Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____Н.В. Лукова-Чуйко
« » червня 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього рівня)

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

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: Механізми захисту веб-застосунків від кіберзагроз

Виконавець: студентка IV курсу, групи КБ-42

_____ **Кулішенко Софія Антонівна**
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Пархоменко І.І.	
Нормоконтроль	Зюбіна Р.В.	

Київ 2021

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____Н.В. Лукова-Чуйко
« » жовтня 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентці **КБ-42** **Кулішенко Софії Антонівні**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Механізми захисту веб-застосунків від кіберзагроз

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол № 2 від 08.10.2020 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Клієнт-серверна архітектура, структура веб-застосунків, основні загрози веб-застосункам та методи їх попередження, базова конфігурація LAMP-стеку

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Організація процесу взаємодії суб'єктів у веб-просторі, архітектура веб-застосунків, нормативно-правова база в сфері захисту веб-додатків та інформації в цілому, основні загрози веб-застосункам та методи захисту від них; конфігурація

потенційно захищеного LAMP-стеку

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Поєднання та використання механізмів захисту серверної частини веб-застосунку на основі LAMP-стеку

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 12 жовтня 2020 року.

Завдання видав _____ І.І.Пархоменко
(підпис) (ініціали, прізвище)

Завдання прийняла до виконання _____ С.А.Кулішенко
(підпис) (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1.	Уточнення постановки задачі	25.01.2021 - 27.01.2021	*Виконано*
2.	Аналіз літератури	28.01.2020 - 11.02.2021	*Виконано*
3.	Аналіз процесу взаємодії суб'єктів у веб-просторі	12.02.2021 - 15.02.2021	*Виконано*
4.	Огляд архітектури веб-додатків та нормативно-правової бази стосовно їх захисту	16.02.2021 - 04.03.2021	*Виконано*
5.	Огляд основних кіберзагроз серверній частині веб-застосунків	05.03.2021 - 21.03.2021	*Виконано*
6.	Дослідження механізмів захисту від розглянутих загроз	22.03.2021 - 10.05.2021	*Виконано*
7.	Виконання практичної реалізації захисту серверної частини веб-застосунку	11.05.2021 - 27.05.2021	*Виконано*
8	Оформлення пояснювальної записки та презентації	28.05.2021 - 08.06.2021	*Виконано*
9.	Підготовка до захисту	09.06.2021 - 21.06.2021	*Виконано*

Завдання видав _____ І.І.Пархоменко
(підпис) (ініціали, прізвище)

Завдання прийняла до виконання _____ С.А.Кулішенко
(підпис) (ініціали, прізвище)

Термін подання дипломної роботи до ЕК 08 червня 2021

РЕФЕРАТ

Дипломна робота складається зі вступу, основної частини, що містить 3 розділи, висновків і списку літератури та джерел. Загальний обсяг роботи – 80 сторінок. Робота включає зміст, вступ, 3 розділи дипломної роботи, висновки, список використаних джерел та містить 24 рисунки, 2 таблиці.

Мета роботи – створити потенційно захищену серверну частину веб-застосунку шляхом поєднання запропонованих у дипломній роботі механізмів захисту від кіберзагроз.

У роботі проаналізована існуюча література, що стосується клієнт-серверної взаємодії, архітектури веб-застосунків, а також основних загроз, що стосуються серверної частини веб-застосунку та механізмів захисту від них.

Практична цінність отриманих результатів полягає в поєднанні та застосуванні методів захисту серверної частини веб-застосунку від кіберзагроз.

Результати здійснених досліджень можуть бути використані у складі комплексної організації захисту веб-застосунку, створеного для різноманітних цілей.

Напрямки подальших досліджень можуть включати розширення запропонованого в дипломній роботі комплексу механізмів захисту веб-серверів та веб-застосунків.

Ключові слова: захист веб-серверу, веб-застосунок, кіберзагроза, несанкціонований доступ, кібератака.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

(D)DoS	–	(Distributed) Denial-of-Service
SSL	–	Secure Sockets Layer
TLS	–	Transport Layer Security
HTTP	–	Hypertext Transfer Protocol
TCP	–	Transmission Control Protocol
IP	–	Internet Protocol
URL	–	Uniform Resource Locator
SQL	–	Structured Query Language
HTML	–	HyperText Markup Language
XSS	–	Cross Site Scripting
SQL	–	Structured Query Language
СУБД	–	Система Управління Базами Даних
B2B	–	Business 2 (to) Business
CMS	–	Content Management Systems
IDS	–	Intrusion Detection System
IPS	–	Intrusion Prevention System
WAF	–	Web Application Firewall
CSR	–	Certificate Signing Request
CA	–	Certificate Authority

ЗМІСТ

ВСТУП

Актуальність даної дипломної роботи обумовлюється активним використанням веб-застосунків, що безперервно обслуговують користувачів мережі Інтернет, а тому неодмінно працюють з чутливою інформацією, такою як, наприклад, персональні дані, які надаються веб-сервісам користувачами та які обробляються у веб-середовищі, а тому мають бути надійно захищені від несанкціонованих дій сторонніх осіб, які потенційно можуть призвести до витоку, небажаної модифікації або втрати даних.

На даний момент неймовірно великі обсяги інформації циркулюють у веб-просторі. Сервери отримують, передають, зберігають та обробляють різноманітну інформацію. Дуже часто така інформація є конфіденційною і не має бути доступною стороннім особам. Але нерідко трапляються спроби порушити роботу або отримати несанкціонований доступ до чутливої інформації з боку веб-застосунків заради корисних цілей або розваг.

Дуже важливим аспектом функціонування веб-застосунків є їхня безпека, адже без належного рівня безпеки стає неможливою обробка конфіденційних даних, тому що в такому випадку в будь-який момент часу може бути порушена цілісність, доступність та конфіденційність такої інформації. Саме тому гостро стає питання захисту веб-застосунків від різноманітних кіберзагроз.

Безумовно, хакери є дуже винахідливими, тому на сьогоднішній день існує велика кількість кіберзагроз чутливим веб-ресурсам та веб-застосункам в цілому. В цій роботі розглянуті найчастіше реалізовані загрози та наведені рекомендації щодо методів захисту від них.

Складовими веб-застосунків є програмна частина, що розробляється власне відповідно до мети функціонування застосунку та включає в себе інформаційне наповнення, інтерфейс користувача та інше, та серверна частина, що обслуговує застосунок. До серверної частини можна віднести веб-сервер, на базі якого будується застосунок, та бази даних, призначені для обробки інформації, що

циркулює в цьому середовищі. Дана дипломна робота націлена на пошуки та подальше поєднання механізмів захисту веб-застосунку з боку його серверної частини для розширення області можливого практичного застосування отриманих результатів.

Тому *метою роботи* є створення потенційно захищену серверну частину веб-застосунку шляхом поєднання запропонованих у дипломній роботі механізмів захисту від кіберзагроз.

Для досягнення зазначеної мети поставлено *наступні завдання*:

- проаналізувати процес взаємодії суб'єктів у веб-просторі;
- оглянути архітектуру веб-додатків та нормативно-правову базу стосовно їх захисту;
- розглянути основні кіберзагрози веб-застосункам;
- дослідити механізми захисту від розглянутих загроз;
- виконати практичну реалізацію захисту серверної частини веб-застосунку.

Об'єктом дослідження є процес захисту серверної частини веб-застосунків від кіберзагроз.

Предметом дослідження є механізми захисту серверної частини веб-застосунків від кіберзагроз.

Методи дослідження – аналіз матеріалів та літературних джерел, пов'язаних з темою дипломної роботи, порівняння, системний підхід.

Робота була почата ще в 2018 році, тому апробація деяких частин дипломної роботи була проведена на V та VI науково-практичних конференціях "IT & I". Були написані наступні публікації:

1. Кулішенко С.А, Пархоменко І.І. «Загрози безпеці веб-ресурсів та способи захисту від них», 2018;
2. Кулішенко С.А, Пархоменко І.І. «Способи реалізації безпечної клієнт-серверної взаємодії у веб-просторі», 2019.

РОЗДІЛ 1
ВЗАЄМОДІЯ КЛІЄНТА ТА СЕРВЕРА У ВЕБ-ПРОСТОРІ

1.1 Огляд клієнт-серверної архітектури

1.1.1 Загальний опис архітектури клієнт-сервер

Клієнт-серверна модель обчислень - це розподілена структура додатків, яка розділяє завдання або робочі навантаження між провайдерами ресурсу або послуги, які називаються серверами, і запитувачами послуг, які називаються клієнтами. Часто клієнти та сервери спілкуються через комп'ютерну мережу на окремому обладнанні, але і клієнт, і сервер можуть знаходитись в одній системі. Серверна машина - це хост, на якому запущена одна або кілька серверних програм, які діляться своїми ресурсами з клієнтами. Клієнт не ділиться жодним із своїх ресурсів, але запитує вміст сервера або функцію служби. Тому клієнти ініціюють сеанси зв'язку з серверами, які очікують на вхідні запити. Характеристика клієнт-сервер описує взаємозв'язок програм, що співпрацюють у додатку. Серверний компонент надає функцію або послугу одному або багатьом клієнтам, які ініціюють запити на такі послуги. Такі функції, як обмін електронною поштою, доступ до Інтернету та доступ до баз даних, побудовані за моделлю клієнт-сервер. Користувачі, які отримують доступ до банківських послуг зі свого комп’ютера, використовують клієнт веб-браузера для надсилання запиту на веб-сервер банку. Ця програма може, в свою чергу, переслати запит до власної клієнтської програми бази даних, яка надсилає запит на сервер баз даних іншого комп'ютера банку для отримання інформації про рахунок. Баланс рахунку повертається клієнту бази даних банку, який, у свою чергу, подає його назад клієнту веб-браузера, відображаючи результати користувачеві. Клієнт-серверна модель стала однією з центральних ідей мережевих обчислень. Багато бізнес-додатків, що пишуться сьогодні, використовують модель клієнт-

сервер. Так само як і основні протоколи додатків Інтернету, такі як HTTP, SMTP, Telnet та DNS.

Важливо розуміти концепцію обчислень архітектури клієнт-сервера. Модель клієнт-сервер - це форма розподілених обчислень, коли одна програма (клієнт) спілкується з іншою програмою (сервером) з метою обміну інформацією.

Модель клієнт-сервер використовується для опису загальних послуг, що надаються в неоднорідному обчислювальному середовищі. Хоча можуть застосовуватися й інші моделі, моделі клієнт-сервер слугують також засобом опису послуг, описаних у цьому розділі. У моделі клієнт-сервер вузол в неоднорідному обчислювальному середовищі (наприклад, персональний комп'ютер, робоча станція, міні-комп'ютер, мейнфрейм) може характеризуватися як клієнт, сервер або клієнтський сервер. Клієнтський вузол не надає послуг жодному іншому вузлу. Клієнт може бути лише одержувачем послуг, що надаються іншими вузлами. Вузолом, що надає послуги, є сервер. Часто сервер також може бути клієнтом. Такий вузол називається клієнтським сервером. Термін клієнт може також позначати процес, який працює на клієнтському вузлі, або особу, від імені якої діє клієнтський процес. Подібним чином термін сервер може також позначати процес, який працює на вузлі сервера. У цій роботі термін клієнт відноситься до вузла клієнта, а термін сервер відноситься до вузла сервера [1].

Задачі клієнта, як правило, полягають в:

1. обробці інтерфейсу користувача;
2. перекладі запиту користувача у потрібний протокол;
3. надсиланні запит на сервер;
4. очікуванні відповіді сервера;
5. перетворенні відповідь на результати, що читаються людиною;
6. презентації результати користувачеві;

Функції сервера включають:

1. отримання запиту клієнта;
2. обробку цього запиту;
3. повернення результатів назад клієнту.

Типова взаємодія клієнт-сервер виглядає так:

1. користувач запускає клієнтське програмне забезпечення для створення запиту;

2. клієнт підключається до сервера;

3. клієнт надсилає запит на сервер;

4. сервер аналізує запит;

5. сервер обчислює результати запиту;

6. сервер надсилає результати клієнту;

7. клієнт представляє результати користувачеві.

На рис.1.1 клієнтами є персональний комп'ютер та робоча станція. Вони не надають жодних служб будь-якому іншому вузлу, але вони можуть бути одержувачем послуги, наприклад, монтування файлової системи або каталогу з сервера чи клієнтського сервера. Клієнтський сервер також може бути одержувачем послуг.

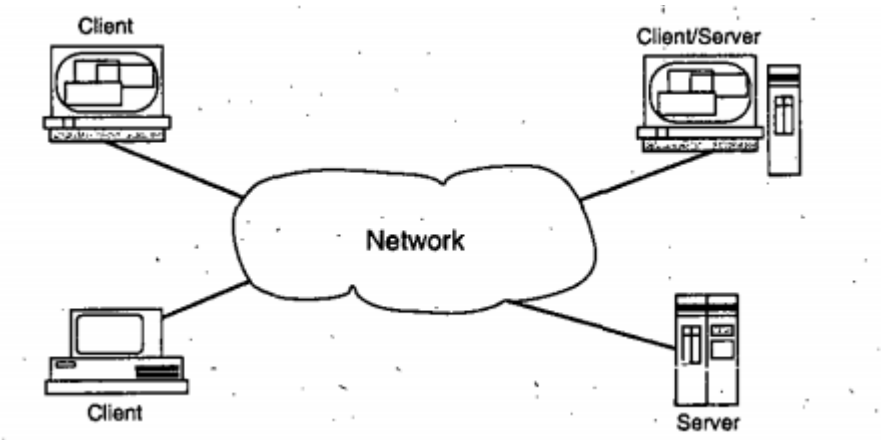

Рисунок 1.1 – Схема комунікацій клієнтів та серверів

Розподілені обчислення - це термін, що використовується для впровадження технологій у різнорідні середовища. Для операційних систем різнорідні обчислення означають можливість спілкування з іншими операційними системами та протоколами. Розподілені обчислення - це складна архітектура. Це передбачає переархітектуру програм, переробку систем та підвищення ефективності підтримки

мережі в цілому. Багато розподілених вузлів працюють від імені одного запитуючого клієнта. Це робить систему стійкою до відмов і децентралізованою, що є очевидною перевагою перед централізованими системами. Щоб технологія стала ефективною та революційною, розробники розподілених додатків повинні зробити все можливе, щоб мінімізувати складність розробки та обслуговування та інтегрувати своє програмне забезпечення з різними застосунками.

Проектування клієнт-серверних додатків вимагає модуляризації програм та їх функцій на окремі компоненти. Ці компоненти повинні бути обмежені лише інкапсульованими даними та функціями, які можуть переміщуватися між системами. Ця модель дизайну надає програмному забезпеченню клієнт-сервер більше адаптивності та гнучкості.

Модульний характер системи клієнт-сервер - сервер може бути замінений без негативного впливу на решту системи. Наприклад, можна оновити сервер до більш потужного комп'ютера без видимих змін для кінцевого користувача. Ця здатність змінювати компонентну систему робить системи клієнтського сервера особливо сприйнятливими до нових технологій як в апаратному, так і в програмному забезпеченні.

1.1.2 Переваги та недоліки архітектури клієнт-сервер

Переваги:

Економічна: з вищесказаного стає зрозуміло, що кожен компонент, а саме клієнт і сервер у клієнтських серверних обчисленнях, виступає як окремий об'єкт, який може бути з'єднаний мережею. Однією з переваг такого типу дизайну є можливість вибору системи відповідно до потреб кожного компонента обчислювального середовища клієнтського сервера. Іншими словами, системна потреба системного компонента в обчисленні клієнтського сервера відрізняється від потреби клієнтського компонента. Ми знаємо, що сервер в клієнт-серверних обчисленнях управляє даними в базі даних. Отже, для управління базою даних система серверних компонентів у клієнт-серверних обчисленнях повинна бути

системою з більшою пам’яттю, а також високим процесором. швидкість. Потрібно вибрати систему для сервера в клієнт-серверних обчисленнях, яка задовольняє зазначені вище потреби. З іншого боку, система для клієнтського компонента в обчисленні клієнтського сервера не повинна мати зазначену вище конструкцію. Тобто система, що представляє клієнтський компонент в обчислювальних роботах клієнтського сервера, не повинна мати більше пам'яті або високу швидкість процесора. Але оскільки клієнт представляє користувальницький інтерфейс або інтерфейсну програму, система повинна мати можливість підтримувати графічні утиліти або додатки, а також мати або підтримувати вторинне сховище відповідно до потреб. Отже, вибираючи систему для клієнтського компонента програми клієнтського сервера, потрібно врахувати цей момент і зробити вибір. Отже, з наведеного вище пояснення стає зрозуміло, що обрану систему для сервера можна зробити окремо, а для клієнта окремо відповідно до потреб. Це допомагає організації зробити економічно вигідну систему шляхом придбання систем відповідно до потреб.

Підвищена продуктивність: продуктивність також збільшується без особливих зусиль. Наприклад, якщо організація вирішить вибрати систему на стороні сервера для підвищення продуктивності, це можна зробити, не впливаючи на користувальницьку програму, оскільки в цьому випадку не потрібно порушувати клієнтський компонент обчислень клієнт-сервера. Простота в роботі та обслуговуванні: Оскільки компонент сервера та клієнта, а саме користувальницький інтерфейс та управління базами даних, розділені в конструкції обчислювального сервера клієнта, це полегшує зусилля адміністратора бази даних, а також розробників. Це пояснюється тим, що розробники можуть сконцентруватися лише на програмі користувача. Наприклад, наприклад, якщо відбувається зміна користувальницького інтерфейсу, систему можна змінити без особливих зусиль, вносячи зміни лише на стороні клієнта, не вносячи змін до компонентів сервера [2].

Гнучка розробка користувальницького інтерфейсу є найбільш очевидною перевагою обчислень клієнт-сервера. Можна створити інтерфейс, який не залежить від сервера, що розміщує дані. Таким чином, інтерфейс користувача програми

клієнт-сервер може бути записаний на базі DOS або UNIX, а сервер може бути записаний на мейнфреймі. Це дозволяє зберігати інформацію на центральному сервері та поширювати її на різні типи віддалених комп’ютерів. Оскільки за користувальницький інтерфейс відповідає клієнт, сервер має більше обчислювальних ресурсів, які потрібно витратити на аналіз запитів та розповсюдження інформації. Це ще одна велика перевага обчислень клієнт-сервер; він, як правило, використовує сильні сторони різноманітних обчислювальних застосунків для створення більш потужних додатків. Незважаючи на те, що його обчислювальні можливості та можливості зберігання даних перевершують можливості мейнфрейму, немає жодної причини, чому Macintosh не міг бути використаний як сервер для менш вимогливих програм. Коротше кажучи, обчислення клієнт-сервер надає механізм співпраці різних комп'ютерів над одним обчислювальним завданням.

Недоліки

Незважаючи на те, що переваг безліч, є деякі недоліки і в обчисленні клієнтського сервера, завдяки чому інші технології, такі як мейнфрейми, набувають своєї форми та популярності.

