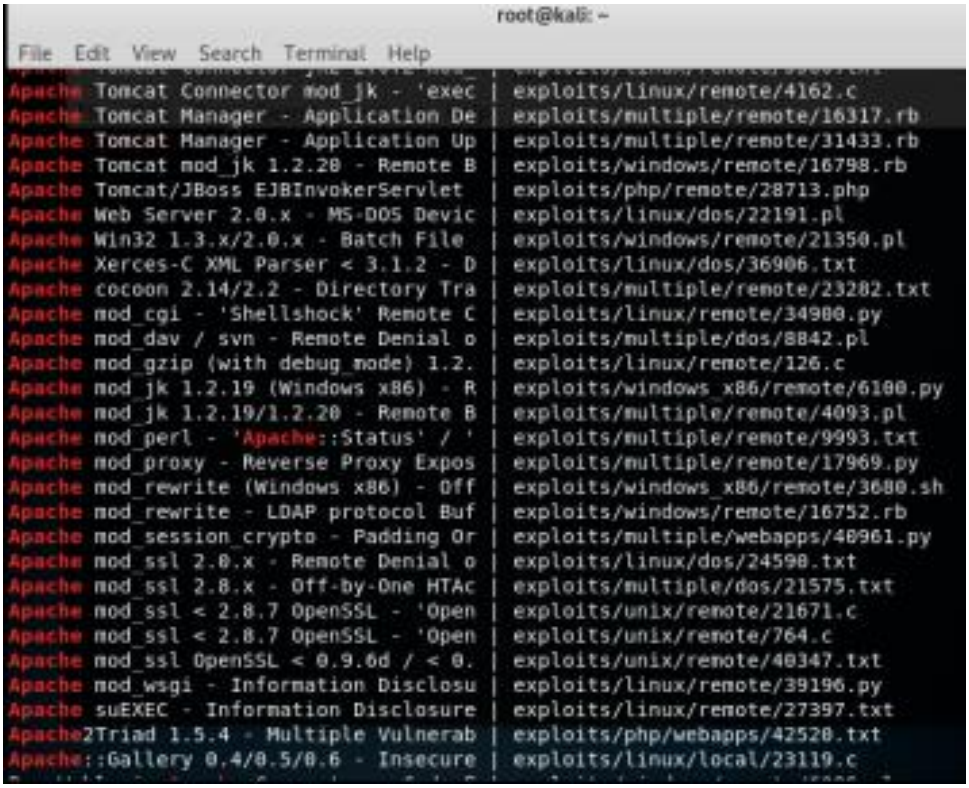

Рисунок 3.3 – Результат аналізу експлойтів для apache

- Аналіз searchsploit openssh можна побачити на рисунку 3.4

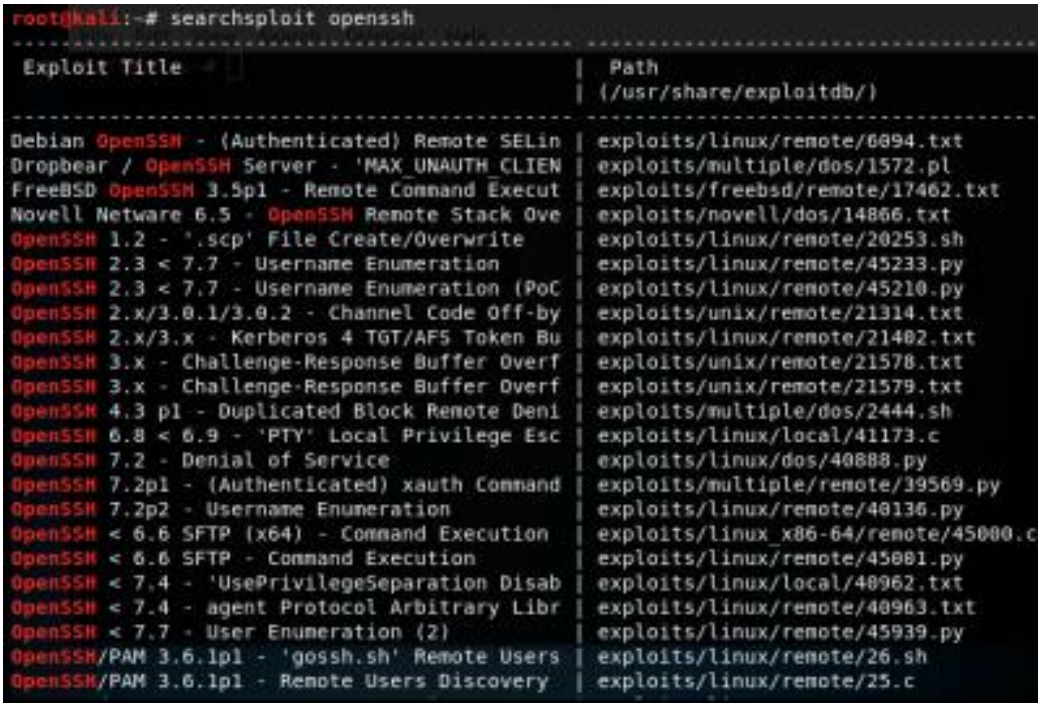

Рисунок 3.4 – Результат аналізу експлойтів для openssh

- Аналіз searchsploit mysql можна побачити на рисунку 3.5

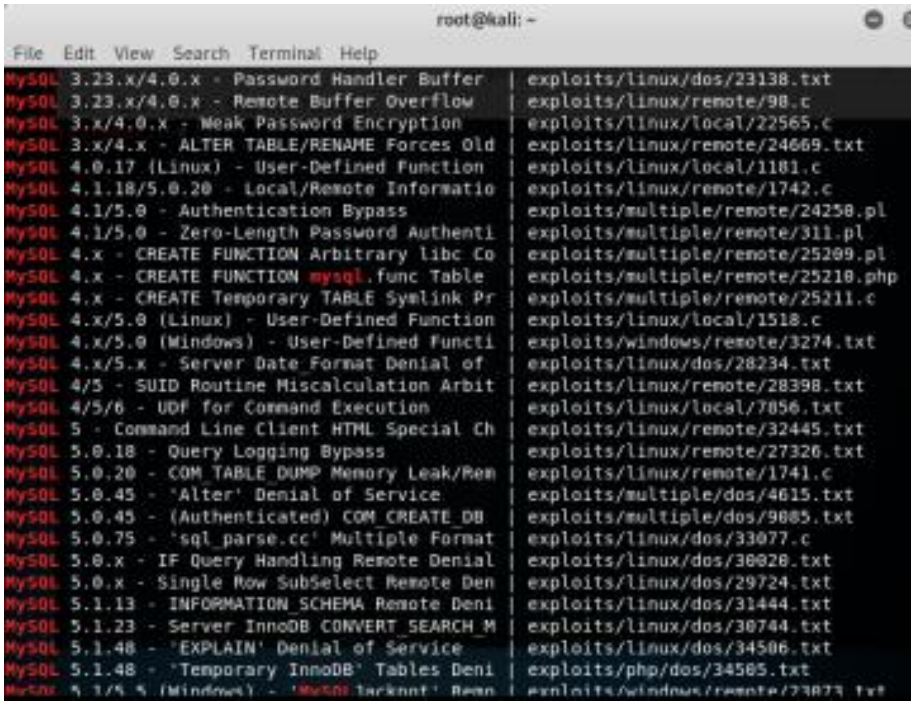

Рисунок 3.5 – Результат аналізу експлойтів для mysql

3.2.3 Аналіз на можливість SQL-ін’єкцій

Аби провести перевірку на можливість виконання SQL-ін’єкції не так вже й складно. Іноді може вистачити просто додати ' або “ в поля які ми маємо намір перевірити. Безперечно якщо з’являється якесь повідомлення, то можна із повною впевненістю сказати що для цього поля можлива SQL-ін'єкція.

Вводимо ‘iron man’ для перевірки роботи таблиці, та бачимо що результат вірний. Висвітлилась інформація про фільм який був вписаний, і не видно ніяких повідомлень про помилки це затверджує результат на рисунку 3.6.