Ми вже бачили вище, що обчислення клієнтського сервера є економічно вигідними. Це правда, як було сказано вище, а також тому, що в порівнянні з мейнфреймами вартість апаратного забезпечення дуже менша в обчисленнях клієнтського сервера. Але в клієнт-серверних обчисленнях присутнє не лише апаратне забезпечення, але є й інші витрати, такі як вартість обслуговування кількох клієнтів, що включає користувачів, розробників, адміністраторів для обслуговування бази даних на стороні сервера тощо. Коли ми враховуємо все це, вартість обчислень у моделі клієнт-сервер вища. Ми знаємо, що мейнфрейм - це централізована система управління базами даних. Хоча система є дорожчою з апаратного погляду, вона забезпечує надійну та захищену систему, яка дуже потрібна для великих додатків у реальному часі. Це пояснюється тим, що в ці реальні часи величезних додатків основною вимогою є те, щоб система не повинна зупинятися і повинна працювати безвідмовно, що досягається централізованими

системами мейнфреймів. Але в клієнт-серверних обчисленнях це не підтримується централізовано, а надійність не дуже велика, що означає, що якщо система зупиниться між ними, тоді величезна програма в реальному часі зіткнеться з величезними втратами в плані продуктивності та вартості [3].

1.2 Загальний огляд архітектури веб-застосунків

Пересічний користувач Інтернету бачить певну сторінку у своїй системі за допомогою низки взаємодій між різними компонентами програм, користувальницькими інтерфейсами, системами проміжного програмного забезпечення, базами даних, сервером та браузером.

Фреймворком, який пов’язує це відношення та взаємодію, є архітектура веб-додатків. У двох словах, потік процесів, як правило, включає перегляд користувачем URL-адреси, після чого браузер запускає пошук.

В результаті пошуку мережа надсилає дані в браузер із сервера, і браузер відображає запитану сторінку.Простіше кажучи, архітектура веб-додатків включає різні компоненти та зовнішні програми.

Крім того, додатки стають все більш складними, і розробники, яким доручено створити додаток, все частіше схиляються до архітектури розробки повного стеку.

Робота веб-додатків

Веб-програми включають два різні набори програм, які працюють окремо, але одночасно із спільною метою гармонійної роботи з надання рішень.

Як правило, два набори програм включають код у браузері, який працює відповідно до входів користувача, та код на сервері, який працює відповідно до запитів протоколів, HTTPS.

Іншими словами, веб-розробники повинні мати можливість вирішувати функції коду на сервері та функції коду в браузері, а також те, як вони функціонуватимуть по відношенню один до одного.

Існує безліч типів веб-застосунків, такі як наприклад:

Статичні веб-програми

Це найосновніший тип веб-додатків, у якому мало вмісту чи місця для пересування. Ці веб-програми часто створюються з використанням CSS та HTML і можуть обробляти анімований вміст, такий як GIFS та відео. Змінення вмісту, включеного в статичні веб-програми, може бути складним, і для внесення серйозних змін потрібно завантажити HTML-код, а також змінити та завантажити його на сервер. Крім того, якщо додаток розробляли не ви, ані ваша компанія, вам, ймовірно, доведеться шукати особу, яка планувала та розробляла статичний веб-додаток.

Динамічні веб-програми

Цей веб-додаток у стилі є більш технічно складним у порівнянні з вищезазначеним статичним додатком. У випадку з динамічними веб-додатками вони містять бази даних або форуми з постійною можливістю оновлювати або змінювати наявну інформацію. Зазвичай це відбувається в результаті CMS або системи управління вмістом.

CMS, як правило, супроводжує динамічну веб-програму, щоб адміністратори або користувачі могли легко оновлювати або редагувати вміст, який включений. Багато різних веб-мов можна використовувати з динамічними веб-програмами. Однак PHP та ASP є найпоширенішими, оскільки їх найпростіше зрозуміти, коли йдеться про структурування вмісту.

Редагувати вміст простіше за допомогою динамічних веб-додатків, а оновлення самого вмісту може бути досить простим. Однак серверна частина або частина програмування може бути більш складною залежно від сервера та інших факторів. З огляду на це, елементи дизайну можна легко модифікувати відповідно до ваших особистих переваг.

Поширений приклад статичних веб-програм включає онлайн-портфоліо або цифрові навчальні програми. У тому ж ключі, ви можете уявити цільову сторінку як статичну веб-програму з відображеною контактною та відносною інформацією.

Додатки для електронної комерції

Якщо ви коли-небудь купували щось через Інтернет, ви, мабуть, знайомі з одним із найпоширеніших веб-додатків, а саме електронною комерцією. Якщо веб-

додаток має форму магазину чи магазину, цей тип розробки класифікується як електронна комерція. Цей процес розробки додатків є більш складним, ніж два вищезазначені, оскільки вимагає способу збору електронних платежів. Замовлення, платежі, а також опублікування та видалення елементів повинні підтримуватися за допомогою панелі управління, кодованої на веб-сайті розробником. Крім того, у багатьох випадках потрібно буде розглянути мобільний додаток еквівалентності.

Веб-програма порталу

Веб-програма порталу просто відноситься до програми, в якій багато різних розділів або категорій доступні через домашню сторінку. Сторінки цього типу можуть варіюватися від форумів, чатів, електронних листів, сторінки браузера тощо.

Анімований веб-додаток

Щоб анімовані веб-програми мали бути сумісними з технологією FLASH. Це може бути складним для тих, хто не має досвіду роботи з цими технологіями. Однак, якщо ви шукаєте веб-додаток з досить сучасними можливостями або який-небудь рух, тоді робота з FLASH є важливою.

На жаль, при використанні цієї технології є кілька недоліків, оскільки вона не буде працювати як з SEO-оптимізацією, так і з позиціонуванням в Інтернеті, оскільки Google та інші пошукові системи не можуть правильно прочитати її інформацію. Отже, якщо ви тільки починаєте працювати з веб-сайтом свого бізнесу або розробляєте цільову сторінку, може бути важливо почекати наступні етапи.

Системи управління контентом

По суті, це означає, що інтерфейс, до якого можна отримати доступ та оновити, і система управління вмістом необхідна, якщо ви шукаєте, щоб ваш веб-додаток постійно розроблявся. Ці системи дуже популярні для особистих блогів, корпоративних блогів, джерел засобів масової інформації тощо. Існує кілька загальних систем управління вмістом, і це:

- WordPress: Ця платформа для ведення блогів користується широкою популярністю та довірою як серед людей, так і серед професіоналів, і це, мабуть, завдяки інтуїтивно зрозумілому інтерфейсу та обсягу доступних онлайн-посібників.

- Joomla: ця платформа популярна і виходить наступною на черзі WordPress як система управління вмістом з відкритим кодом для управління веб-сайтами та блогами. Оскільки цей інтерфейс не такий популярний, як WordPress, і тому не має такої великої спільноти. Однак інтерфейс все ще настільки ж інтуїтивно зрозумілий [5].

Архітектура веб-додатків: компоненти

Архітектура веб-додатків складається з різних компонентів, які розділені на дві категорії компонентів - компоненти програми інтерфейсу користувача та структурні компоненти.

Компоненти програми інтерфейсу користувача

Це посилання на веб-сторінки, які виконують роль, що стосується дисплея, налаштувань та конфігурацій.

Це пов'язано з інтерфейсом / досвідом, а не з розробкою, і, отже, воно має справу з інформаційними панелями дисплея, налаштуваннями конфігурації, сповіщеннями та журналами тощо.

Структурні компоненти

Структурні компоненти веб-програми в основному відносяться до функціональності веб-програми, з якою взаємодіє користувач, управління та зберігання бази даних.

Іншими словами, це має більше спільного із структурними аспектами архітектури, як і випливає з назви, та, в основному, включає (1) веб-браузер або клієнт, (2) сервер веб-додатків та (3) сервер баз даних.

Веб-браузер або клієнт дозволяє користувачам взаємодіяти з функціями веб-програм і, як правило, розробляється з використанням HTML, CSS та JavaScript.

Сервер веб-додатків обробляє центральний хаб, що підтримує бізнес-логіку та багаторівневі програми, і, як правило, розробляється за допомогою Python, PHP, Java, .NET, Ruby та Node.js.

Сервер баз даних пропонує бізнес-логіку та відповідну інформацію / дані, які зберігаються та керуються сервером веб-додатків. Він зберігає, отримує та надає інформацію.

Архітектура веб-сервера

За всіма визначеннями, це стосується ідеального розміщення веб-сервера, що полегшить проектування, розробку та розгортання веб-сервера.

Роль полягає у задоволенні запитів клієнтів, включаючи браузери та мобільні програми через захищені протоколи. Запити можуть стосуватися ресурсів сторінки, а також можуть бути пов'язані з REST API.

Веб-сервери невід'ємні для роботи веб-програм, що вимагає посилення акценту на архітектурі веб-серверів, включаючи його фізичну ємність - зберігання, пам'ять, обчислювальну потужність та продуктивність, крім рівнів додатків.

Це може бути де завгодно - або всередині сервера, через мережу або операційні системи. До різних типів архітектури веб-серверів належать:

Архітектура веб-додатків Java

Завдяки тому, що Java є універсальною мовою програмування, вона популярна в середовищі розвитку підприємства.

Вимоги рішення визначають ступінь / складність архітектури веб-додатків - наприклад, рішення можуть бути як простими, так і багаторівневими програмами.

Незалежно від складності та характеру програми, архітектура веб-додатків Java є найкращою платформою для розробників для створення рішень та реалізації відповідно до очікувань.

Однією з виразних переваг цієї архітектури є можливість поєднувати та покладатись на власні інструменти Java та основи для створення додатків, що розширюють весь спектр - від простих до найскладніших програм.

Хмарна архітектура веб-додатків

Перехід до хмари є скоріше імперативом, ніж вибором, головним чином в результаті переваг за всіма параметрами.

Отже, була розроблена архітектура веб-додатків, заснована на хмарі, що призвело до створення наслідків - роз'єднання даних. Іншими словами, хмарні додатки функціонують та зберігають інформацію на локальних серверах та хмарі.

Архітектура веб-додатків Node.js

В основі архітектури веб-додатків Node.js лежить шаблон представлення моделі; наприклад, модель-view-контролер, модель-view-view модель і модель-view-презентатор.

Node.js дозволяє створювати шаблони з метою ідентифікації елементів коду, а також для конфігурації елементів, крім маршрутизації.

Це спирається на взаємозв'язок сутності, що допомагає забезпечити безперебійну роботу програми через систематизацію даних, розбиття логіки на модулі, обробку цінних статистичних даних із журналів та розподіл коду.

Ще однією важливою перевагою є той факт, що архітектура веб-додатків Node.js допомагає створювати масштабовані веб-програми.

Архітектура веб-додатків .NET

Це обробляє вимоги, включаючи підтримку між платформами, контейнери Docker, мікросервіси та паралельне керування версіями.

Основним моментом цього фреймворку є можливість зберігати дані без необхідності застосовувати код бази даних.

Відомий як рівень доступу до даних, він допомагає покращити функціональність та розвиток, оскільки для оптимізації архітектура покладається на ASP.NET Core та .NET Core.

Архітектура веб-додатків PHP

В силу того, що PHP є найменш складною та високофункціональною мовою розвитку, PHP є однією з найпопулярніших серед спільноти.

Архітектура дозволяє забезпечити надійну безпеку, швидку розробку, спеціальну структуру, просте обслуговування та розширену підтримку спільноти розробників.

Архітектура веб-додатків AngularJS

Ця архітектура працює в подвійному режимі для HTML і TypeScript як платформа, а також як фреймворк.

Використання NgModules для побудови пропонує безліч переваг завдяки розробці Angular, включаючи досвід користувачів із ледачим завантаженням, крім зменшення розміру коду.

Веб-розробка Laravel

Laravel, інший веб-фреймворк PHP, приймає архітектурний шаблон контролера подання моделі та має в основі виразний, креативний та елегантний синтаксис.

Він пропонує спрощений процес веб-розробки завдяки кращій маршрутизації, сеансам, аутентифікації та кешуванню; як результат, для завершення проектів потрібно менше часу.

Це сприяє збільшенню трафіку та включає гнучкі функції, включаючи модульні системи упаковки.

Одним із основних моментів є той факт, що різні маршрути можна створювати з наявною назвою маршруту, створюючи унікальні URL-адреси. Автозавантажувальне обладнання відмовляється від потреби у спеціальних шляхах включення.

Веб-фреймворки Python

Python полегшує розробку веб-додатків із коротким / стислим кодом, легко читабельним та ремонтопридатним. Це одна з причин, чому розробники захоплюються Python для використання в якості мови сценаріїв на стороні сервера.

Це пришвидшує веб-додаток, тому що відмовляється від необхідності обслуговування веб-додатків.

Спеціальна веб-програма завдяки розробці Python досягається використанням декількох веб-фреймворків Python, включаючи повний стек.

Мало того, що ця мова динамічна, вона не вимагає тривалого кодування, як правило, зменшуючи довжину кодування на 1/5 інших мов програмування.

Мабуть, найбільшою перевагою Python є те, що мова вважається найбільш підходящою для прототипування.

На додаток до цього, Python дуже гнучкий і добре інтегрується з іншими мовами, крім великої бібліотеки, що знаходиться в його розпорядженні [6].

1.3 Нормативно-правова база із захисту веб-застосінків та безпеки інформації

ДСТУ ISO/IEC 27001

«Інформаційні технології. Методи захисту системи управління інформаційною безпекою. Вимоги»

ISO / IEC 27001 - це міжнародний стандарт щодо управління інформаційною безпекою. У ньому детально викладено вимоги щодо створення, впровадження, підтримки та постійного вдосконалення системи управління інформаційною безпекою (СУІБ) - метою якої є допомогти організаціям зробити інформаційні активи, якими вони володіють, більш захищеними [7].

ДСТУ ISO/IEC 27005

«Інформаційні технології. Методи захисту. Управління ризиками інформаційної безпеки»

ISO / IEC 27005 надає керівні принципи для встановлення системного підходу до управління ризиками інформаційної безпеки, який необхідний для виявлення організаційних потреб щодо вимог інформаційної безпеки та створення ефективної системи управління інформаційною безпекою. Більше того, цей міжнародний стандарт підтримує концепції ISO / IEC 27001 і призначений для сприяння ефективному впровадженню інформаційної безпеки на основі підходу до управління ризиками [8].

ДСТУ ISO/IEC 27002

«Інформаційні технології. Методи захисту. Звід практик щодо заходів інформаційної безпеки»

ISO 27002 надає сотні потенційних засобів контролю та механізмів контролю, які розроблені для впровадження з керівництвом, передбаченим ISO 27001. Запропоновані засоби контролю, перелічені у стандарті, призначені для вирішення

конкретних проблем, виявлених під час офіційної оцінки ризику . Стандарт також покликаний надати керівництво з розробки стандартів безпеки та ефективної практики управління безпекою [9].

ДСТУ ISO/IEC TS 27034

«Інформаційні технології. Захист застосунків»

ISO / IEC 27034 пропонує вказівки щодо інформаційної безпеки тим, хто визначає, розробляє та програмує, або забезпечує, впроваджує та використовує прикладні системи, іншими словами менеджерів бізнесу та IT, розробників та аудиторів та, зрештою, кінцевих користувачів ІКТ. Мета полягає в тому, щоб забезпечити, щоб комп'ютерні програми забезпечували бажаний або необхідний рівень безпеки на підтримку системи управління інформаційною безпекою організації, адекватно вирішуючи багато ризиків безпеки ІКТ [10].

Висновки за розділом 1

В першому розділі дипломної роботи було оглянуто клієнт-серверну архітектуру, що розділяє взаємодіючі системи на клієнтську та серверну частини. Перша запитує та використовую послуги, а друга їх надає та здійснює керування ними. Додатково було окреслено переваги та недоліки такої взаємодії. Було стисло описано і деякі архітектури серверів веб-застосунків, такі як Python, Java, .NET та інші. Типи веб-застосунків, такі як, наприклад, статичні, динамічні веб-застосунки, розглядалися в першому розділі. Крім цього, було перераховано та описано зміст деяких нормативно-правових документів, що стосуються веб-додатків та захисту інформації в цілому.

РОЗДІЛ 2
ОСНОВНІ ЗАГРОЗИ СЕРВЕРНІЙ ЧАСТИНІ ВЕБ-ЗАСТОСУНКІВ ТА МЕТОДИ ЇХ ПОПЕРЕДЖЕННЯ

2.1 Атаки у мережі клієнт-сервер

Будь-які комунікації, які відбуваються через Інтернет, потребують повної ізоляції одне від одного. Це складає основу безпечного спілкування. Посилення безпеки системи вимагає великого ступеню оцінки та тестування мережі, інфраструктур тощо для цієї конкретної системи, і тоді результати можуть бути використані як вхідні дані для розробки ефективної методології безпеки інфраструктури. Безпека зв'язку через мережу залежить від його ефекту та важливості інформації, що надсилається по мережі.

Як правило, існують різні атаки, які можуть вплинути на конкретний комп'ютер, систему або навіть інфраструктуру. На мою думку, атаки можна класифікувати загалом за такими категоріями:

- атаки на мережу;
- атаки на систему;
- атаки соціальної інженерії.

Атаки на мережу: Атаки на мережу додатково поділяються на наступні категорії:

- атаки на протоколи;
- атаки на програми;
- атаки "людина в середині".

Атаки на протоколи:

Атаки на протоколи є дуже поширеним явищем, і воно експоненціально зростає з кожним днем. Існують різні рівні, в яких можуть проводитися атаки; найпростішим буде прикладний рівень. Оскільки більшість протоколів розроблені

для роботи від архітектури зверху вниз архітектури з відкритим вихідним кодом або загальновідомої моделі OSI.

Поширені атаки:

- Сканування хостів: Проведіть пінг на хост-машині за допомогою широкомовних пакетів ICMP, відповідь, отримана в пакетах ICMP, потім використовується для визначення кількості користувачів у мережі та IP-адрес хост-машини. Це дає зловмисникові широку картину мережевих хостів, а потім дозволяє зловмиснику звузитись щодо того, які його наміри щодо цієї конкретної мережі.
- Сканування портів: За допомогою простого сканера портів зловмисник може сканувати мережу на наявність будь-яких відкритих портів і досліджувати подальші відкриті порти, щоб отримати доступ до мережі, отримавши таким чином доступ до хост-машин. У цій атаці більшість мереж матимуть деякі часто використовувані послуги та їх номери портів, такі як порт 80 для HTTP, - 8 - порт 22 для SSH; порт 21 для FTP, порт 23 для Telnet тощо стане вразливим для цих атак.

Зворотна підробка DNS та ARP: При зворотній підробці DNS зловмисник може змінити служби доменних імен, щоб змінити шлях пакетів до атакуючої машини з початкового місця призначення. У ARP-підробці MAC-адреса відіграє вирішальну роль для прослуховування пакетів, в деяких випадках можливе навіть перенаправлення пакетів (атаки MITM).

Відмова та розподілена відмова в обслуговуванні. У разі відмови в обслуговуванні сервер заповнюється кількома запитами, таким чином, що перестає відповідати через певний проміжок часу, тим самим відмовляючи в обслуговуванні будь-яким законним користувачам, які мали бути підключеними до цього сервера. Цей тип атаки проводиться з однієї машини зловмисника. При розподіленій відмові в службі зловмисник компрометує (у більшості випадків) або об’єднується разом із кількома хостами, щоб заповнити сервер пакетами, а отже, збільшує трафік, таким чином і досягається відмова в обслуговуванні. Різниця між DOS і DDOS полягає в тому, що в DDOS атака здійснюється різними машинами в різних місцях [11].

Атаки на програми: Атаки на мережеві програми також зростають, загальною причиною цього є те, що більша частина мережевих додатків також побудована на протоколі TCP. Вторгнення відбувається на рівні програми, наприклад, для компрометації сеансів, сценаріїв між сайтами тощо.

Атаки "людина посередині": зловмисник, використовуючи сніфер, підслуховує пакети в мережі і використовує цю інформацію для подальшої атаки.

Атаки на систему: до цільової категорії даної атаки належить будь-яка хост-машина, яку зламує зловмисник. Ця атака не обов'язково виникає внаслідок мережевої атаки. Атака може бути зловмисним кодом, прихованим в електронній пошті, зловмисником, який контролює хост-систему шляхом встановлення бекдора, імплементації троянів, хробаків у систему, а також створення вірусної програми та спокушення користувача завантажити файл, що містить шкідливі віруси, які розмножуватимуться в системі.

Атаки соціальної інженерії: це дуже важка атака, щоб досягти успіху, оскільки саме користувачеві належить контроль над інформацією, яку він / вона може ненавмисно передати зловмиснику. Соціальна інженерія - це метод, де зловмисник, наприклад, залучає користувачів на певний веб-сайт і намагається отримати якомога більше інформації для того, щоб визначити, який тип мережі, тип систем тощо [12].

2.2 Атаки відмови в обслуговуванні

Великою перешкодою для подолання явища DDoS є те, що воно неоднорідне і охоплює різноманітні тактики. Для початку є три всеохоплюючі категорії цих атак, що становлять основу цієї екосистеми:

Мережева відмова в обслуговуванні

Наприклад так звана "атака споживання смуги пропускання": зловмисник намагатиметься використати всю доступну пропускну здатність мережі ("затоплення") таким чином, що законний трафік більше не зможе переходити до / з цільових систем. Крім того, зловмисники можуть використовувати "розподілене відображення відмови в обслуговуванні" (DRDoS), щоб змусити інші, несвідомі

системи, щоб допомогти в атаці, заливаючи ціль мережевим трафіком. Під час цієї атаки законним користувачам та системам забороняється доступ, який вони зазвичай мають до інших систем атакованої мережі. Варіант цієї атаки з подібними результатами передбачає зміну (або збиття) самої мережі шляхом націлювання на пристрої мережевої інфраструктури (наприклад, комутатори, маршрутизатори, бездротові точки доступу тощо) таким чином, що вони більше не дозволяють мережевому трафіку надходити до / як правило, від цільових систем, що призводить до подібних результатів відмови в обслуговуванні без необхідності повені.

Відмова в обслуговуванні, орієнтована на систему

Ці атаки спрямовані на підрив зручності використання цільових систем. Вичерпання ресурсів є загальним вектором атаки, коли зловмисник навмисно «витрачає» обмежені системні ресурси (наприклад, пам'ять, центральний процесор, дисковий простір), щоб скалічити звичайні дії цілі. Наприклад, затоплення SYN - це цілеспрямована на систему атака, яка використовуватиме всі доступні вхідні мережеві підключення до цілі, перешкоджаючи законним користувачам та системам створювати нові мережеві підключення. Результати атаки, орієнтованої на систему, можуть варіюватися від незначного збою або уповільнення до прямого збою системи. Хоча це не часто, атака постійної відмови в обслуговуванні (PDoS) може навіть пошкодити ціль до такої міри, що її потрібно буде фізично відремонтувати або замінити.

Відмова в обслуговуванні, орієнтована на додатки

Орієнтація на додаток є популярним вектором атак DoS. Деякі з цих атак використовують існуючу звичну поведінку програми, щоб створити ситуацію відмови в обслуговуванні. Приклади цього включають блокування користувачів із їхніх облікових записів або надсилання запитів, які підкреслюють невід'ємний компонент програми (наприклад, центральну базу даних) до такої міри, що інші користувачі не можуть отримати доступ або використовувати програму за призначенням чи очікуванням. Інші атаки, орієнтовані на додатки, покладаються на уразливості програми, такі як активація стану помилки, що викликає збій програми,

або використання експлойту, що полегшує прямий доступ до системи для подальшого посилення атаки DoS.

DoS-атаки різняться методами їхньої реалізації, саме тому на сьогоднішній день існує велика кількість видів таких атак. Нижче стисло розглянуто декілька доволі часто використовуваних хакерами різновидів атак відмови в обслуговуванні:

1. SYN Flood

Ця атака використовує тристороннє рукостискання TCP, техніку, яка використовується для встановлення будь-якого зв'язку між клієнтом, хостом і сервером за допомогою протоколу TCP. Зазвичай клієнт подає на сервер повідомлення SYN (синхронізувати) з проханням встановити з'єднання.

Коли триває атака SYN Flood, злочинці надсилають безліч цих повідомлень з підробленої IP-адреси. В результаті приймаючий сервер стає нездатним обробляти та зберігати стільки пакетів SYN і відмовляє в обслуговуванні реальним клієнтам.

2. LAND атака

Щоб здійснити атаку заперечення локальної мережі (LAND), суб’єкт загрози надсилає сфабриковане повідомлення SYN, в якому вихідні та кінцеві IP-адреси однакові. Коли сервер намагається відповісти на це повідомлення, він потрапляє в цикл, періодично генеруючи відповіді собі. Це призводить до сценарію помилки, і цільовий хост може в підсумку вийти з ладу.

3. SYN-ACK Flood

Логіка цього вектора атаки полягає у зловживанні стадією зв'язку TCP, коли сервер генерує пакет SYN-ACK для підтвердження запиту клієнта. Щоб виконати цей натиск, шахраї затоплюють ресурси центрального процесора та оперативної пам'яті сервера безліччю неправдивих пакетів SYN-ACK.

4. ACK & PUSH ACK Flood

Як тільки тристороннє рукостискання TCP призведе до встановлення зв'язку між хостом і клієнтом, пакети ACK або PUSH ACK надсилаються вперед і назад, доки сеанс не буде припинено. Сервер, націлений на цей тип DDoS-атаки, не може ідентифікувати походження сфальсифікованих пакетів і витрачає всю свою обробну потужність, намагаючись визначити, як з ними оброблятись.

5. Фрагментований ACK Flood

Ця атака є початком вищезгаданої техніки ACK & PUSH ACK Flood. Це зводиться до поглинання цільової мережі порівняно невеликою кількістю фрагментованих пакетів ACK, які мають максимально допустимий розмір, зазвичай 1500 байт кожен. У мережевому обладнанні, такому як маршрутизатори, закінчуються ресурси, які намагаються зібрати ці пакети. Крім того, фрагментовані пакети можуть ковзати нижче радарів систем запобігання вторгненню (IPS) та брандмауерів.

6. Підроблений сесійний Flood (фальшива атака сесії)

Для того, щоб обійти засоби захисту мережі, кіберзлочинці можуть ефективніше підробляти сеанс TCP, подаючи підроблений пакет SYN, серію пакетів ACK та принаймні один пакет RST (скидання) або FIN (припинення з'єднання). Ця тактика дозволяє шахраям обійти захист, який лише контролює вхідний трафік, а не аналізує зворотний трафік [13].

7. UDP Flood

Як випливає з назви, ця DDoS-атака використовує декілька пакетів UDP (User Datagram Protocol). Для запису, UDP-з'єднанням бракує механізму рукостискання (на відміну від TCP), і тому можливості перевірки IP-адреси дуже обмежені. Коли ця експлуатація в розпалі, обсяг фіктивних пакетів перевищує максимальну здатність цільового сервера для обробки та відповіді на запити.

8. DNS Flood

Це варіант UDP Flood, який спеціально розміщується на серверах DNS. Зловмисник генерує безліч фальшивих пакетів запитів DNS, схожих на законні, які, здається, походять з величезної кількості різних IP-адрес. DNS Flood - один із найскладніших рейдів відмови в обслуговуванні для запобігання та відновлення.

9. NTP Flood (NTP Amplification)

Мережевий протокол часу (NTP), один з найстаріших мережевих протоколів, на який покладена синхронізація синхронізації між електронними системами, лежить в основі іншого вектора атаки DDoS. Ідея полягає у використанні

загальнодоступних серверів NTP для перевантаження цільової мережі великою кількістю пакетів UDP.

10. HTTP Flood

Виконуючи DDoS-атаку HTTP Flood, противник надсилає нібито законні запити GET або POST на сервер або веб-програму, витягуючи більшість або всі свої ресурси. Ця техніка часто включає ботнети, що складаються з "зомбі" комп'ютерів, раніше заражених шкідливим програмним забезпеченням.

11. Рекурсивний HTTP GET Flood

Щоб здійснити цю атаку, зловмисний суб’єкт вимагає від сервера масив веб-сторінок, перевіряє відповіді та ітеративно просить кожен елемент веб-сайту вичерпати ресурси сервера. Експлуатація виглядає як низка законних запитів і може бути важко визначити.

12. ICMP Flood

Також називається Ping Flood, це вторгнення має на меті затопити сервер або інший мережевий пристрій численними підробленими запитами exo-сигналів або пінгами протоколу ICMP. Отримавши певну кількість пінгів ICMP, мережа відповідає такою ж кількістю пакетів відповідей. Оскільки ця можливість відповідати обмежена, мережа досягає свого порогу продуктивності і стає невідповідною.

13. Атака неправильно використаної програми

Замість використання підроблених IP-адрес, ця атака паразитує на законних клієнтських комп'ютерах, на яких запущені ресурсомісткі програми, такі як інструменти P2P. Шахраї перенаправляють трафік від цих клієнтів на сервер жертви, щоб збити його через надмірне навантаження обробки. Цю техніку DDoS важко запобігти, оскільки трафік виникає на реальних машинах, раніше скомпрометованих зловмисниками.

14. IP Null Attack

Цей здійснюється шляхом надсилання безлічі пакетів, що містять недійсні заголовки IPv4, які повинні нести деталі протоколу транспортного рівня. Фокус у тому, що суб’єкти загроз встановлюють для цього значення заголовка значення null.

Деякі сервери не можуть належним чином обробити ці корумповані пакети і витрачати свої ресурси, намагаючись з’ясувати, як з ними поводитися.

15. Пінг атаки

Щоб запустити цей рейд, хакери отруюють мережу жертв нетрадиційними пінг-пакетами, розмір яких значно перевищує максимально допустиме значення (64 байта). Ця невідповідність призводить до того, що комп’ютерна система виділяє занадто багато ресурсів для повторного збирання неправдивих пакетів. Після цього система може зіткнутися з переповненням буфера або навіть збоєм.

16. Повільний вірус

Ця атака виділяється серед натовпу, оскільки вимагає дуже низької пропускної здатності і може бути здійснена за допомогою лише одного комп’ютера. Це працює, ініціюючи кілька одночасних з'єднань з веб-сервером і тримаючи їх відкритими протягом тривалого періоду часу. Зловмисник періодично надсилає часткові запити та доповнює їх заголовками HTTP, щоб переконатися, що вони не досягають етапу завершення. Як результат, здатність сервера підтримувати одночасне з’єднання вичерпується, і він більше не може обробляти підключення від законних клієнтів [14].

17. ReDoS

ReDoS розшифровується як "відмова у наданні регулярних виразів". Його мета - перевантажити реалізацію регулярних виразів програмою екземплярами дуже складних шаблонів пошуку рядків. Шкідливий суб’єкт може ініціювати сценарій обробки регулярних виразів, чия алгоритмічна складність призводить до того, що цільова система витрачає зайві ресурси та сповільнює або збій.

У загальному випадку DDoS-атаки можна розділити на декілька видів залежно від того, на якому рівні моделі взаємодії відкритих систем (OSI) відбувається атака. Атаки на мережевому рівні (рівень 3), транспортному рівні (рівень 4), рівні подання (рівень 6) і рівні додатків (рівень 7) найбільш поширені. Приклади векторів атак для кожного рівня моделі OSI можна побачити у Таблиці 1.1:

Таблиця 1.1

Вектори атак відмови в обслуговуванні відповідно до рівнів моделі OSI

#	Рівень	Інформація	Опис рівня	Приклад вектора
7	Прикладний	Дані	Мережевий процес на адресу застосунка	флуд DNS-запитів, HTTP-флуд
6	Подання	Дані	Подання та шифрування даних	Порушення SSL
5	Сеансовий	Дані	Сеанс зв'язку між хостами	Не застосовується
4	Транспортний	Сегменти	Зв'язок між кінцевими пунктами і надійність	SYN-Повінь
3	Мережевий	Пакети	Визначення маршруту і логічна адресація	Атаки з відображенням UDP-пакетів
2	Канальний	Кадри	фізична адресація	Не застосовується
1	Фізичний	Біти	Середовище передачі, сигнал і виконавчі дані	Не застосовується

2.3 Атаки викрадення сеансу

Типове викрадення сеансу - це добре відома атака "людина посередині" у світі мережевої безпеки та одна з найулюбленіших атак для зловмисників через її характер. Користувач, який вже ввійшов (автентифікований) на веб-сервер, має дійсний сеанс між клієнтом та сервером, зловмисник бере під контроль такий сеанс, в основному викрадає сеанс у користувача і продовжує підключення до сервера, видаючи себе за користувача. Це стає все більш поширеним, оскільки зловмисники

мають велику перевагу, тому що їм не потрібно витрачати години на злом пароля або намагатися здійснити інші атаки на сервер, оскільки користувач вже пройшов аутентифікацію та під час активної сесії це значно полегшує просто прослуховування трафіку в мережі без відома користувача.

Основна причина, через яку зловмиснику стає простіше прослуховувати мережу та видавати себе за користувача на сервері, це те, що, коли користувач спочатку аутентифікується на сервері, в деяких випадках сервер використовує для автентифікації захищене зашифроване з’єднання, наприклад HTTPS, а решта з'єднання користувача надсилається у звичайному тексті.

Існує три типи атак викрадення сесії:

- активне викрадення сеансу;
- пасивне викрадення сеансу;
- гібридне викрадення сеансу.

2.3.1 Активне викрадення сеансу

Активне викрадення сеансу - це таке, під час якого зловмисник бере під контроль активний сеанс жертви і починає маскуватися як справжнього користувача, спілкуючись із сервером. Існує декілька методів припинення підключення користувача до сервера, один із найпоширеніших - залити цільову машину величезним обсягом трафіку, і цей тип атаки відомий як Відмова в обслуговуванні. Роблячи це, зловмисник переводить користувача в офлайн режим, і тепер зловмисник має повний контроль над сеансом. Протягом усього цього процесу зловмисник перебуває в невидимому режимі, прослуховуючи та відстежуючи пакети, що обходять мережу, використовуючи інструменти сніфінгу пакетів, наприклад wireshark, ethereal, тощо.

Як проілюстровано на рисунку 2.1 нижче, зловмисник здійснює типову атаку викрадення сеансу між клієнтом та сервером. Трафік постійно контролюється за допомогою інструменту захоплення пакетів, а потім аналізуються пакети, щоб

зрозуміти, який пакет містить інформацію про сеанс, необхідну для автентифікації на сервері.

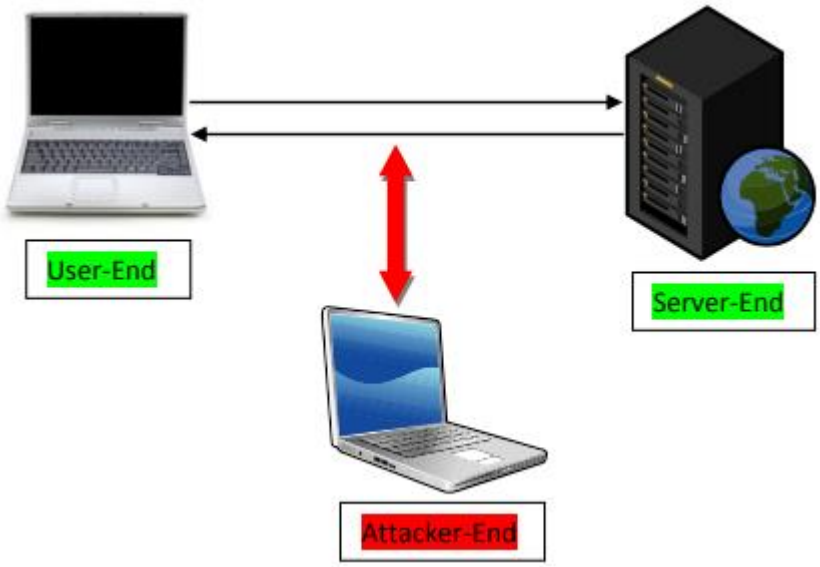

Рисунок 2.1 – Типова атака викрадення сеансу

2.3.2 Пасивне викрадення сеансу

Під час пасивного викрадення сеансу зловмисник прослуховує всі дані та фіксує їх для майбутніх атак. У більшості випадків для здійснення будь-якого виду атаки викрадення важливо, щоб зловмисник починав із пасивного режиму. Недоліком атаки в пасивному режимі є те, що зловмисник може бути не настільки ефективним, щоб досягти успіху, коли користувач видає себе за сервер, за винятком випадків, коли сеанс користувача все ще активний, але у більшості випадків цього не буде, якщо користувач вийде з сервера. Ще однією типовою атакою "людина посередині" є повторне відтворення сесії, на відміну від викрадення сесії, під час повторного відтворення зловмисник захоплює весь пакет і змінює інформацію про пакет, перш ніж відправляти його на сервер для аутентифікації, це типовий спосіб атаки "людина посередині", оскільки зловмисник фактично знаходиться між користувачем та сервером, змінюючи пакет та надсилаючи його. На наступному рисунку показано типову атаку відтворення сеансу:

Рисунок 2.2 – Атака відтворення сеансу

Зловмисник перериває трафік між сервером і клієнтом і модифікує його перед відправкою назад. Цей тип атаки викликає багато підозр у адміністратора мережі або самого користувача. А також займає набагато більше часу, ніж у порівнянні з іншими типами атак "людина посередині".

2.3.3 Гібридне викрадення сеансу

У гібридному захопленні сеансу зловмисник реалізує як пасивний, так і активний режими атаки для успішного завершення атак. При цьому типі атаки зловмисник відстежує структуру трафіку, який було надіслано через мережу, і зловмисник обирає сеанс для видання. Типовим прикладом може бути публічна незахищена бездротова мережа, де зловмисник має доступ до декількох поточних сеансів. Все, що зловмиснику потрібно зробити в цій ситуації, - це почекати правильного сеансу та викрасти його у користувача [15].

Надалі атаки викрадення сеансу можна класифікувати на 2 різних підтипи, які залежать від методів спуфінгу, що використовуються для атаки:

- сліпа спуфінг-атака;
- несліпа спуфінг-атака.