Рисунок 3.6 – Результат перевірки в полі введення

Далі вже вводимо, як було сказано раніше, ‘iron man`’ і на рисунку 3.7 можна побачити що виникло повідомлення про помилку, з цього можна зробити висновок що існує можливість проведення SQL-ін'єкції.

Рисунок 3.7 – Підтвердження що SQL-ін'єкція можлива

В Kali Linux виконуємо команду:

«sqlmap –u `http://192.168.31.130:80/bWAPP/sqli_1.php?title=` --cookie=`PHPSESSID= b2a7d02cc634679f754fe391fb25f304`»

Результат виконаної команди можна побачити на рисунку 3.8:

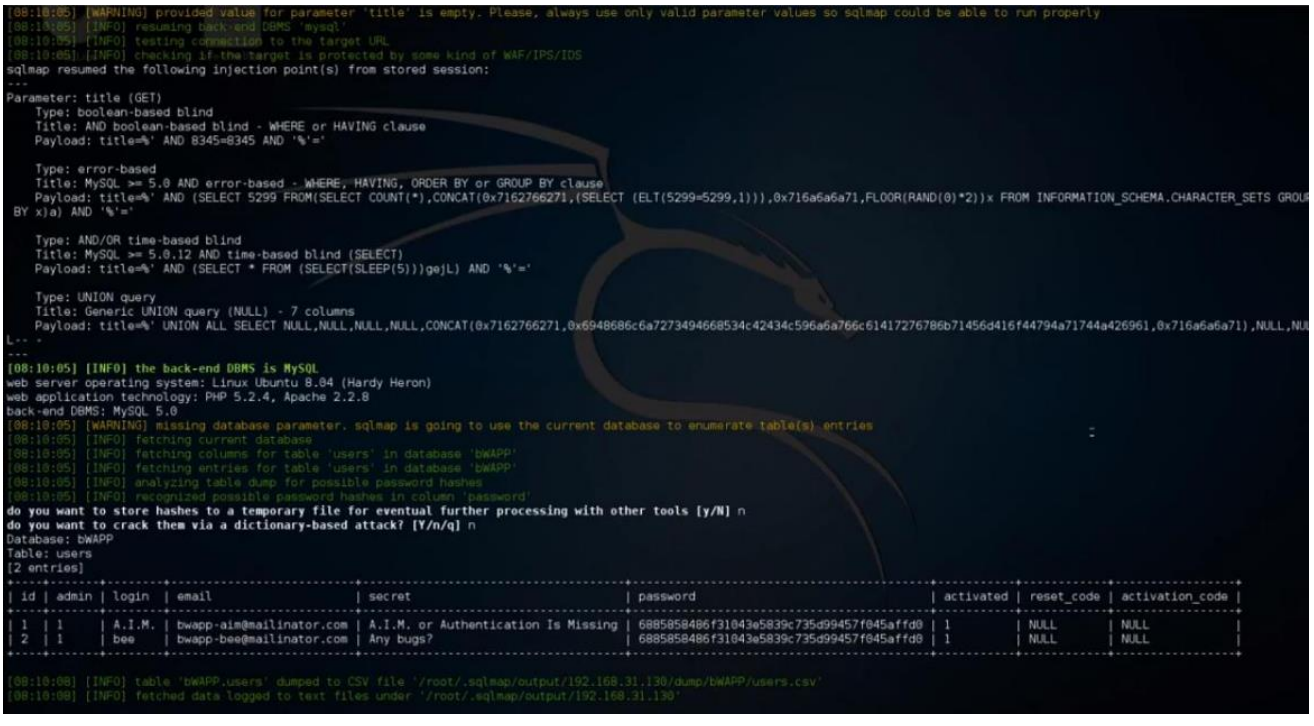

Рисунок 3.8 – Результат команди sqlmap

3.2.4 Аналіз на можливість XSS-атаки

Щоб дізнатись чи є наявність XSS вразливостей потрібно просто перевірити, чи відповідає веб-додаток на запити в яких містяться прості сценарії які можуть бути виконані веб-браузером.

Для перевірки цього у поле реєстрації користувача відправляємо такі дані: ‘<script>alert(/Hacked/)</script)’. Цей сценарій просто виводить надпис в окремому віконці. На рисунку 3.9 можна побачити введення сценарію.