Сліпа спуфінг -атака:

Спуфінг - це техніка компрометації цільової машини без залучення уваги до атаки. Коли трафік не видно і якщо зловмисник просто скидає трафік між клієнтом і

сервером, а потім, вгадуючи номер послідовності tcp, намагається автентифікуватися на сервері. Це робить тип нападу найскладнішим, і в більшості випадків зловмисник в кінцевому підсумку проводить багато часу без жодного успіху (Andrew & Daniel, 2006).

Несліпа спуфінг-атака:

Це найпоширеніший тип атаки, оскільки під час сліпої підробки зловмисник може бачити трафік між клієнтом та серверною машиною. Це дозволяє зловмисникам легко аналізувати пакети в активному режимі та продовжувати атаку, видаючи себе за користувача на сервері. Це стає важко в комутованій мережі, оскільки комутатори передають не всі пакети на всі хости, а на певний хост. Але з деякою розширеною конфігурацією, якщо зловмисник може скомпрометувати порт VLAN (Віртуальна локальна мережа), то викрадення сеансу стає можливим в мережі.

Деякі інструменти, які використовуються для викрадення сесій:

- Hunt;
- T-Sight;
- Juggernaut;
- TTY Watcher;
- Hamster and Ferret;
- Wireshark;
- Ethereal;

2.3.4 Методи практичної реалізації атак викрадення сеансу

Існує чотири основні методи, які використовуються для здійснення викрадення сеансу. Це:

Фіксація сеансу, коли зловмисник встановлює ідентифікатор сеансу користувача на відомий їм, наприклад, надсилаючи користувачеві електронне

повідомлення із посиланням, що містить певний ідентифікатор сеансу. Тепер зловмисникові залишається лише почекати, поки користувач увійде в систему.

Викрадення сессії збоку (Session side jacking), де зловмисник використовує сніффінг пакетів для зчитування мережевого трафіку між двома сторонами для викрадення сеансового файлу cookie. Багато веб-сайтів використовують SSL-шифрування для сторінок входу, щоб запобігти зловмисникам бачити пароль, але не використовують шифрування для решти веб-сайтів після автентифікації. Це дозволяє зловмисникам, які можуть зчитувати мережевий трафік, перехоплювати всі дані, що надходять на сервер або веб-сторінки, які переглядає клієнт. Оскільки ці дані включають файли cookie сеансу, вони дозволяють видавати себе за жертву, навіть якщо сам пароль не порушений. Незахищені точки доступу Wi-Fi особливо вразливі, оскільки кожен, хто ділиться мережею, як правило, зможе прочитати більшу частину веб-трафіку між іншими вузлами та точкою доступу.

Міжсайтовий сценарій (Cross-site scripting або XSS) , коли зловмисник обманює комп'ютер користувача на запущений код, який вважається надійним, оскільки він, як видається, належить серверу, що дозволяє зловмисникові отримати копію файлу cookie або виконати інші операції.

Зловмисне програмне забезпечення та небажані програми можуть використовувати володіння браузером, щоб викрасти файли cookie браузера без відома користувача, а потім виконувати дії (наприклад, встановлення програм Android) без відома користувача. Зловмисник із фізичним доступом може просто спробувати викрасти ключ сеансу, наприклад, отримавши вміст файлу або пам'яті відповідної частини комп'ютера користувача або сервера.

Після успішного отримання відповідних файлів cookie сеансу суперник може скористатися технікою передачі файлів cookie для здійснення викрадення сесії.

2.4 Атаки ескалації привілеїв

Ескалація привілеїв може бути визначена як атака, яка передбачає отримання незаконного доступу до підвищених прав або привілеїв, понад те, що призначене

або має право для користувача. Ця атака може залучати зовнішнього суб'єкта загрози або інсайдера. Ескалація привілеїв є ключовим етапом ланцюга кібератак і, як правило, включає використання вразливості ескалації привілеїв, наприклад, помилки системи, неправильної конфігурації або неадекватного контролю доступу [16].

2.4.1 Вертикальна та горизонтальна ескалація привілеїв

Атаки ескалації привілеїв можна розділити на дві великі категорії - горизонтальну ескалацію привілеїв та вертикальну ескалацію привілеїв. Часто їх плутають між собою, ці терміни можна диференціювати наступним чином:

Горизонтальна ескалація привілеїв передбачає отримання доступу до прав іншого облікового запису - людини чи машини - з подібними привілеями. Ця дія називається "account takeover". Як правило, це стосуватиметься облікових записів нижчого рівня (тобто стандартного користувача), які можуть не мати належного захисту. Коли кожен новий горизонтальний обліковий запис скомпрометований, зловмисник розширює сферу свого доступу подібними привілеями.

Вертикальна ескалація привілеїв, також відома як атака на підвищення привілеїв, передбачає збільшення привілеїв / привілейованого доступу понад те, що вже має користувач, програма чи інший актив. Це передбачає перехід від низькорівневого привілейованого доступу до більшого обсягу привілейованого доступу. Досягнення вертикальної ескалації привілеїв може зажадати від зловмисника виконати ряд посередницьких кроків (тобто виконати атаку переповнення буфера тощо), щоб обійти або перевизначити елементи керування привілеями, або використати недоліки програмного забезпечення, прошивки, ядра або отримати привілейовані облікові дані для інших застосунків або самої операційної системи. У 2020 р. рівень вразливостей привілеїв становив 44% усіх вразливостей Microsoft, згідно зі звітом про вразливості Microsoft [16] за 2021 рік.

Кожен локальний, інтерактивний сеанс або сеанс віддаленого доступу представляє певну форму привілейованого доступу. Це охоплює все: від гостьових

привілеїв, що дозволяють лише локальний вхід, до адміністраторських або кореневих привілеїв для віддаленого сеансу та потенційно повного управління системою. Тому кожному обліковому запису, який може взаємодіяти з системою, присвоєні деякі привілеї.

Стандартний користувач рідко володіє правами на базу даних, конфіденційні файли або що-небудь цінне. Отже, як суб'єкт загроз орієнтується в середовищі та отримує права адміністратора або root, щоб використовувати їх як вектор атаки? Існує п'ять основних методів:

- експлуатація облікових даних;
- вразливості та експлойти;
- неправильні конфігурації;
- зловмисне програмне забезпечення;
- соціальна інженерія.

Суб'єкти загроз, як правило, йдуть шляхом найменшого опору. Якщо час дозволить, вони приберуть докази своєї діяльності, щоб залишатися не виявленими. Незалежно від того, чи це стосується маскування їх вихідної IP-адреси або видалення журналів на основі облікових даних, якими вони користуються, будь-які дані про їх наявність відображають показник компрометації (IoC). Як тільки організація виявить вторгнення, вона може відстежувати наміри зловмисника та / або потенційно призупинити або припинити сеанс доступу.

Як правило, другий крок у ланцюжку кібератак передбачає ескалацію привілеїв до облікових записів з адміністративними, корінними або вищими привілейованими правами, ніж акаунт, який був скомпрометований спочатку. Звичайно, можливо, початкова компрометація стосувалася адміністративного чи кореневого облікового запису. Якщо це так, суб'єкт загроз продовжує розвивати свої шкідливі плани і, можливо, вже є власником середовища.

Вразливості підвищення привілеїв (які дозволяють вертикальну ескалацію привілеїв) є причиною багатьох найгірших подвигів за останні роки, включаючи BlueKeep, WannaCry та NotPetya.

2.4.2 Вектори атак ескалації привілеїв

Вектор атаки - це техніка, за допомогою якої суб'єкт загрози, хакер або зловмисник отримує доступ до системи, додатку чи ресурсу для здійснення зловмисної діяльності. Це може включати все, від встановлення шкідливого програмного забезпечення, зміни файлів або даних, до навіть певної форми постійної розвідки.

Вектори атак ескалації привілеїв, можливо, представляють найгіршу з усіх кіберзагроз, оскільки зловмисник може стати адміністратором та власником усіх ресурсів інформаційних технологій у вашій компанії. І завдяки цій владі всі дані, активи, додатки та ресурси потенційно можуть потрапити під певний вид несанкціонованого контролю.

Найпоширеніші методи, які може бути використано для атак ескалації привілеїв:

Атаки на словники - це автоматизована техніка, яка використовує список паролів проти дійсного облікового запису для виявлення пароля. Сам список є словником слів. Основні зловмисники паролів використовують ці списки поширених окремих слів, таких як «бейсбол», щоб зламати пароль, зламати обліковий запис та розкрити повну інформацію, що використовується для автентифікації.

Якщо суб'єкт загроз знає ресурс, який вони намагаються скомпрометувати, наприклад вимоги до довжини пароля та складності, він може налаштувати словник для більш ефективного націлювання на ресурс. Тому більш просунуті програми часто використовують словник на додачу до змішування цифр або загальних символів на початку або в кінці спроби імітувати реальний пароль із вимогами складності.

Слабкою стороною словникових атак є те, що вони покладаються на реальні слова та похідні слова, надані користувачем словника за замовчуванням. Якщо справжній пароль вигаданий, використовує кілька мов або використовує більше одного слова чи фрази, це зірве атаку за словником.

Найпоширенішими методами зменшення загрози словникової атаки є спроби блокування облікового запису та політики складності паролів. Захист від блокування означає, що після "n" разів помилкових спроб обліковий запис користувача автоматично блокується на певний час, розблокується відповідальною особою вручну або за допомогою автоматичного рішення для скидання пароля.

Однак у багатьох середовищах, особливо для нелюдських облікових записів, спроби блокування облікового запису можуть перешкоджати роботі бізнесу. Тому цей параметр безпеки іноді вимикається. Отже, якщо помилки входу не відстежуються в журналах подій, атака на словник є ефективним вектором атаки для суб'єкта загрози. Це особливо вірно, якщо в привілейованих облікових записах цей параметр не ввімкнено як стратегію зменшення наслідків. І пам'ятайте, атака на словник може включати найпоширеніші слова, змінені за допомогою простих прийомів складності паролів, таких як зміна "a" на "@" або "o" на "0". Тому лише складність - не найкраще рішення [17].

Атаки Rainbow Table - це підмножина атак на словники. Якщо зловмисник знає алгоритм хешування паролів, який використовується для шифрування паролів для ресурсу, таблиці веселки можуть дозволити їм здійснити зворотне проектування цих хешів у фактичні паролі. У хакера є хеш-словники, які можна зіставити з вихідним паролем. Сучасні порушення відкрили величезні масиви хешу паролів, але без основи в алгоритмі шифрування, таблиці веселки та подібні методи майже марні без певної форми інформації.

Атаки Brute Force є найменш ефективним методом спроби зламати пароль, тому, як правило, використовуються в крайньому випадку. Атаки грубої сили використовують програмний метод, щоб спробувати всі можливі комбінації для пароля. Цей метод ефективний для паролів, коротких за довжиною та складністю рядків (символів), але може стати неможливим - навіть для найшвидших сучасних систем - із паролем від восьми символів або більше.

Якщо пароль містить лише алфавітні символи, усі великі літерами або всі малі літерами (не змішані), знадобиться 8031810176 здогадок. У вас більше шансів виграти в лотерею! Ця оцінка також передбачає, що зловмисник загроз знає

довжину пароля та вимоги до складності. Інші фактори включають цифри, чутливість до регістру та спеціальні символи локалізованою мовою.

Хоча атака грубої сили з належними параметрами врешті-решт знайде пароль, час і обчислювальна потужність можуть зробити тестування грубої сили самим суперечливим до того моменту, коли це буде зроблено. При цьому час, необхідний для виконання атак, залежить не тільки від швидкості, необхідної для генерації всіх можливих перестановок пароля, але також від складності та часу відгуку на несправність цільової системи. Цей час затримки відповіді - це те, що насправді важливо при спробі грубого примусу пароля.

Атаки Pass-the-Hash (PtH) - це техніка злому, яка дозволяє зловмисникові автентифікуватися на ресурсі, використовуючи відповідний NT LAN Manager (NTLM) хеш пароля користувача, замість використання зручного для читання пароля облікового запису. Після того, як суб'єкт загрози отримує дійсне ім'я користувача та хеш для пароля за допомогою різноманітних методів, таких як вишкрібання активної пам'яті системи, вони можуть використовувати облікові дані для автентифікації на віддаленому сервері або службі за допомогою автентифікації LM або NTLM.

Атаки PtH використовують слабку сторону реалізації в протоколі аутентифікації, де хеш пароля залишається статичним для кожного сеансу, поки сам пароль не буде змінений. PtH може виконуватися майже на будь-якому сервері чи службі, що приймають аутентифікацію LM або NTLM, незалежно від того, використовує ресурс Windows, Unix, Linux або іншу операційну систему. На жаль, сучасне шкідливе програмне забезпечення може містити методи вилучення пам'яті на хеші, що робить будь-якого активного запущеного користувача, програму, службу або обробляє потенційною ціллю. Після отримання хешу командування та управління або інша автоматизація дозволяє здійснювати додаткове бічне переміщення (горизонтальне) або вилучення даних.

Сучасні системи можуть захищатись від атак, що проходять через хеш, різними способами. Однак часта зміна пароля (після кожного інтерактивного сеансу) є хорошим захистом, щоб хеш не змінювався між сесіями. Рішення для

управління паролями, які можуть часто обертати паролі або налаштовувати маркер безпеки, є хорошим захистом від цієї техніки.

«Набивання» облікових даних. «Набивання» облікових даних - це тип автоматизованої техніки злому, яка використовує викрадені облікові дані, що складаються зі списків імен користувачів (або адрес електронної пошти) та відповідних паролів для отримання несанкціонованого доступу до системи або ресурсу. Зазвичай ця техніка включає автоматизацію подання запитів на вхід до програми та фіксацію успішних спроб входу в систему для подальшої експлуатації.

Такі атаки не намагаються зламати або вгадати будь-які паролі. У цих атаках суб'єкт загроз автоматизує автентифікацію на основі раніше виявлених облікових даних. Результатом можуть бути мільйони спроб визначити, де користувач потенційно повторно використовував свої облікові дані на іншому веб-сайті чи в додатку. «Набивання» облікових даних атакує повторне використання паролів і ефективно лише тому, що багато користувачів повторно використовують одні й ті самі комбінації облікових даних на кількох сайтах.

Зміни та скидання паролів:

Без менеджера паролів зберігати всі свої паролі унікальними та складними є непростим завданням - навіть для найдосвідченішого фахівця з безпеки.

На жаль, існує загальний ризик скидання (не плутати зі зміною) паролів, що робить їх цільовими для суб'єктів загроз. Скидання пароля - це акт примусової зміни пароля кимось іншим, а не зміна, ініційована користувачем пароля. Ризики, пов'язані зі скиданням паролів, включають:

- паролі на основі шаблону, які легко вгадуються, під час скидання;
- паролі, які скидаються за допомогою електронної пошти або текстового повідомлення та зберігаються кінцевим користувачем;
- скидання паролів довідковою службою, яке використовується повторно щоразу, коли запитується скидання пароля;
- автоматичне скидання пароля, яке надається наосліп через блокування облікового запису;

- паролі, які передаються усно і їх можна почути вголос;
- складні скидання паролів, записані кінцевим користувачем.

Щоразу, коли пароль скидається, існує неявне підтвердження того, що старий пароль знаходиться під загрозою та потребує зміни. Можливо, пароль був забутий, закінчився термін дії або сталося блокування через численні невдалі спроби. Скидання, передача та зберігання нового пароля є ризиком, поки кінцевий користувач не змінить пароль знову.

Атаки перерахування ідентифікаційних даних, включаючи ті, що експлуатують sudo, трапляються, коли суб'єкт загрози може застосувати такі методи, як груба сила, щоб або вгадати, або підтвердити, що дійсні користувачі доступні для автентифікації ресурсу. Перерахування користувачів часто асоціюється з веб-програмами, хоча його також можна знайти в будь-якій програмі, яка вимагає традиційної автентифікації на основі користувачів та облікових даних. Дві найпоширеніші області, де відбувається перерахування користувачів:

На сторінці входу в програму на основі невдалої відповіді автентифікації.

Функція "Забутий пароль", яка може викликати робочий процес або відповісти "обліковий запис не знайдено"

По суті, суб'єкт загроз шукає відповідь сервера на основі дійсності поданих облікових даних, щоб визначити, чи дійсний обліковий запис, який вони спробували. Це загальний механізм реагування для багатьох додатків.

Коли користувач вводить дійсне ім'я користувача та недійсний пароль, сервер повертає відповідь із повідомленням про неправильний пароль. Якщо суб'єкт загрози вводить недійсне ім'я користувача, незалежно від пароля, типові програми відповідають, не знаходячи облікового запису. Отже, суб'єкт загроз може визначити, чи використовує його спроба злому дійсний обліковий запис і неправильний пароль, чи обліковий запис, який вони намагаються, ніколи не буде автентифікувати. На основі автоматизації та перевірки грубої сили вони можуть перерахувати дійсні облікові записи ресурсу та робити спроби майбутніх

привілейованих атак на основі загальних паролів, повторно використаних паролів або інших даних, отриманих від попередніх атак.

Нарешті, якщо суб’єкт загрози може визначити шаблон іменування для компанії (тобто перше початкове прізвище), то створення списку для переліку та майбутніх атак стає набагато простішим.

Шкідливе програмне забезпечення - це будь-яка частина комп'ютерного програмного забезпечення (включаючи мікропрограму, мікрокод тощо), написана з метою пошкодження пристроїв, викрадення даних і, як правило, призводить до того, що ресурс поводиться не так, як передбачалося.

Існує вісім різних типів та джерел шкідливих програм, будь-який з яких може бути використаний для атак ескалації привілеїв:

Баги: тип помилок, недоліків, вразливостей або збоїв, що призводить до небажаних або несподіваних результатів через погане програмне кодування або несподівані умови експлуатації. Помилки можуть існувати в будь-якому типі програмного забезпечення. Коли помилки можуть бути використані проти програми та її даних, вони називаються вразливими місцями (тобто підвищеною вразливістю привілеїв), а програмне забезпечення, що використовується для їх використання, називаються експлойтами. Технічно, помилка сама по собі не є справжнім шкідливим програмним забезпеченням, оскільки вона не створюється зі зловмисними намірами, але коли вона використовується, вона може бути настільки ж руйнівною. У ігровому світі їх зазвичай називають "глюками" [18].

Черв’яки: покладаються на помилки, вразливості та експлойти, щоб доставити корисне навантаження та розповсюдитись на інші ресурси. Початкові зараження можуть ховатися у вкладеннях або файлах, які завантажуються, але після їх запуску вони можуть просканувати мережу (або Інтернет) на наявність інших вразливих систем. Виходячи з їх конструкції, вони споживають величезну кількість смуги пропускання або працюють у повільному, невидимому режимі. Черви можуть повністю вимкнути мережу або веб-сервер. Вимагальна програма, яка може самостійно поширюватися, щоб заразити кілька систем самостійно, є різновидом хробака.

Вірус: будь-яка шкідлива програма, яка завантажується на ваш веб-сайт або комп'ютер без вашого відома. Намір вірусу може бути не очевидним із початкової інфекції, і, загалом, він може знаходитись на ресурсі, доки його не спровокують виконати зловмисну дію. Вірус може використовувати прийоми, щоб затушувати виявлення, наприклад, при захопленні або приховуванні DLL (бібліотеки динамічних посилань) як кореневий набір.

Боти: зловмисне програмне забезпечення, створене для виконання певного набору завдань із відомим наміром. Суб'єкт загрози може використовувати ботів для надсилання спаму або участі в атаці відмови в розподілі (DDoS), щоб зруйнувати цілий веб-сайт, мережу або Інтернет-службу. Боти можуть служити транспортним засобом для горизонтальної привілейованої ескалації в поєднанні з хробаками і часто є частиною спостереження за атакою.

Троян: Подібно до міфічного Троянського коня, це шкідливе програмне забезпечення маскується під звичайний файл або додаток і обманює користувача завантажувати, відкривати або виконувати його. Корисне навантаження може запустити будь-яку іншу форму шкідливого програмного забезпечення і продовжувати обманювати користувача, вважаючи, що він взаємодіє із законним програмним забезпеченням. Атаки на основі автентифікації, як правило, базуються на троянських програмах.

Вимагач: програмне забезпечення забороняє доступ до ваших файлів, як правило, за допомогою шифрування, і вимагає викуп (як правило, у формі цифрових та криптовалют, таких як біткойн), щоб звільнити владу загрози щодо ваших даних. Якщо викуп сплачений, а суб'єкт загрози працює із справжньою службою викупу, вони нададуть спосіб дешифрування ваших файлів і дозволять вам отримати доступ до активів. У деяких випадках жертва сплачує викуп, але суб'єкт загрози давно відмовляється від їхньої схеми, залишаючи жертву зараженими системами та непоправними фінансовими втратами.

Рекламне ПЗ: Рекламне ПЗ - це тип шкідливого ПЗ, який автоматично відображає небажану та потенційно незаконну рекламу кінцевому користувачеві. Натискання посилання на електронну пошту або відкриття вкладення може

завантажити шкідливе програмне забезпечення, запустити експлойт або перенаправити вас на шкідливий веб-сайт. Мета - виставити кінцевому користувачеві невідповідні послуги та обдурити їх на виконання додаткових кроків для завантаження більшої кількості шкідливого програмного забезпечення чи програмного забезпечення на основі спостереження.

Шпигунське програмне забезпечення: Шпигунське програмне забезпечення - це тип шкідливого програмного забезпечення, яке здійснює спостереження за діяльністю користувача. Ці функції можуть включати моніторинг екрана користувача, зйомку натискань клавіш і навіть увімкнення камери та мікрофона об’єкта для спостереження. Ця інформація збирається та передається через Інтернет або зберігається локально для подальшого пошуку суб’єктом загрози. У сучасному світі, поряд із вимогами, це найнебезпечніше шкідливе програмне забезпечення, яке використовують суб’єкти загроз, оскільки зібрана інформація, як правило, полегшує ескалацію привілеїв.

2.5 Методи захисту від розглянутих загроз

2.5.1 Захист від атак викрадення сеансу

Для захисту від викрадення сеансів існують різні засоби виявлення вторгнень та деякі просунуті методи. Нижче розглянуто декілька часто використовуваних інструментів:

- Arp-ON;
- ARP-PING;
- ANTI-SNIFF;
- Cookie Monster;
- Wavelet based detection;
- Intrusion Detection System (IDS);
- Intrusion Prevention System (IPS).

Дослідження працюють над створенням розумної системи, яка може не тільки виявляти, але й запобігати будь-яким спробам викрадення сесії. Існує кілька алгоритмів та методів, заснованих на дослідженнях, які були запропоновані та успішно впроваджені. Наприклад:

Забезпечення безпечного формування файлів cookie

Файл cookie - це той, в якому вся інформація про користувача надсилається для автентифікації певного користувача на сервер. Деякі відомості, вбудовані в файли cookie:

- ім'я користувача (логін);
- пароль;
- ID (Ідентифікатор сесії);
- дата створення сеансу - позначка часу;
- час закінчення сеансу.

Коли у пасивному або активному режимі зловмисник викрадає сесію, він аналізує цю інформацію про файли cookie, що надсилаються через мережу, для автентифікації та компрометації облікового запису користувача.

Впровадження технології CIA, C- Confidentiality, I - Integrity, A- Authentication

Було запропоновано недороге надійне рішення: змінити cookie, що генерується для кожної передачі між клієнтом та сервером, що реалізує технологію CIA. Це досягається створенням Java-скрипта на стороні сервера та наданням файлу cookie доступного лише браузеру клієнта, що також допомагає запобігти міжсайтовим сценаріям. Пізніше протокол, що використовувався для захисту бездротової мережі, був розроблений як стандартний плагін, який можна використовувати для таких браузерів, як Firefox, Google chrome тощо.

Використання зашифрованого з'єднання для передачі, як HTTPS

Блокування певного сеансу для відповідних користувачів

Дуже ефективним методом запобігання викраденню сеансу є блокування сеансу для конкретного користувача, і це гарантує, що навіть якщо сеанс був викрадений, інформація, яку отримає зловмисник, не буде йому корисною.

Згенерований з початкового входу через SSL маркер зберігається і повторно використовується в ідентифікаторі фрагмента браузера, що робить ефективне та недороге рішення для захисту сеансу та від прослуховування.

2.5.2 Як запобігти та пом'якшити атаки ескалації привілеїв

Оскільки атаки з ескалації привілеїв можуть запускати і просувати безліч різних шляхів, для захисту потрібні численні стратегії та тактики захисту. Однак впровадження підходу, орієнтованого на особистість, та управління привілейованим управлінням доступом допоможуть організації захиститися від найширшого спектру атак і досягнути найменших можливостей для зменшення впливу атаки. Ось декілька найкращих практик:

- Повністю керувати життєвим циклом ідентифікаційних даних, включаючи надання та скасування надання ідентифікаційних даних та облікових записів, щоб переконатися, що немає осідаючих облікових записів, які можуть бути викрадені.
- Використовувати рішення для управління паролями, щоб послідовно застосовувати ефективні практики управління обліковими даними (виявлення, зберігання, централізоване управління, реєстрація, виїзд) як для людей, так і для машин. Це також передбачає усунення типових та жорстко закодованих облікових даних.
- Застосовувати найменші привілеї: вилучити права адміністратора в користувачів і зменшити привілеї програм та машин до необхідного мінімуму. Також слід застосовувати своєчасний доступ, щоб зменшити постійні або постійні привілеї.
- Застосовувати розширений контроль та захист додатків, щоб забезпечити детальний контроль над усіма спробами доступу до програм, спілкування та підвищення привілеїв.
- Відстежувати та керувати усіма привілейованими сеансами, щоб виявляти та швидко вирішувати будь-яку підозрілу активність, яка може свідчити про

викрадений обліковий запис або незаконну спробу ескалації привілеїв або бічний рух.

- Затвердити системи та програми: Це доповнює принцип найменших привілеїв і може включати зміни конфігурації, видалення непотрібних прав та доступу, закриття портів тощо. Це покращує безпеку системи та додатків та допомагає запобігти та зменшити потенціал для помилок, які залишають уразливість до введення шкідливого коду (тобто ін'єкції SQL), переповнення буфера тощо або інших бекдорів, які можуть дозволити ескалацію привілеїв.
- Управління вразливостями: Постійно виявляйте та усувайте уразливості, такі як виправлення, виправлення помилкових конфігурацій, усунення типових та / або вбудованих облікових даних тощо.
- Безпечний віддалений доступ завжди слід контролювати та керувати ним для будь-якої форми привілейованого доступу, оскільки атаки можуть відбуватися горизонтально та вертикально для використання привілеїв [19].

2.5.3 Методи захисту від DDoS-атак

1. Зменшення зон, доступних для атаки

Одним з перших методів нейтралізації DDoS-атак є зведення до мінімуму розміру зони, яку можна атакувати. Подібний прийом обмежує можливості зловмисників для атаки і забезпечує можливість створення централізованого захисту. Необхідно переконатися, що доступ до додатка або ресурсів не було відкрито для портів, протоколів або додатків, взаємодія з якими не передбачено. Таким чином, зведення до мінімуму кількості можливих точок для атаки дозволяє зосередити зусилля на їх нейтралізації. У деяких випадках цього можна домогтися, розмістивши свої обчислювальні ресурси за мережами розповсюдження контенту або балансувальник навантаження і обмеживши прямий інтернет-трафік до певних частин своєї інфраструктури, таким як сервери баз даних. Також можна

використовувати брандмауери або списки контролю доступу (ACL) , щоб контролювати, який трафік надходить в додатки.

2. План масштабування

Двома основними елементами нейтралізації великомасштабних DDoS-атак є пропускна здатність (або транзитний потенціал) і продуктивність сервера, достатня для поглинання і нейтралізації атак.

Транзитний потенціал. При проектуванні додатків необхідно переконатися, що постачальник послуг хостингу надає надлишкову пропускну здатність підключення до Інтернету, яка дозволяє обробляти великі об'єми трафіку. Оскільки кінцева мета DDoS-атак - вплинути на доступність ресурсів або додатків, необхідно розміщувати їх поруч не тільки з кінцевими користувачами, а й з великими вузлами мережевого обміну трафіком, які легко забезпечать вашим користувачам доступ до додатка навіть при великому обсязі трафіку. Робота з інтернет-додатками забезпечує ще більш широкі можливості. В цьому випадку можна скористатися мережами розповсюдження контенту (CDN) і сервісами інтелектуального перетворення адрес DNS, Які створюють додатковий рівень мережевої інфраструктури для обслуговування контенту і дозволу DNS-запитів з місць, які часто розташовані ближче до кінцевих користувачів.

Продуктивність сервера. Більшість DDoS-атак є об'ємними і споживають багато ресурсів, тому важливо мати можливість швидко збільшувати або зменшувати обсяг своїх обчислювальних ресурсів. Це можна забезпечити, використовуючи надлишковий обсяг обчислювальних ресурсів або ресурси зі спеціальними можливостями, такими як більш продуктивні мережеві інтерфейси або поліпшена мережева конфігурація , що дозволяє підтримувати обробку великих обсягів трафіку. Крім того, для постійного контролю і розподілу навантажень між ресурсами та запобігання перевантаження будь-якого одного ресурсу часто використовуються відповідні балансувальник.

3. Відомості про типовий і нетиповий трафік

Кожен раз, коли виявляється підвищення обсягу трафіку, що потрапляє на хост, в якості орієнтира можна брати максимально можливий обсяг трафіку, який

хост може обробити без погіршення його доступності. Така концепція називається обмеженням швидкості. Більш просунуті методи захисту відповідно володіють додатковими можливостями і можуть інтелектуально приймати тільки трафік, який дозволений, аналізуючи окремі пакети. Для використання подібних засобів необхідно визначити характеристики хорошого трафіку, який зазвичай отримує цільовий об'єкт, і мати можливість порівнювати кожен пакет з цим еталоном.

4. Розгортання брандмауерів для відображення складних атак рівня додатків

Проти атак, які намагаються використовувати уразливість в додатку, наприклад проти спроб впровадження SQL-коду або підробки міжсайтових запитів, рекомендується використовувати Web Application Firewall (WAF) . Крім того, через унікальність цих атак ви повинні бути здатні самостійно нейтралізувати заборонені запити, які можуть мати певні характеристики, наприклад можуть визначатися як відмінні від хорошого трафіку або виходити з підозрілих IP-адрес, з несподіваних географічних регіонів і т. д.

2.3 Модель загроз серверній частині веб-застосунку

Враховуючи загрози серверній частині веб-застосунку, що були разглянуті в другому розділі дипломної роботи, було побудовано наступну модель загроз щодо зовнішніх порушників:

Таблиця 3.1

Модель загроз серверній частині веб-застосунків

№	Загроза	Властивість, що порушується	Рівень шкоди	Імовірність
1	Міжсайтовий скриптинг	конфіденційність	середній	Середня
2	Відмова в обслуговуванні	Доступність	високий	Висока
3	Шкідливе ПЗ	Цілісність, доступність,	високий	Висока

		конфіденційність		

Продовження Таблиці 3.1

4	Експлуатація вразливостей програмного коду	Цілісність, доступність, конфіденційність	Високий	Середня
5	Експлуатація облікових даних	Цілісність, доступність, конфіденційність	Високий	Висока
6	Соціальна інженерія	конфіденційність	Низький	Низька
7	Експлуатація неправильних конфігурацій	Цілісність, доступність, конфіденційність	Високий	Висока
8	Сніфінг пакетів, що передаються мережею	Конфіденційність	Низький	Висока

Висновки за розділом 2

У другому розділі дипломної роботи було оглянуто три глобальних типи атак на серверну частину веб-застосунку:

- атаки, які мають за мету відмову в обслуговуванні, що можуть бути націлені на мережу, щоб унеможливити передачу даних через перенавантаження мережі, на систему, щоб вичерпати її ресурси та на додатки, щоб безпосередньо завадити в обслуговуванні клієнтів. Як приклади таких атак, було наведено різні методики проведення, наприклад різноманітний флуд.

- Атаки, націлені на викрадення активного сеансу користувача, щоб отримати доступ на сервер під виглядом вже автентифікованого користувача.
- Атаки, націлені на ескалацію привілеїв, щоб збільшити можливості доступу або модифікації інформації на сервері.

Також було побудовано модель загроз веб-застосунку, включаючі розглянуті атаки та інші відомі мережеві загрози.

РОЗДІЛ 3
ПОБУДОВА ПОТЕНЦІЙНО ЗАХИЩЕНОЇ СЕРВЕРНОЇ ЧАСТИНИ ВЕБ-ЗАСТОСУНКУ

3.1 Розгортання веб-застосунку на основі LAMP-стеку. Опис компонентів LAMP-стеку

LAMP-стек - це дуже поширений приклад стека веб-сервісу, названий акронімом назв вихідних чотирьох компонентів з відкритим кодом: операційна система Linux, HTTP-сервер Apache, система управління реляційними базами даних MySQL та мова програмування PHP .

Операційна система Linux

Linux - це безкоштовна операційна система з відкритим кодом, випущена під загальною публічною ліцензією GNU (GPL). Будь-хто може запускати, вивчати, модифікувати та розповсюджувати вихідний код або навіть продавати копії свого модифікованого коду, якщо це робиться за тією самою ліцензією.

Linux став найбільшим проектом програмного забезпечення з відкритим кодом у світі. Професійні та програмісти- любителі з усього світу роблять внесок у ядро Linux, додаючи функції, знаходячи та виправляючи помилки та недоліки безпеки, та пропонуючи нові ідеї - і все це, ділячись своїми внесками з громадою.

Операційна система - це програмне забезпечення, яке безпосередньо управляє апаратним забезпеченням та ресурсами системи, такими як ЦП, пам’ять та пам’ять. ОС розміщена між додатками та обладнанням та забезпечує зв’язок між усім програмним забезпеченням та фізичними ресурсами, що виконують роботу.

Linux був розроблений, щоб бути схожим на UNIX, але еволюціонував до роботи на широкому спектрі обладнання від телефонів до суперкомп'ютерів. Кожна ОС на базі Linux включає ядро Linux - яке управляє апаратними ресурсами - і набір програмних пакетів, що складають решту операційної системи [20].

ОС включає деякі загальні основні компоненти, зокрема інструменти GNU. Ці інструменти дають користувачеві можливість керувати ресурсами, наданими ядром, встановлювати додаткове програмне забезпечення, налаштовувати параметри продуктивності та безпеки тощо. Всі ці інструменти, що входять до складу, складають функціональну операційну систему. Оскільки Linux - це ОС з відкритим кодом, комбінації програмного забезпечення можуть відрізнятися залежно від дистрибутивів Linux.

Побудова потенційно захищеного веб-сайту в даній роботі буде відбуватися на базі Linux Ubuntu 16.04.

Ubuntu - це операційна система Linux на базі Debian, яка працює від робочого столу до хмари до всіх ваших підключених до Інтернету речей. Це найпопулярніша у світі операційна система для загальних хмар та OpenStack. Це платформа номер один для контейнерів; від Docker до Kubernetes до LXD, Ubuntu може запускати ваші контейнери в масштабі. Швидкий, безпечний і простий, Ubuntu забезпечує мільйони ПК у всьому світі.

Розробкою Ubuntu керує Canonical Ltd. Canonical приносить прибуток за рахунок продажу технічної підтримки та інших послуг, пов'язаних з Ubuntu. Проект Ubuntu публічно відданий принципам розробки програмного забезпечення з відкритим кодом; людям пропонується використовувати безкоштовне програмне забезпечення, вивчати, як воно працює, вдосконалювати його та поширювати.

Веб-сервер Apache

Apache вважається програмним забезпеченням з відкритим кодом, що означає, що оригінальний вихідний код є у вільному доступі для перегляду та співпраці. Відкритий код зробив Apache дуже популярним серед розробників, які створили та сконфігурували власні модулі для застосування певної функціональності та вдосконалення основних функцій [21].

Apache існує з 1995 року і відповідає як основна технологія, яка допомогла стимулювати початкове зростання Інтернету в зародковому стані. Одним з плюсів Apache є його здатність обробляти великі обсяги трафіку з мінімальною конфігурацією. Він легко масштабується, і завдяки своїй модульній

функціональності в основі, ви можете налаштувати Apache робити те, що ви хочете, як хочете. Також можливо видалити небажані модулі, щоб зробити Apache більш легким та ефективним.

Apache можна розгорнути на Linux, MacOS та Windows. Продукт функціонує як спосіб комунікації через мережі від клієнта до сервера за допомогою стеку протоколів TCP / IP.

Особливості веб-сервера Apache:

- завантажувані динамічні модулі;
- кілька режимів обробки запитів, включаючи Event-based/Async, Threaded and Prefork;
- високо масштабований (легко обробляє більше 10000 одночасних з'єднань);
- обробка статичних файлів, файлів індексів, автоматичне індексування та узгодження вмісту;
- підтримка конфігурації .htaccess для кожного каталогу;
- зворотний проксі з кешуванням;
- балансування навантаження за допомогою внутрішньосмугових перевірок працездатності;
- відмовостійкість і автоматичне відновлення після відмови;
- підтримка WebSocket, FastCGI, SCGI, AJP та uWSGI з кешування
- динамічна конфігурація;
- TLS / SSL з підтримкою скріплення SNI та OCSP через OpenSSL або wolfSSL;
- віртуальні сервери на основі імен та IP-адрес;
- сумісний з IPv6;
- підтримка HTTP / 2;
- дрібнодисперсний контроль автентифікації та авторизації доступу;
- стиснення та декомпресія gzip;
- переписування URL-адрес;

- перезапис заголовків та вмісту;
- регулювання швидкості обробки запитів;
- регулювання смуги пропускання;
- геолокація на основі IP-адреси;
- відстеження користувачів та сеансів;
- вбудовані сценарії Perl, PHP та Lua;
- public_html веб-сторінки для користувача;
- узагальнювач парсерних виразів;
- перегляди стану в реальному часі;
- підтримка FTP (окремим модулем).