Рисунок 3.9 – Введення сценарію в поле реєстрації

Якщо вразливість яка призводить до XSS-атаки існує, то перед користувачем вспливе вікно яке можна побачити на рисунку 3.10:

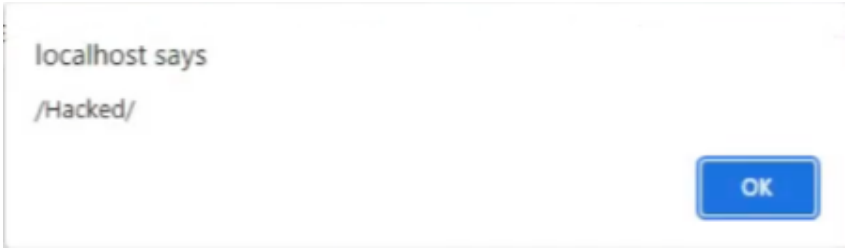

Рисунок 3.10 – Результат успішної XSS-атаки

3.2.5 Аналіз на можливість CSRF-атаки

Для аналізу потрібно знати URL-адреси в обмеженій (автентифікованій) зоні. Вони можуть грати як жертву, так і злочинця, якщо мають законні повноваження. У цій ситуації просто перегляд програми покаже URL-адреси, які потрібно перевірити.

Якщо немає належних облікових даних, доведеться розпочати реальну атаку, щоб змусити законного користувача увійти за посиланням. Для цього може знадобитися соціальна інженерія.

Аналіз може бути побудований таким чином:

- дайте URL-адресу, що проходить аналіз; наприклад, http://www.example.com/

- напишіть HTML-сторінку, яка буде містити запит http, який посилається на URL;

- запевніться, що користувач зайшов до веб-додатку;

- підштовхувати користувача до переходу за посиланням, який посилається на URL-адресу, що підлягає перевірці;

- переглядати результат, чи був запит на веб-сервер.

2. Провести перевірку веб-додатку, щоб переконатися, що сеанси вразливі. Якщо керування сеансом на значення клієнту, то додаток є вразливим. Значення на стороні клієнта мається на увазі файли cookie та ідентифікаційні дані HTTP-аутентифікації. Інформація в URL, у вигляді неідентифікованого або непередбачуваного користувачем. Хоча POST-запит можна автоматизувати за допомогою JavaScript і він також є вразливим, ресурси, отримані через запити HTTP GET, легше викрити. Як наслідок, використання лише POST недостатньо для запобігання вразливості CSRF.

Висновок до третього розділу

У третьому розділі дипломної роботи ми розробили метод, який враховує переваги міжнародних методологій аналізу веб-експлойтів.

За допомогою розробленого підходу перевіряються лише основні уразливості зі списку OWASP тобто виходить збільшення швидкості аналізу при зменшенні кількості необхідних перевірок.

Структура методу розбита на етапи, кожен з яких уточнюється необхідними перевірками та ілюстраціями на етапах аналізу. Запропонований метод описаний з урахуванням потенційних знахідок, які можуть вказувати на існування вразливостей. Він також ілюструє використання інструментів аналізу експлойтів та виявлення вразливостей.

Оскільки запропонований метод тестує тільки критичні вразливості та виходячи з цього використовує невелику кількість етапів, то можна сказати що тестування ефективне.

ВИСНОВОК

У кваліфікаційній магістерській роботі було розроблено метод потенційного підвищення ефективності та швидкості аналізу веб-експлойтів на основі мови програмування JavaScript. Для досягнення поставленої мети роботи було проведено обстеження функціонування та стану веб-додатків. Крім того, було виявлено й оцінено поширеність уразливостей веб-додатків і потенціал атак через недоліки, оскільки пошук уразливостей є одним із найважливіших етапом аналізу. Також згідно з дослідженнями було проаналізовано існуючі інструменти які використовують в світі для аналізу.

Крім того, було визначено що більшість методів охоплюють широкий спектр проблем кібербезпеки, тому необхідний додатковий час, який витрачається на аналіз вразливостей відповідно до існуючих методів і вибір, зокрема, тих компонентів, які підходять для аналізу веб-додатків.