База даних MySQL

MySQL - це реляційна система управління базами даних (СУБД), розроблена Oracle, яка базується на структурованій мові запитів (SQL).

База даних - це структурована сукупність даних. Це може бути що завгодно - від простого списку покупок до галереї картин чи місця для зберігання величезної кількості інформації в корпоративній мережі. Зокрема, реляційна база даних - це цифрове сховище, яке збирає дані та організовує їх відповідно до реляційної моделі. У цій моделі таблиці складаються з рядків і стовпців, а взаємозв'язки між елементами даних мають сувору логічну структуру. СУБД - це просто набір програмних засобів, що використовуються для фактичної реалізації, управління та запитування такої бази даних [22].

MySQL є невід’ємною частиною багатьох найпопулярніших стеків програмного забезпечення для створення та обслуговування всього - від веб-додатків, спрямованих на клієнтів, до потужних B2B-послуг, керованих даними. Його природа з відкритим кодом, стабільність та багатий набір функцій, що поєднуються з постійною розробкою та підтримкою від Oracle, означають, що такі критично важливі для Інтернету організації, як Facebook, Flickr, Twitter, Wikipedia та YouTube, також використовують MySQL.

Мова програмування PHP

PHP розпочався як невеликий проект з відкритим кодом, який еволюціонував у міру того, як все більше людей дізнавалося, наскільки це корисно. Расмус Лердорф випустив першу версію PHP ще в 1994 році.

PHP - це рекурсивна абревіатура від "PHP: Гіпертекстовий препроцесор".

PHP - це мова сценаріїв на стороні сервера, яка вбудована в HTML. Застосовується для управління динамічним вмістом, базами даних, відстеженням сеансів, навіть побудовою цілих сайтів електронної комерції.

Він інтегрований з низкою популярних баз даних, включаючи MySQL, PostgreSQL, Oracle, Sybase, Informix та Microsoft SQL Server.

PHP дуже приємний у виконанні, особливо коли він компілюється як модуль Apache на стороні Unix. Після запуску сервер MySQL виконує навіть дуже складні запити з величезними наборами результатів за час встановлення записів.

PHP підтримує велику кількість основних протоколів, таких як POP3, IMAP та LDAP. PHP4 додав підтримку Java та архітектури розподілених об'єктів (COM та CORBA), що вперше надало можливість розвитку n-рівня.

Синтаксис PHP подібний до C.

3.2 Послідовне налаштування LAMP-стеку

Командою *apt install lamp-server^* встановлюються всі компоненти стеку разом. В процесі інсталляції пакетів сервісів, здійснюється їх базова конфігурація:

Першим кроком, потрібно встановити та повторити пароль юзера-адміністратора “root”, щоб обмежити доступ до бази даних іншим користувачам (рис.3.1-3.2):

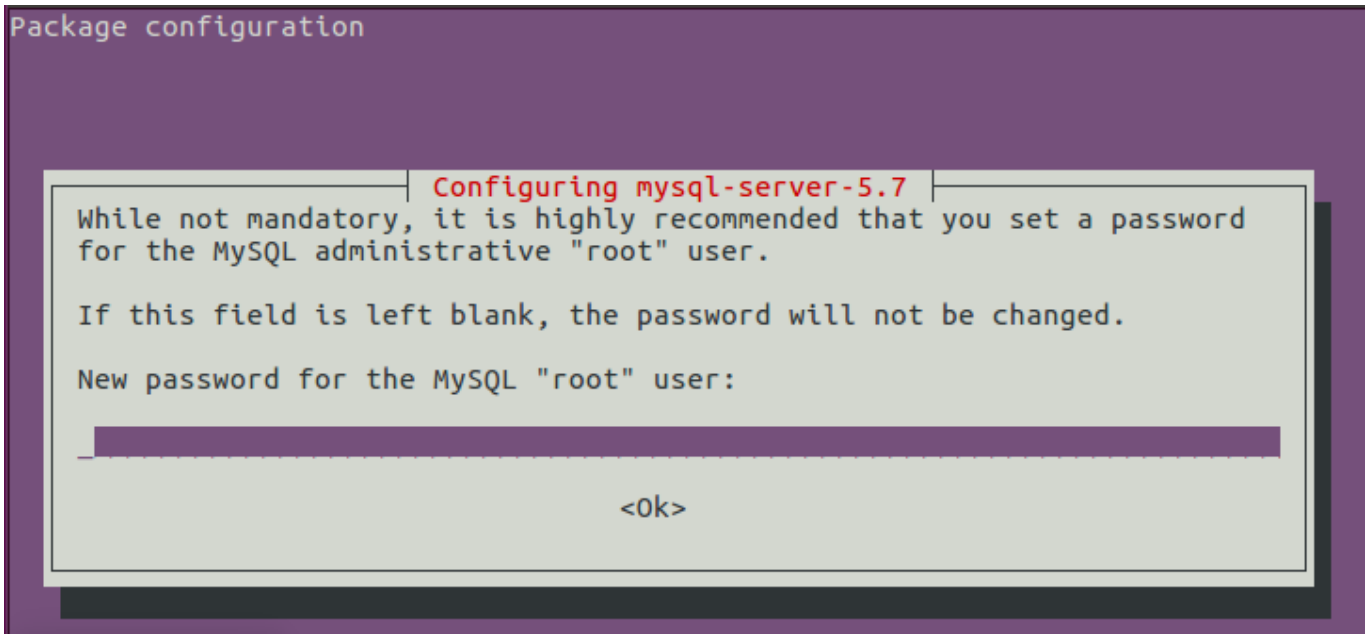

Рисунок 3.1 – Запит пароля при кофігурації бази даних

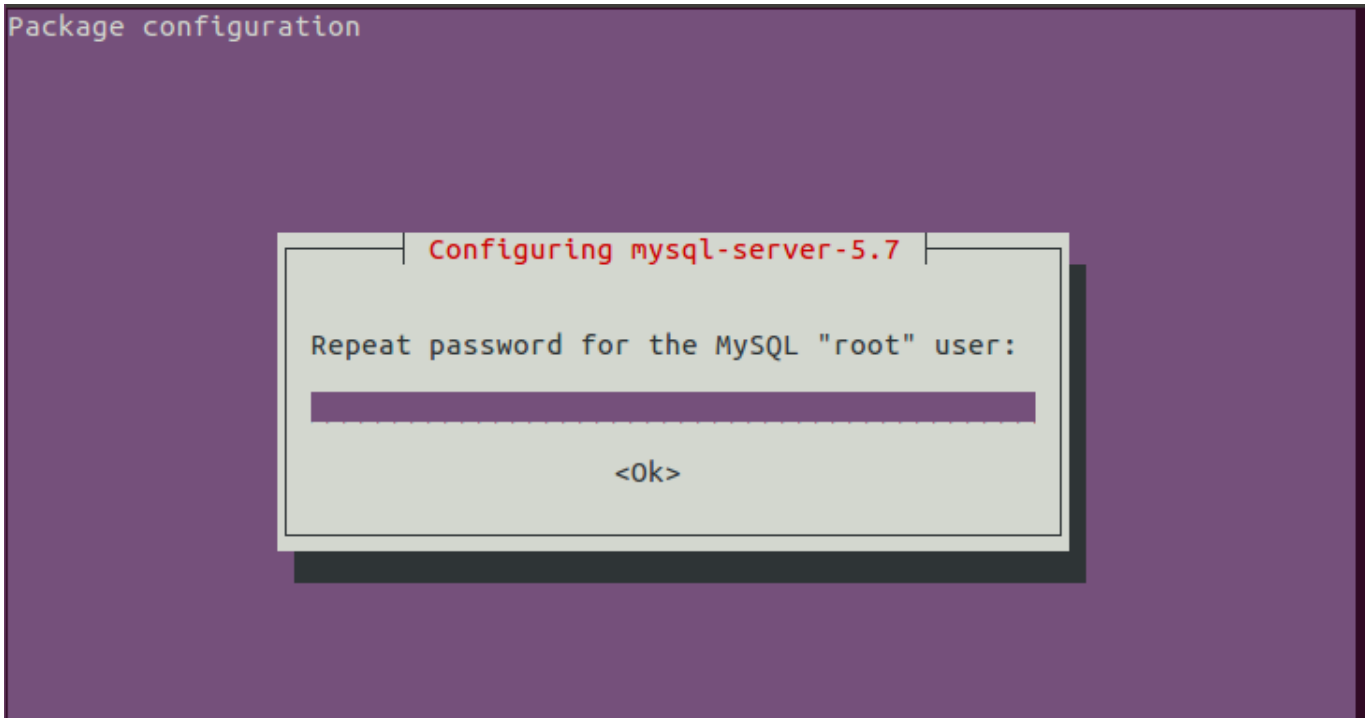

Рисунок 3.2 – Повторення пароля при кофігурації бази даних

Командою service apache2 status перевіряється статус веб-серверу. Статус active(running) свідчить про працездатність серверу. Вивод цієї команди можна побачити на рисунку 3.3:

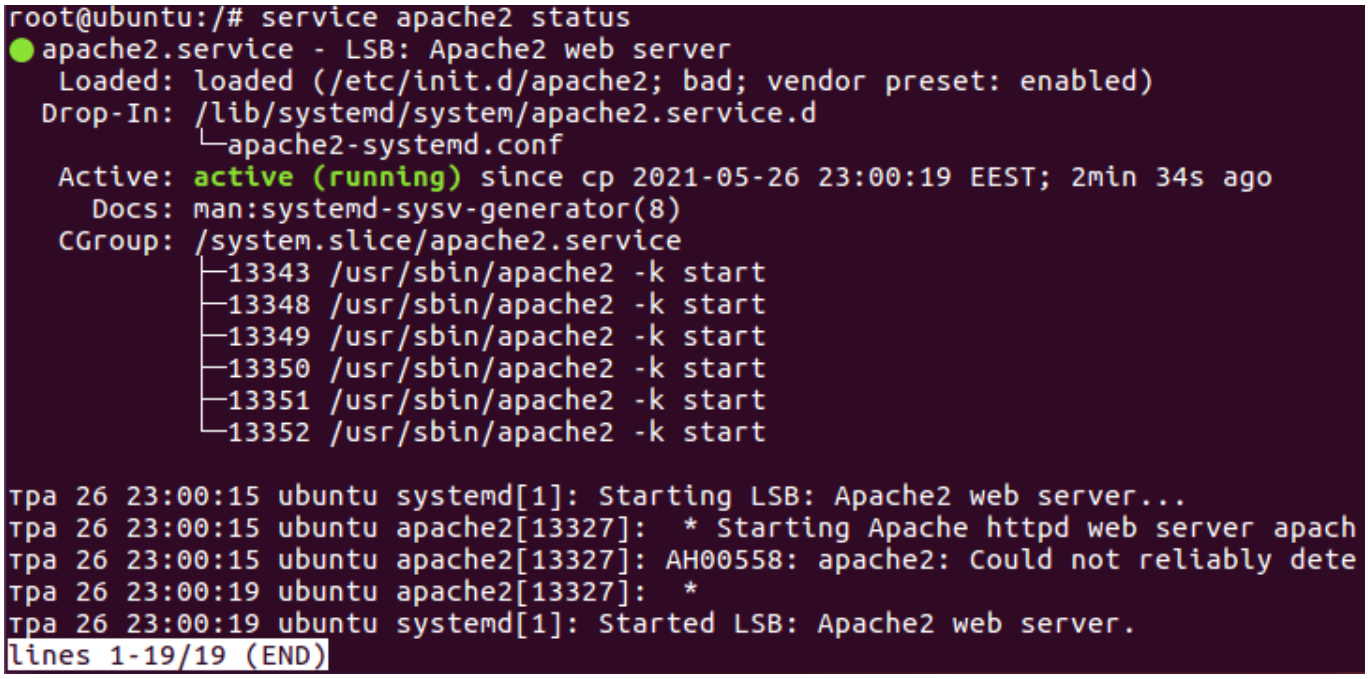

Рисунок 3.3 – Статус веб-сервера

Схожою командою service mysql status перевіряється статус бази даних. Статус active(running) свідчить про її працездатність. Вивод цієї команди можна побачити на рисунку 3.4:

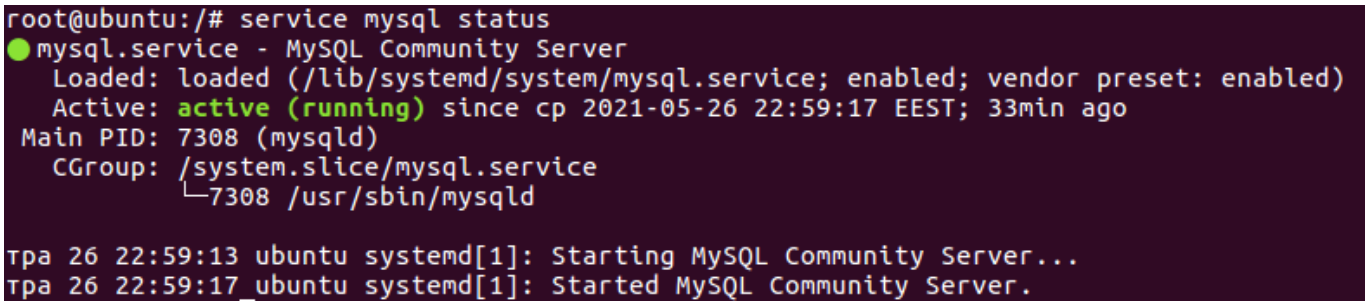

Рисунок 3.4 – Статус бази даних

Далі командою *service apache2 restart* перезавантажуємо веб-сервер. В адресній строці браузера треба прописати адресу http://localhost або http://server_ip. Початкова сторінка Apache web server буде відображена за умови правильної конфігурації при установці. Зміст цієї сторінки можна побачити на рис.3.5:

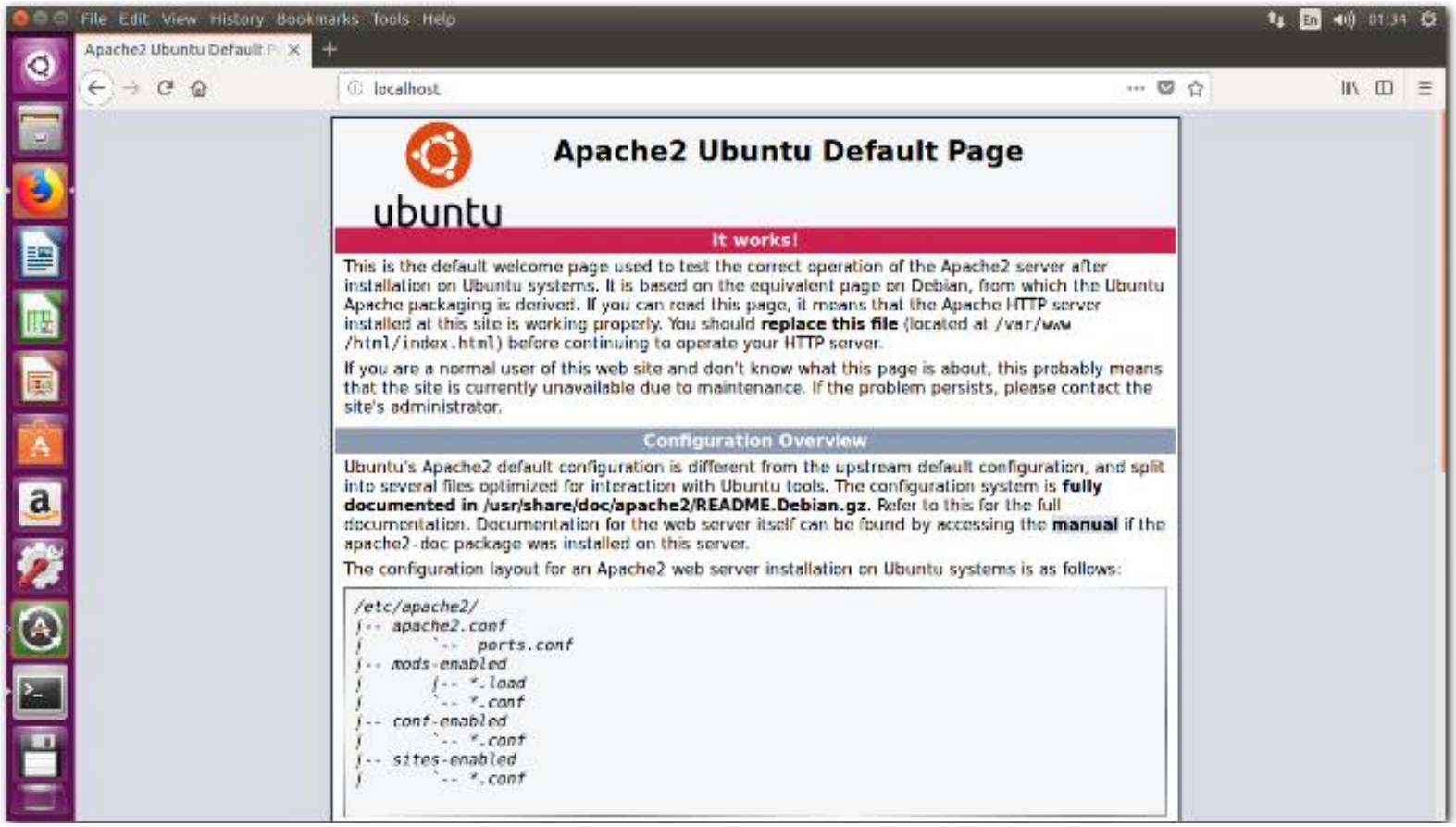

Рисунок 3.5 – Початкова сторінка веб-серверу в браузері

Зміст кофігураційного файлу зображений на рис.3.6:

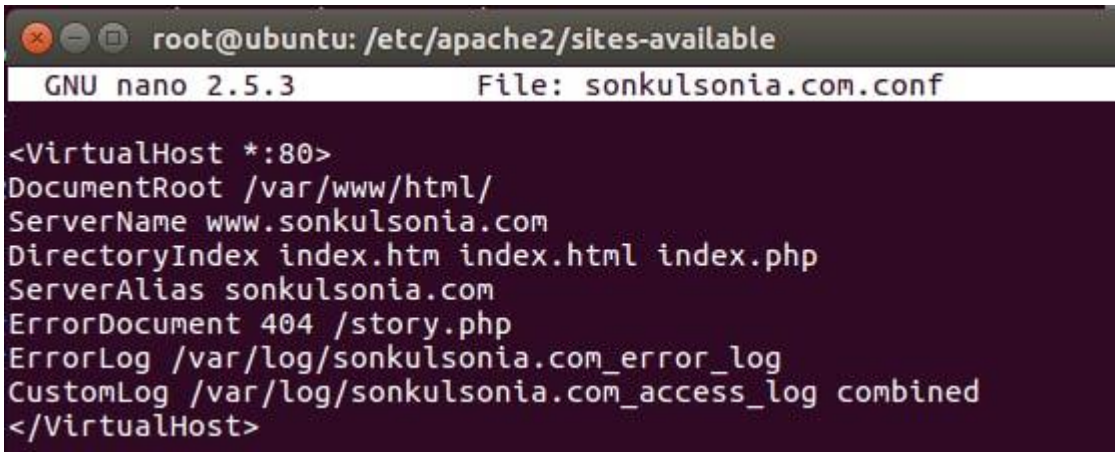

Рисунок 3.6 – Конфігураційний файл для веб-сайту sonkulsonia.com

3.2 Конфігурування потенційно захищеного LAMP-стеку

Підпис сервера (ServerSignature) - це загальнодоступна інформація вашого веб-сервера, що містить конфіденційну інформацію, яка може бути використана для експлуатації будь-якої відомої вразливості. Вимкнення підпису сервера вважається хорошою практикою безпеки, щоб уникнути розголошення версій програмного забезпечення, які ви використовуєте.

Директива ServerTokens контролює відповідь, яку надсилає сервер, включаючи деталі сервера, ОС та інші відповідні модулі.

Директива ServerTokens Prod повідомляє apache повертати лише Apache у заголовку сервера, що повертається при кожному запиті сторінки.

Тому в конфігураційному файлі веб-серверу пропишемо наступне:

ServerTokens Prod

ServerSignature Off

Використання файлів .htaccess

Загалом, файли .htaccess слід використовувати лише тоді, коли у вас немає доступу до основного файлу конфігурації сервера. Наприклад, існує поширена помилка, згідно з якою автентифікація користувача завжди повинна виконуватися у файлах .htaccess, а в останні роки - ще одна помилкова думка, що директиви mod_rewrite повинні входити у файли .htaccess. Це просто не так. Ви можете помістити конфігурації автентифікації користувачів у основну конфігурацію сервера, і це, насправді, кращий спосіб робити щось. Так само директиви mod_rewrite працюють краще, багато в чому, в конфігурації основного сервера [23].

Файли .htaccess слід використовувати в тому випадку, коли постачальники вмісту повинні вносити зміни до конфігурації сервера на основі кожного каталогу, але не мають кореневого доступу в системі серверів. Якщо адміністратор сервера не бажає часто вносити зміни в конфігурацію, може знадобитися дозволити окремим користувачам робити ці зміни у файлах .htaccess для себе. Це особливо вірно, наприклад, у випадках, коли Інтернет-провайдери розміщують кілька сайтів

користувачів на одній машині і хочуть, щоб їх користувачі могли змінювати свою конфігурацію.

Однак загалом слід уникати використання файлів .htaccess, коли це можливо. Будь-яку конфігурацію, яку ви б розглянули, додавши у файл .htaccess, можна так само ефективно зробити у розділі <Каталог> у вашому основному файлі конфігурації сервера.

Є дві основні причини уникати використання файлів .htaccess.

Перша з них - це продуктивність. Якщо для параметра AllowOverride дозволено використання файлів .htaccess, httpd буде шукати файли .htaccess у кожному каталозі. Таким чином, дозвіл .htaccess файлів спричиняє погіршення продуктивності, незалежно від того, використовуєте ви їх чи ні! Також файл .htaccess завантажується кожного разу, коли запитується документ.

Друге питання - це питання безпеки. Ви дозволяєте користувачам змінювати конфігурацію сервера, що може призвести до змін, над якими ви не маєте контролю. Ретельно обміркуйте, чи хочете ви надати своїм користувачам цю привілею. Зауважте також, що надання користувачам менших привілеїв, ніж їм потрібно, призведе до додаткових запитів на технічну підтримку. Обов'язково чітко повідомте своїм користувачам, який рівень привілеїв ви їм надали. Вказавши, на що саме ви встановили AllowOverride, і вказавши їх на відповідну документацію, ви позбавите себе багато плутанини пізніше.

В установці за замовчуванням користувачі можуть змінити конфігурацію apache за допомогою .htaccess. Якщо ви хочете зупинити користувачів від зміни налаштувань сервера Apache, можна додати AllowOverride None в налаштуваннях директорії. None відключає всі .htaccess, файли та директиви. Ця директива успадковується. Це означає, що якщо ви вказали AllowOverride none для якогось каталогу або віртуального хосту. Файли htaccess також будуть вимкнені для всіх підкаталогів.

Захист від DDOS-атак

Щодо захисту від DDOS-атак, то наступні директиви можуть допомогти вам контролювати їх:

TimeOut: Ця директива дозволяє встановити, скільки часу сервер буде чекати завершення певних подій, перш ніж він вийде з ладу. Значення за замовчуванням - 300 секунд. Добре знижувати це значення на тих сайтах, які зазнають DDOS-атак. Ця величина повністю залежить від запиту, який ви отримуєте на своєму веб-сайті. Примітка. Це може спричинити проблеми із сценаріями CGI.

MaxClients: Ця директива дозволяє встановити обмеження на з'єднання, які будуть обслуговуватися одночасно. Кожне нове підключення буде в черзі після цього обмеження. Він доступний як для Prefork, так і для Worker, як MPM. Значення за замовчуванням - 256.

KeepAliveTimeout: це кількість часу, який сервер буде чекати наступного запиту перед тим, як закрити з'єднання. Значення за замовчуванням - 5 секунд.

LimitRequestFields: Це допомагає нам встановити обмеження на кількість полів заголовка HTTP-запиту, які будуть прийняті від клієнтів. Його значення за замовчуванням - 100. Рекомендується знизити це значення, якщо DDos-атаки відбуваються внаслідок такої кількості заголовків http-запитів.

LimitRequestFieldSize: Це допомагає нам встановити обмеження розміру в заголовку HTTP-запиту.

Обмеження методів HTTP

Протокол HTTP 1.1 підтримує багато методів запиту, які можуть не знадобитися, і деякі з них мають потенційний ризик.

Зазвичай вам можуть знадобитися методи запиту GET, HEAD, POST у веб-програмі, які можна налаштувати у відповідній директиві каталогу [24].

Конфігурація підтримує за замовчуванням OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT методи у протоколі HTTP 1.1.

Щоб обмежити методи, можна додати в конфігураційний файл наступне:

<LimitExcept GET POST HEAD>

deny from all

</LimitExcept>

Вимкнення протоколу HTTP 1.0

HTTP 1.0 має слабкі місця безпеки, пов'язані з викраденням сеансу. Можна вимкнути його за допомогою модуля mod_rewrite.

Увімкніть директиву RewriteEngine наступним чином і додайте умову перезапису, щоб дозволити лише HTTP 1.1:

RewriteEngine On

RewriteCond% {THE_REQUEST}! HTTP / 1.1 $

RewriteRule. * - [F]

Для цього потрібно попередньо підключити модуль mod_rewrite, що використовує механізм перезапису на основі правил, заснований на синтаксичному аналізаторі регулярних виразів PCRE, для перезапису запитаних URL-адрес на льоту. За замовчуванням mod_rewrite відображає URL-адресу у шлях до файлової системи. Однак його також можна використовувати для перенаправлення однієї URL-адреси на іншу URL-адресу або для виклику внутрішнього отримання проксі-сервера.

Модуль mod_rewrite забезпечує гнучкий та потужний спосіб маніпулювати URL-адресами, використовуючи необмежену кількість правил. Кожне правило може мати необмежену кількість приєднаних умов правила, що дозволяє вам переписувати URL-адреси на основі змінних сервера, змінних середовища, заголовків HTTP або позначок часу. mod_rewrite працює на повному шляху URL-адреси, включаючи розділ "інформація про шлях". Правило перезапису можна викликати в httpd.conf або .htaccess. Шлях, згенерований правилом перезапису, може включати рядок запиту або може призвести до внутрішньої обробки, зовнішнього перенаправлення запиту або внутрішньої пропускної здатності проксі [25].

На рис.3.7 можна побачити процес підключення модулю до Apache:

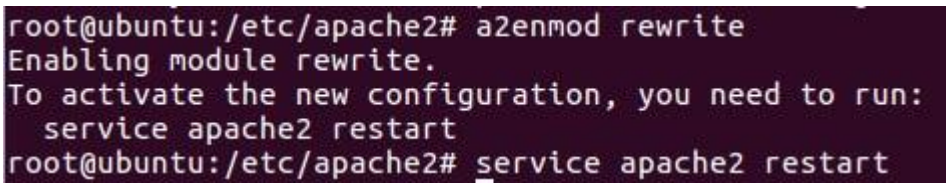

Рисунок 3.7 – Встановлення модуля rewrite

Захист від XSS

Захист від міжсайтових сценаріїв (XSS) можна обійти у багатьох браузерах. Можна додатково застосувати цей захист для веб-застосунку, якщо його вимкнув користувач на своїй стороні, за допомогою наступної директиви в конфігураційному файлі веб-сервера:

Header set X-XSS-Protection "1; mode=block"

Перед цим потрібно підключити модуль *headers,* що надає директиви для управління та модифікації заголовків запитів та відповідей HTTP. Заголовки можна об’єднати, замінити або видалити.

На рис.3.8 можна побачити процес підключення модулю до Apache:

```
root@ubuntu:/etc/apache2# a2enmod headers
Enabling module headers.
To activate the new configuration, you need to run:
  service apache2 restart
root@ubuntu:/etc/apache2# service apache2 restart
```

Рисунок 3.8 – Встановлення модуля headers

Можна пом’якшити більшість поширених атак між сценаріями між сайтами, використовуючи HttpOnly та Secure flag у файлі cookie. Не маючи HttpOnly та Secure, можна вкрасти сеанс веб-додатків та файли cookie або маніпулювати ними, і це небезпечно. Тому наступна директива допоможе в захисті:

Header edit Set-Cookie ^(.)$ $1;HttpOnly;Secure*

Приховування директорій

Якщо на сервері зберігається певна інформація, що не має бути доступною пересічному користувачу, потрібно заборонити доступ до неї. Один із можливих методів описаний нижче.

Для прикладу створимо нову папку з файлами в конфігураційній папці веб-застосунку, як це показано на рисунку нижче:

```
root@ubuntu:/etc/apache2# cd /var/www/html
root@ubuntu:/var/www/html# ls
index.html
root@ubuntu:/var/www/html# mkdir test
root@ubuntu:/var/www/html# cd test
root@ubuntu:/var/www/html/test# touch hi
root@ubuntu:/var/www/html/test# touch hello
```

Рисунок 3.9 – Створення нової директорії з файлами

Створена директорія відображається на веб-сторінці в наступному вигляді:

Рисунок 3.10 – Відображення змісту директорії test на веб-сторінці

В конфігураційному файлі веб-серверу пропишемо наступне:

<Directory /var/www/html/test> - шлях до створеної папки

Options -Indexes

</Directory>

Або

<Directory /var/www/html/test >

Options None

</Directory>

Після перезапуску веб-сервера сторінка буде виглядати наступним чином:

Рисунок 3.11 – Відображення змісту директорії test на веб-сторінці після налаштувань безпеки

Захист за допомогою ModSecurity

ModSecurity, який іноді називають Modsec, - це брандмауер веб-додатків з відкритим кодом (WAF). Спочатку розроблений як модуль для HTTP-сервера Apache, він еволюціонував, щоб забезпечити масив запитів протоколу Hypertext і можливості фільтрації відповідей, а також інші функції безпеки на ряді різних платформ, включаючи Apache HTTP Server, Microsoft IIS та Nginx. Це безкоштовне програмне забезпечення, випущене за ліцензією Apache 2.0.

Платформа надає мову конфігурації правил, відому як "SecRules", для моніторингу в реальному часі, реєстрації та фільтрації зв'язку протоколу передачі гіпертексту на основі визначених користувачем правил.

Хоча це не єдина конфігурація, ModSecurity найчастіше розгортається для забезпечення захисту від загальних класів вразливостей за допомогою OWASP ModSecurity Core Rule Set (CRS). Це набір правил з відкритим кодом, написаний мовою SecRules ModSecurity. Проект є частиною OWASP, Проекту безпеки відкритих веб-додатків. Також доступні кілька інших наборів правил.

Для виявлення загроз модуль ModSecurity розгортається вбудованим у веб-сервер або як проксі-сервер перед веб-програмою. Це дозволяє двигуну сканувати вхідні та вихідні комунікації HTTP до кінцевої точки. Залежно від конфігурації правила, механізм вирішить, як слід обробляти комунікації, що включає можливість передачі, скидання, перенаправлення, повернення заданого коду стану, виконання скрипту користувача тощо [26].

На рисунку 3.12 зображений процес інсталяції ModSecurity, як додаткового модулю до веб-серверу:

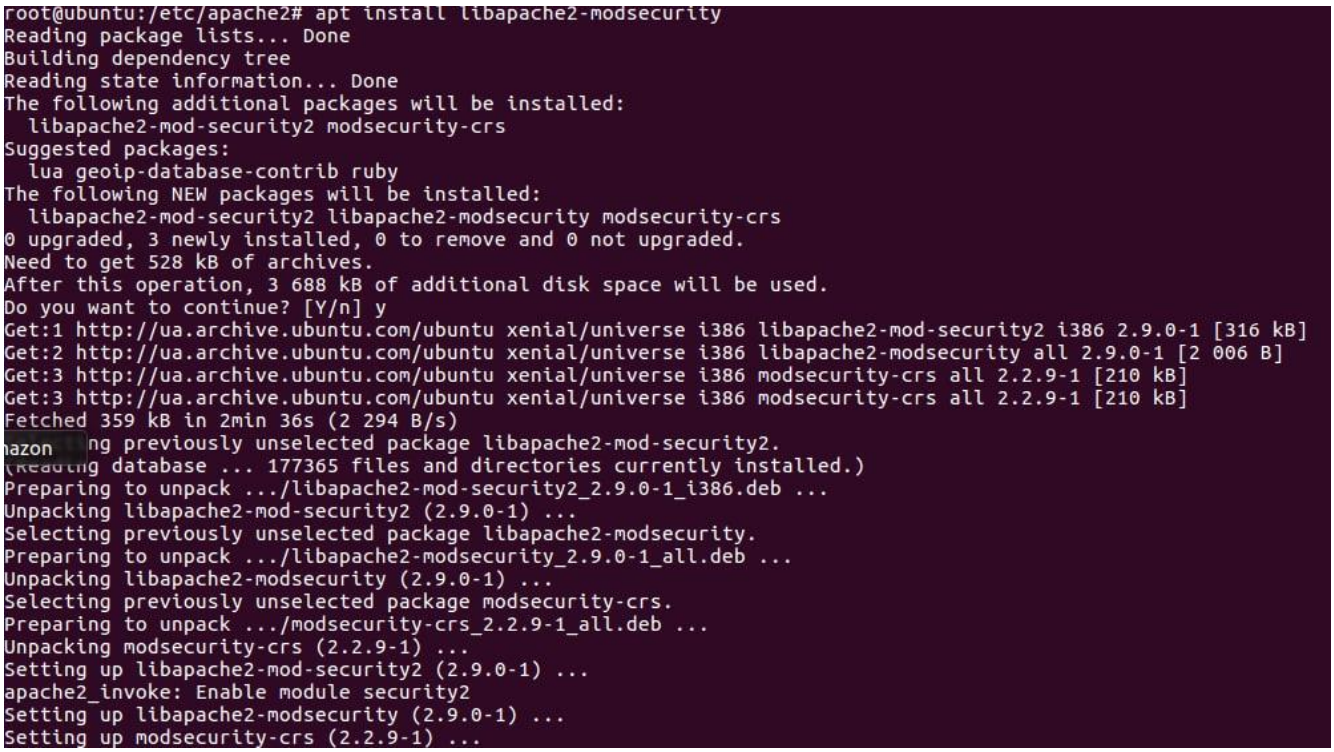

Рисунок 3.12 – Інсталяція modsecurity

ModSecurity є доволі гнучким в створенні правил безпеки, тому адміністратори веб-серверів можуть легко створити правила відповідно до їх мети. Далі показаний приклад (рис.3.13) створення правила, що буде забороняти відправлення HTTP методу POST, якщо в ньому містяться слова “attack”, “spam”, “threat”. Можна здогадатися, що це правило націлено на захист від спаму шляхом фільтрації введених користувачами даних. Результат дії правила зображений на рис. 3.14-3.15:

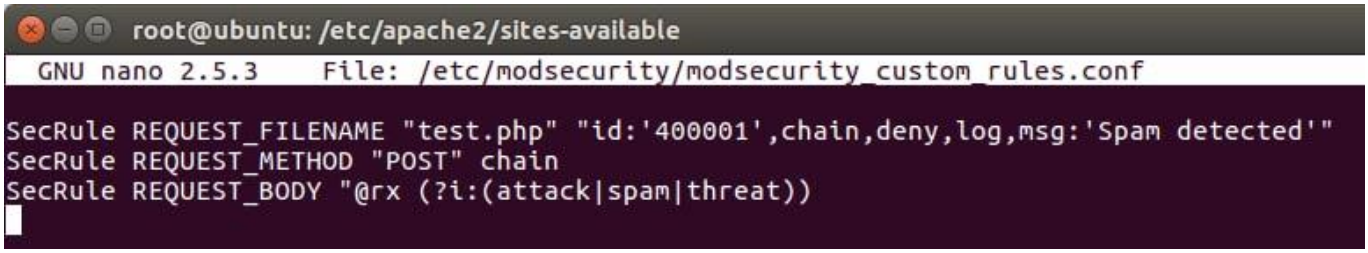

Рисунок 3.13 – Приклад створення власного правила modsecurity

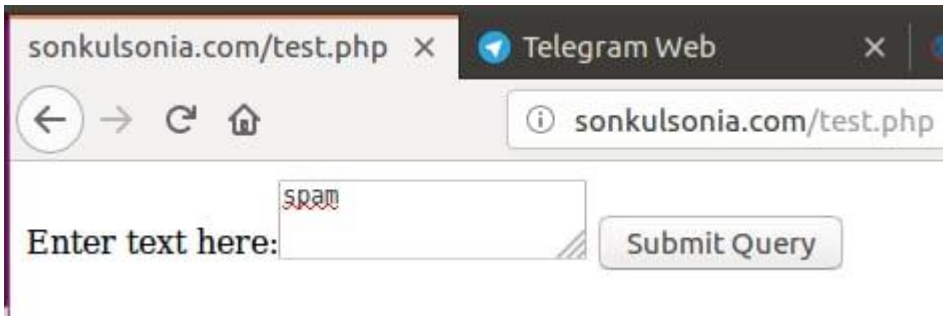

Рисунок 3.14 – Сторінка з можливістю введення даних користувачем

Рисунок 3.15 – Результат дії правила modsecurity

Захист передачі даних за допомогою TLS/SSL

Згенерувати відкритий і закритий ключ для організації шифрування трафіку можна за допомогою утиліти OpenSSL. Вона також створює запит на підпис сертифікату, що може бути підписаний CA (Certification Authority).

OpenSSL - це інструмент командного рядка з відкритим кодом, який зазвичай використовується для створення приватних ключів, створення CSR, встановлення сертифіката SSL / TLS та визначення інформації про сертифікат.

Першим кроком до отримання сертифіката SSL є використання OpenSSL для створення запиту на підпис сертифіката (CSR), який можна надіслати до Центру сертифікації (CA) (наприклад, DigiCert). CSR містить загальні імена, для яких потрібно захистити ваш сертифікат, інформацію про вашу компанію та ваш відкритий ключ. Для того, щоб створити CSR, він повинен мати приватний ключ, з якого витягується відкритий ключ. Це можна зробити, використовуючи існуючий закритий ключ або генеруючи новий закритий ключ.