За допомогою результатів аналізу та досліження було розроблено адаптований метод який враховує міжнародні досягнення в цьому напрямку. Завдяки цьому метод дозволяє перевіряти наявність найпоширеніших уразливостей. Ефективність також підвищується в результаті рішення оцінювати лише недоліки з критичним рівнем ризику.

Практичне значення роботи полягає в скороченні часу та обґрунтуванні вибору засобів аналізу. Результати дипломної роботи можуть бути використані для аналізу експлойтів в веб-додатках.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ganore P. What Is A Web Server And How Does It Function? [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.milesweb.com/blog/hosting/web-server-function/.

2. What is a web server [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:

https://developer.mozilla.org/ru/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server.

3. Tamara J. What Is A Web Server. How it works and more.[Електронний ресурс] – 2023 – Режим доступу до ресурсу: https://www.hostinger.com/tutorials/what-is-a-web-server.

4. OWASP (Open Web Application Security Project) Top 10 Web Application Security Risks. [Електронний ресурс] – 2021 – Режим доступу до ресурсу: https://owasp.org/Top10/.

5. Which web attack is a server side attack. [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.alibabacloud.com/tech-news/web-server/giqt1yos1h-which-web-attack-is-a-server-side-attack.

6. What is client side exploitation. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/what-is-client-side-exploitation/.

7. What is an Exploit [Електронний ресурс] – Режим доступу до ресурсу: https://www.fortinet.com/resources/cyberglossary/exploit#:~:text=An%20exploit%20(in%20its%20noun,horses%2C%20worms%2C%20or%20viruses.

8. Bryan Sullivan, Vincent Liu. Web Application Security: A Beginner's Guide [Електронний ресурс] – 2011 – Режим доступу до ресурсу: http://surl.li/gkuon.

9. What is JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

10. JavaScript Security. JavaScript vulnerabilities and best practices explained. [Електронний ресурс]. – Режим доступу до ресурсу: https://snyk.io/learn/javascript-security/.

11. JavaScript DOM Tree [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://javascript.info/dom-nodes.

12. Common Vulnerability Scoring System (CVSS). [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.techtarget.com/searchsecurity/definition/CVSS-Common-Vulnerability-Scoring-System.

13. Common Vulnerability Scoring System v3.0: Specification Document [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.first.org/cvss/specification-document.

14. OWASP SQL-injection. [Електронний ресурс]. – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/SQL_Injection.

15. What is SQL Injection(SQLi) and How to Prevent It. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.acunetix.com/websitesecurity/sql-injection/.

16. OWASP Cross-Site Request Forgery (CSRF) [Електронний ресурс]. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).

17. Cross Site Scripting (XSS) Attack Tutorial With Examples, Types & Prevention. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/.

18. What is XSS? Impact, Types, and Prevention. [Електронний ресурс]. – 2022 – Режим доступу до ресурсу: https://brightsec.com/blog/xss/.

19. Diego P. What is Broken Authentication. [Електронний ресурс]. – 2020 – Режим доступу до ресурсу: https://auth0.com/blog/what-is-broken-authentication/.

20. Broken Authentication: What is It and How to Prevent IT. [Електронний ресурс]. – 2022 – Режим доступу до ресурсу: https://www.authgear.com/post/broken-authentication-what-is-it-and-how-to-prevent-it.

21. What is Sensitive Data Exposure Vulnerability and How to Avoid it. [Електронний ресурс]. – Режим доступу до ресурсу: https://securiti.ai/blog/sensitive-data-exposure/.

22. Sensitive Data Exposure: What is It and How to Avoid it? [Електронний ресурс]. – 2022 – Режим доступу до ресурсу: https://www.polar.security/post/sensitive-data-exposure.

23. Phongphun K. , Rattikorn H. Exploit-based Analysis of Attack Models. [Електронний ресурс] – 2013 – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/6623661.

24. Difference Between Static Malware Analysis and Dynamic Malware Analysis. [Електронний ресурс]. – Режим доступу до ресурсу: http://www.differencebetween.net/technology/difference-between-static-malware-analysis-and-dynamic-malware-analysis/.