Команда для генерації ключів та запиту на підпис сертифіката зображена на рис.3.16:

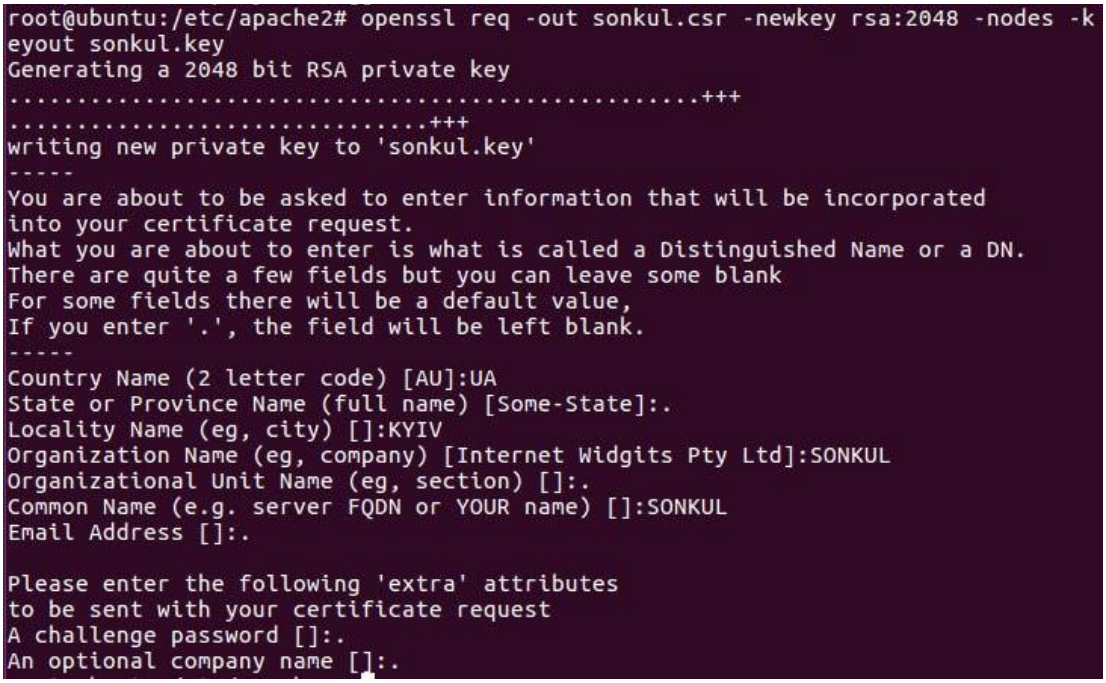

Рисунок 3.16 – Процес створення запиту на сертифікат

Після виконання команди буде створено 2 файли: ключ та запит на підпис. Зміст файлу з ключем зображено на рис.3.17, а зміст файлу з запитом на підпис зображено на рис.21:

```
root@ubuntu:/etc/apache2# cat sonkul.key
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDT8uPY9B4htjHr
hUOFjsF1/TXyjW5/50evWb++7XZN2mwxjEtLgtWuPcIrA6EsfV1AFnAi50PNbwBx
pvgHg1qpBquJuesugGJyS2D8Nw0727vBQAy3hU/3Q3bfBrG3Jk82mkQ4jY60UHkl
rwaBOFrdr7I2f+c4qFXp05zg2Uz7Jizf4gwkPMl9pcy5altCxmmbwvk6P0NTV+Kr
EALn46saWi1wMmCfh1aZ0MBBWLR0O5aQ2Zm8Rs86UFs9ctzfb6mgU2O44lB8V0oh
YohruZHQburDGQggjtt2pMuu+go29Pbakz8Y+f2tFqbl2VDpT2ZNyvqea9s1ZB0z
UluaDiD3AgMBAAECggEBAJF1AzvUpY07Td02ExXo2AqSoWB6AdjTsBW+gWKV83gQ
q+oBP3duLhmrlsMQdZ9j1ghynbjGgHjPcj2NenEli6WuxzD5IFdEksufwcq2OWXY
Mm24+EFGw4+NhFohbPul1f1COAmb1NEY3VuoV819EJdNyf+vHc97b6LOsL57Q2Kk
GtvDujQiI+eCFk6/QgJhxi+X16EqJnW7nCfZrZFUcwH4yivkos9Kh049k9g0Tv8N
hbrFHaaON+2iuezF5NSsnEEMH57qsMyEvoVtCh3BRRRNsv2djG1qSmlsORTNUEz+
vO0yq1YjcIzhpwa0E9RMMG9KZQm2+KAEJBREg7szV+kCgYEA+oOT8q2cGWM7sont
QRq3t34T5JT/ng3Ry2ftR0dWH6XyVWVLTKutBog6k3sLshyZMxXG+eSwYrCDMTj+
v/29MTQbTkEqlZHPVG3oSRqgvMyANW7SppEc+af1z1x1wC6W085vRm6grcFvMDcU
l6jGGJvs2nhqEr1yINZMTEU9TGUCgYEA2JccAZHscJiaBz0LF1XcTB9eV1p0NiPP
lYDzFaGrHJlOhTHQGJ1S+y66stz8ya2snjUB8ezo6ZaB6lWwHMDQKMMkXo5Irdi5
W9EQ5Q6zMdEeCiWGbj9fzk8TVEQjFBEmYt0+DccDmHGO6OL3o7HzdQFcPKQXdJyB
SC52HUQ2XCsCgYBpBsj8MTUEn8njbFf/t4fWchE9AAq65OhT8tpTy+CbnCMhEjqj
hkYgOvTz595bagoAlNy639njRkV6WX7UVzj+a+4WvNNxqZPWvc1o+LTy8nm97Rx7
oNMZCAXSTd1AGUg6vI2CZdcPgc1v1WfYvXVv089VOK6f9LTod2+9ejnc0QKBgQCL
2E156gk6zh0trUlbANIjyGE2j63XQxoGEuNhRShhZ88eCwKs0e/BRPPBBxSk92Es
KVefJ/Ne1xH9BuSWPBxrJUx3TPE6ZmOQGBYLf4lOEhKORJpe1sZ6Q+5cCz6G92fM
j8NATA5TQIcgcWOpqv1hv8YZVW3bmi2TOISE8wUSTwKBgFpOEhDebyuXkp2tWFHZ
z9jhf/6zohR0FlMMa12nLTxoKDvxzQD5jAmJEofs6OLskokp/CPaGWBxbXgS8mFC
mAOTyDm3sxS7XVPwTyZg5CJ+cJW896SP8S4AUkFADAudX+SJtOAMXLN0qfESJDNx
w5k0p1zWw9KbXV8ZRCqsGmgL
-----END PRIVATE KEY-----
```

Рисунок 3.17 – Зміст файлу з приватним ключем

```
root@ubuntu:/etc/apache2# cat sonkul.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICgzCCAWsCAQAwPjELMAkGA1UEBhMCVUExDTALBgNVBAcMBEtZSVYxDzANBgNV
BAoMBlNPTktVTDEPMA0GA1UEAwwGU09OS1VMMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEA0/Lj2PQeIbYx64VDhY7Bdf018o1uf+dHr1m/vu12TdpsMYxL
S4LVrj3CKwOhLH1dQBZwIudDzW8Acab4B4NaqQaribnrLoBicktg/DcNO9u7wUAM
t4VP90N23waxtyZPNppEOI2OtFB5Ja8GgTha3a+yNn/nOKhV6dOc4NlM+yYs3+IM
JDzJfaXMuWpbQsZpm8L5Oj9DU1fiqxAC5+OrGlotcDJgn4dWmdDAQVi0dDuWkNmZ
vEbPOlBbPXLc32+poFNjuOJQfFdKIWKIa7mR0G7qwxkIII7bdqTLrvoKNvT22pM/
GPn9rRam5dlQ6U9mTcr6nmvbNWQdM1Jbmg4g9wIDAQABoAAwDQYJKoZIhvcNAQEL
BQADggEBAKVulH3Vw2qWjN4ImZndit5xFi22+p8VsvHn+a3TArT0cA7qw+TIrI5E
qycgU3RnL5hiW+/IOKg5+FpQZzJhCFCsnkVTkfDARsBkMBrZy7b+Jq7SFENRLE8s
ZpWJJNYf6KdTxakK8+3Smoedef3c7H1e0mUpgVXdxT7XDW6/1WX894My8t3WQEGb
BVylFOtZeCCQ6qjpZO6bbPeKek9c/8fyLoOb+dGaL4FPVHwBnSed51bD2uqn5qhG
/axW7hU3FmubwQngSOcntV1MOU8Uiji0+Yxb8EJ7nIMUu4KNDgDICnDx1+J+F3XL
8tMjLs/+RjyPJ7Wm/VB6PuW7ImOluhU=
-----END CERTIFICATE REQUEST-----
```

Рисунок 3.18 – Зміст файлу з запитом на сертифікат

З файлу ключа можна отримати публічний та приватний ключі, необхідні для асиметричного шифрування, завдяки якому суб'єкти здійснюють обмін секретним ключем, яким буде відбуватися шифрування трафіку. Процес отримання публічного ключа та власне сам ключ зображені на рис.3.19:

```
root@ubuntu:/etc/apache2# openssl rsa -in sonkul.key -pubout -out sonkul_public.
key
writing RSA key
root@ubuntu:/etc/apache2# cat sonkul_public.key
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0/Lj2PQeIbYx64VDhY7B
df018o1uf+dHr1m/vu12TdpsMYxLS4LVrj3CKwOhLH1dQBZwIudDzW8Acab4B4Na
qQaribnrLoBicktg/DcNO9u7wUAMt4VP90N23waxtyZPNppEOI2OtFB5Ja8GgTha
3a+yNn/nOKhV6dOc4NlM+yYs3+IMJDzJfaXMuWpbQsZpm8L5Oj9DU1fiqxAC5+Or
GlotcDJgn4dWmdDAQVi0dDuWkNmZvEbPOlBbPXLc32+poFNjuOJQfFdKIWKIa7mR
0G7qwxkIII7bdqTLrvoKNvT22pM/GPn9rRam5dlQ6U9mTcr6nmvbNWQdM1Jbmg4g
9wIDAQAB
-----END PUBLIC KEY-----
```

Рисунок 3.19 – Публічний ключ

Для посилення захисту від ескалації привілеїв потрібно запускати веб-сервер від імені користувача з найменшими привілеями. Для цього потрібно створити нового користувача, який буде адмініструвати виключно веб-сервер та не матиме root привілеїв на сервері (рис.3.20):

```
root@ubuntu:/etc/apache2# adduser apacheadmin
adduser: The user `apacheadmin' already exists.
root@ubuntu:/etc/apache2# adduser servadm
Adding user `servadm' ...
Adding new group `servadm' (1003) ...
Adding new user `servadm' (1003) with group `servadm' ...
Creating home directory `/home/servadm' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for servadm
Enter the new value, or press ENTER for the default
        Full Name []: Apache admin
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

Рисунок 3.20 – Створення нового користувача

В кофігураційний файл потрібно додати:

User servadm

Та змінити власника для папки веб-серверу командою в терміналі:

chown servadm –R /etc/apache2

Запустити службу веб-серверу від імені нового користувача. Процеси щодо веб-серверу будуть виглядати наступним чином:

```
servadm@ubuntu:/etc/apache2$ service apache2 stop
servadm@ubuntu:/etc/apache2$ service apache2 start
servadm@ubuntu:/etc/apache2$ ps aux |grep apache
root      4194  1.0  1.2 128152 26652 ?        Ss   15:30   0:00 /usr/sbin/apache2 -k start
servadm   4197  0.0  0.4 128184  8424 ?        S    15:30   0:00 /usr/sbin/apache2 -k start
servadm   4198  0.0  0.4 128184  8424 ?        S    15:30   0:00 /usr/sbin/apache2 -k start
servadm   4199  0.0  0.4 128184  8424 ?        S    15:30   0:00 /usr/sbin/apache2 -k start
servadm   4200  0.0  0.4 128184  8424 ?        S    15:30   0:00 /usr/sbin/apache2 -k start
servadm   4201  0.0  0.4 128184  8424 ?        S    15:30   0:00 /usr/sbin/apache2 -k start
```

Рисунок 3.21 – Процеси адміністратора веб-серверу

Висновки за розділом 3

В третьому розділі дипломної роботи було створено потенційно захищену серверну частину веб-застосунку на базі LAMP-стеку, конфігурацію якого було модифіковано згідно з побудованою у другому розділі моделлю загроз для попередження деяких загроз веб-серверу. До конфігураційного файлу веб-серверу додано директиви, що підвищують його захищеність, а також змінено деякі, наявні за замовчуванням, директиви з цією ж метою. Також створено приватний та публічний ключі, що необхідні для асиметричного шифрування TLS/SSL, що дозволяє обмін секретним ключем, який буде використаний для шифрування трафіку.

ВИСНОВКИ

У дипломній роботі розв’язано актуальні завдання, що стосуються захисту серверної частини веб-застосунку. Метою даної роботи було створення потенційно захищеної серверної частини веб-застосунку шляхом поєднання запропонованих у дипломній роботі механізмів захисту від кіберзагроз.

Для досягнення поставленої мети було виконано поставлені завдання та отримано наступні наукові та практичні результати:

1. проаналізовано клієнт-серверну архітектуру, що є основою взаємодії суб’єктів у веб-просторі. Користувач веб-застосунку є клієнтом, відповідно до даної архітектури, і звертається до серверної частини веб-застосунку для виконання поставлених ним задач. Сервер же обробляє запити клієнта та надсилає йому дані відповідно до запитів.
2. Розглянуто загальну архітектуру веб-застосунків та принципи їхнього функціонування.
3. Досліджено нормативно-правову базу, що стосується захисту веб-застосунків та захисту інформації в цілому.
4. Проаналізовано основні загрози серверній частині веб-застосунку, такі як атаки, націлені на викрадення сессії автентифікованого на сервері користувача, подальшу ескалацію привілеїв та на відмову сервера в обслуговуванні користувачів.
5. Досліджено основні механізми захисту серверної частини веб-застосунку від зазначених загроз.
6. Побудовано модель загроз веб-застосунку, з врахуванням розглянутих загроз.
7. Сконфігуровано потенційно захищену серверну частину веб-застосунку на основі LAMP-стеку шляхом поєднання та практичного використання деяких зазначених механізмів захисту від проаналізованих загроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. K.Rajiv, A.Prashanthi, Ch.Bharadwaja: Web Server Security [Електронний ресурс]. – Режим доступу: https://www.ijert.org/research/web-server-security-IJERTV1IS8098.pdf

2. Кулішенко С.А, Пархоменко І.І. «Загрози безпеці веб-ресурсів та способи захисту від них», Науково-практична конференція “Information Technology and Interactions (IT&I)”, 2018

3. Кулішенко С.А, Пархоменко І.І. «Способи реалізації безпечної клієнт-серверної взаємодії у веб-просторі», Науково-практична конференція “Information Technology and Interactions (IT&I)”, 2019

4. Ubaid Pisuwala. Fundamentals of web application architecture [Електронний ресурс]. – Режим доступу: https://www.peerbits.com/blog/web-application-architecture.html

5. What are the different types of web apps [Електронний ресурс]. – Режим доступу: https://hellodarwin.com/blog/different-web-apps

6. Верховна Рада України Про прийняття та скасування національних стандартів, прийняття поправок до національних стандартів [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0312774-19#Text

7. ISO/IEC 27001 [Електронний ресурс]. – Режим доступу: https://www.iso27001security.com/html/27001.html

8. ISO/IEC 27034 [Електронний ресурс]. – Режим доступу: https://www.iso27001security.com/html/27034.html

9. ISO/IEC 27005 [Електронний ресурс]. – Режим доступу: https://www.iso27001security.com/html/27005.html

10. ISO/IEC 27002 [Електронний ресурс]. – Режим доступу: https://www.iso27001security.com/html/27002.html

11. Client-Server Architecture [Електронний ресурс]. – Режим доступу: https://teachcomputerscience.com/client-server-architecture/

12. Morey J. Haber Privilege Escalation Attack and Defense Explained [Електронний ресурс]. – Режим доступу: https://www.beyondtrust.com/blog/entry/privilege-escalation-attack-defense-explained

13. Zbigniew Banach What Is Session Hijacking: Your Quick Guide to Session Hijacking Attacks [Електронний ресурс]. – Режим доступу: https://www.netsparker.com/blog/web-security/session-hijacking/#:~:text=Session%20hijacking%20is%20an%20attack,ends%20when%20you%20log%20out

14. PHP - Introduction [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/php/php_introduction.htm

15. What is MySQL? Everything You Need to Know [Електронний ресурс]. – Режим доступу: https://www.talend.com/resources/what-is-mysql/

16. David Balaban Are you Ready for These 26 Different Types of DDoS Attacks? [Електронний ресурс]. – Режим доступу: https://www.securitymagazine.com/articles/92327-are-you-ready-for-these-26-different-types-of-ddos-attacks

17. Jerry Louis DETECTION OF SESSION HIJACKING, University of Bedfordshire, 77 с., 2011.

18. The Annual Microsoft Vulnerabilities Report 2021 [Електронний ресурс]. – Режим доступу: https://www.beyondtrust.com/resources/whitepapers/microsoft-vulnerability-report

19. S. Tang, N. Dautenhahn and S.T. King, Fortifying web-based applications automatically, Conference on Computer and Communications Security, 2011, с. 615–626.

20. Jeff Jancula, Security Problems With Web Servers' Session Tracking Mechanisms [Електронний ресурс]. – Режим доступу: http://archives.neohapsis.com/archives/vuln-dev/2001-q3/0548.html

21. CVE, Apache : Security Vulnerabilities [Електронний ресурс]. – Режим доступу: https://www.cvedetails.com/vulnerability-list.php?vendor_id=45&product_id=&version_id=&page=1&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&c

vssscoremin=4&cvssscoremax=4.99&year=0&month=0&cweid=0&order=4&trc=77&sha=ce15f0da4b17e99a6d5893cfdd4a8721eba4ffea

22. Frank Kargl, Joern Maier, Protecting Web Servers from Distributed Denial of Service Attacks [Електронний ресурс]. – Режим доступу: https://www.princeton.edu/~rblee/ELE572Papers/p514-kargl.pdf

23. Sivakorn, Suphannee, Iasonas Polakis, and Angelos D. Keromytis. "The cracked cookie jar: HTTP cookie hijacking and the exposure of private information." Security and Privacy (SP), IEEE Symposium on. IEEE, 2016.

24. M. Kolsek. Session Fixation Vulnerability in Web-based Applications. [Електронний ресурс]. – Режим доступу: http://www.acrossecurity.com/papers/session_fixation.pdf

25. Bharti Nagpal,Naresh Chauhan, Preventive Measures for Securing Web Applications Using Broken Authentication and Session Management Attacks: A Study, 15 c., 2019

26. OpenSSL Quick Reference Guide [Електронний ресурс]. – Режим доступу: https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm