МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

Розробка комплексної системи захисту інформації веб-ресурсу

Здобувача вищої освіти першого (бакалаврського) рівня,
4 курсу, групи КБ-19-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»

_____ Костенко Дмитро Максимович
(підпис) (прізвище, ім'я, по батькові)

Науковий керівник _____ Омельчук Антон Анатолійович, к.т.н.
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми _____ Федотова - Півень Ірина Миколаївна, к.т.н, доцент
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри
протокол від «26» травня 2023 № 13

Завідувач кафедри _____ Філоненко Михайло Миколайович, к.ф.-м.н., доцент
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2023

Реєстрація № 25 06.06.2023 _____
(дата) (підпис відповідальної особи)

Результат перевірки до захисту 29.05.2023 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 14.06.2023 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ В.М. Франчук
(підпис) (ініціали, прізвище)

Члени комісії:

_____ М.М. Філоненко
(підпис) (ініціали, прізвище)

_____ О.В. Редич
(підпис) (ініціали, прізвище)

_____ Є.В. Котух
(підпис) (ініціали, прізвище)

БЛАНК ОЦІНЮВАННЯ ДИПЛОМНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	28		14.06.2023 Франчук В.М. Філоненко М.М Редич О.В. Котух Є.В. Суховецька Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	45		

Оцінювання кваліфікаційної роботи науковим керівником - 45 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалоюЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
90	A	5 (відмінно)	14.06.2023	Франчук В.М. Філоненко М.М Редич О.В. Котух Є.В. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

4

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую
Завідувач кафедри
_____ Філоненко М.М.
«16» 11 2022

Завдання на кваліфікаційну роботу
здобувачу вищої освіти

Костенко Дмитру Максимовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Розробка комплексної системи захисту інформації веб-ресурсу
Затверджена наказом ректора від «15» листопада 2022 № 1205 (зі змінами наказ від «04» квітня 2023 № 361)
2. Строк здачі закінченої роботи «26» травня 2023
3. Мета роботи
Розробка комплексної системи захисту інформації веб-ресурсу
Об'єкт дослідження:
Методи та засоби разробки захисту інформації веб-ресурсу
Предмет дослідження:
Тестовий сайт
База дослідження:
Ресурси для разробки систем захисту інформації веб-ресурсу
4. Вихідні дані до роботи:
Текстовий контент, веб-ресурси, зображання та відео
5. Перелік графічного матеріалу:
13 рисунків та 6 таблиць.
6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом): ВСТУП, РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ
1.1 Поняття про інформаційну безпеку, 1.2 Характеристики Веб-ресурсів, РОЗДІЛ 2 МЕТОДИ ТА АЛГОРИТМИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ, 2.1 Існуючі методи захисту веб-ресурсів, 2.2 Аналітичний огляд розподілених атак на відмову, 2.3 Алгоритм захисту від DDoS атак заснований на IP фільтрування, РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦЯ, 3.1 Програмні засоби для розробки, 3.2 Конфігурації веб-сервера для ефективного протидії DDoS атакам, 3.3 Короткий опис програмної реалізації, 3.4 Тестування захисту від DDoS атак, ВИСНОВОК,

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів робот	
		За планом	Фактич
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.11.2022	15.11.202
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	01.02.2023	03.02.202
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	03.04.2023	08.04.202
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	10.05.2023	15.05.202
5	**Висновки**	15.05.2022	15.05.202
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	26.05.2023	26.05.202
7	Отримання відгуку керівника	06.06.2023	06.06.202
8	Отримання зовнішньої рецензії	08.06.2023	08.06.202

8. Дата видачі завдання «16» листопада 2022

9. Керівник випускної кваліфікаційної роботи _____ (підпис) Омельчук А.А. (прізвище, ініціали)

10. Гарант освітньо-професійної програми _____ (підпис) Федотова-Півен (прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти _____ (підпис) Костенко Д.М. (прізвище, ініціали)

АНОТАЦІЯ

Розробка комплексної системи захисту інформації веб-ресурсу

Мета роботи полягає у створенні тестового сайту і розробці комплексної системи захисту інформації веб-ресурсу.

Завдання роботи включає аналіз загроз, визначення методів та засобів захисту інформації веб-ресурсу, проектування комплексного захисту інформації веб-ресурсу.

Предметом дослідження є тестовий сайт

Об'єктом дослідження є методи та засоби разробки захисту інформації веб-ресурсу

Базою дослідження є ресурси для разробки систем захисту інформації веб-ресурсу.

За результатами дослідження сформульовані ефективні конфігурації сервера та додаткові правила використання які можуть значно підвищити ефективність протидії системи DDoS атакам.

Одержані результати можуть бути використані для реалізації системи комплексного захисту від DDoS атак з використанням міжмережевого екрана для фільтрації небажаного трафіка, та додаткових методів захисту від DDoS атак.

Кваліфікаційна робота містить 72 сторінок, 11 рисунків та 6 таблиць, список

використаних джерел з 29 найменувань.

Ключові слова: веб-ресурс, загроза, атака, захист.

Рік виконання кваліфікаційної роботи 2023 р.

Рік захисту роботи 2023 р.

ABSTRACT

Development of a comprehensive web resource information protection system

The purpose of the work is to create a test site and develop a comprehensive system for protecting web resource information.

The task of the work includes analysis of threats, definition of methods and means of protection of information of web resource, design of complex protection of information of web resource.

The subject of the study is a test site

The object of the study is methods and means of developing the protection of web resource information

The basis of the study is resources for developing web resource information protection systems.

According to the results of the study, effective server configurations and additional usage rules have been formulated that can significantly increase the efficiency of the DDoS system counteraction to attacks.

The results can be used to implement a system of comprehensive protection against DDoS attacks using the firewall to filter unwanted traffic, and additional methods of protection against DDoS attacks.

Qualification work contains 72 pages, 11 figures and 6 tables, list

used sources from 29 names.

Keywords: web resource, threat, attack, protection.

Qualification Work Year 2023

Year of work protection 2023

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1 Поняття про інформаційну безпеку 9
1.2 Характеристики Веб-ресурсів 20
РОЗДІЛ 2 МЕТОДИ ТА АЛГОРИТМИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ 27
2.1 Існуючі методи захисту веб-ресурсів 27
2.2 Аналітичний огляд розподілених атак на відмову 42
2.3 Алгоритм захисту від DDoS атак заснований на IP фільтрування 48
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦЯ 55
3.1 Програмні засоби для розробки 55
3.2 Конфігурації веб-сервера для ефективного протидії DDoS атакам 56
3.3 Короткий опис програмної реалізації 59
3.4 Тестування захисту від DDoS атак 63
ВИСНОВОК 67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 68

ВСТУП

Популярність інтернет-ресурсів значно зросла за останні роки. Додатки, засновані на веб-технологіях, стали затребувані у вирішенні завдань з різних галузей і поступово стали витісняти додатки, засновані на інших технологіях.

Це призвело до ускладнення веб-додатків в плані структури, архітектури та реалізації, до того ж у веб-додатках стала використовуватися розподілена архітектура. Таке ускладнення висунуло нові вимоги до питань безпеки веб-додатків. Відомо, що безпека — це ключовий момент розробки веб-сервісів і розробники повинні приділяти достатньо уваги цьому питанню. Якщо заходи безпеки будуть закладені в додаток на ранньому етапі розробки, то це значно заощадить час і кошти, в порівнянні з варіантом, коли безпека буде додаватися в готовий або практично готовий додаток. Питання безпеки і надійності є актуальними на всіх етапах розробки. І хоча з цього приводу було написано безліч документів і статей, але мало що змінилося - програмісти продовжують писати програми, враховуючи питання безпеки не в повній мірі.

Інтернет тепер є частиною життя кожної людини, а також основним засобом доступу до багатьох корисних послуг із суворими вимогами безпеки. В результаті чого уразливості у веб-просторі можуть призвести до різних атак з катастрофічними наслідками: від економічних втрат, наприклад, в разі нападів на платіжні системи, такі як PayPal, і закінчуючи порушеннями конфіденційності, наприклад, в разі розкриття і підроблення електронних записів. Через зростаючий попит споживачів розробка веб-додатків стає все більш складною. Багато мов програмування разом із складною багаторівневою архітектурою, яка зазвичай бере участь у розробці веб-додатків, сприяють ймовірності помилок програмування. Такі помилки можуть призвести до серйозних вразливостей у системі безпеки, які потім можуть бути використані зловмисниками.

Захист онлайн-сервісів - це досить трудомісткий процес, враховуючи внутрішню складність Інтернету. Веб-екосистема є різнорідною і включає в себе велику кількість різних компонентів і технологій, тому шляхи атаки на веб-додатки неймовірно великі, наприклад, до них можна віднести: недоліки безпеки в веб-браузері можуть розкривати аутентифікаційні та конфіденційні дані, що зберігаються на сторінках; уразливості веб-протоколів можуть порушити захищеність і цілісність сеансу зв'язку; помилки в коді веб-додатки можуть привести до вбудовування шкідливого вмісту в веб-сторінки.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття про інформаційну безпеку

Інформаційна система - це система, яка призначена для обробки, зберігання та подання інформації в необхідній формі для вирішення поставлених завдань. Інформаційна система включає в себе засоби і методи для роботи з інформацією, а також персонал.

У сучасному розумінні інформаційна система передбачає використання обчислювальної техніки як основного засобу обробки інформації, тобто програмні, програмно-апаратні та апаратні засоби.

Інформаційні системи можуть вирішувати широкий спектр завдань різного роду, тому необхідно класифікувати їх. Класифікація інформаційних систем показана на рис. 1.1.

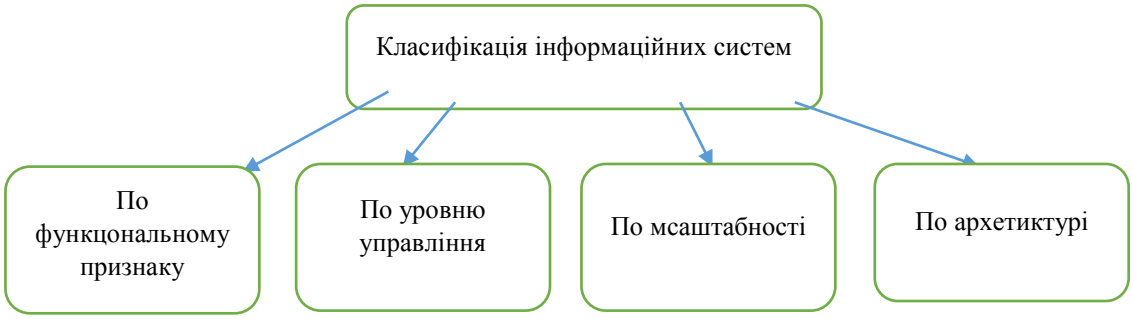


Рис. 1.1 - Класифікація інформаційних систем

Існує кілька видів класифікації інформаційних систем:

– за функціональною ознакою:

– автоматизована система;

– системи прийняття рішень;

– обчислювальні системи;

– довідкові системи, системи навчання) ;

– за рівнем управління:

– системи операційного рівня;

- бухгалтерські, банківські, системи бронювання, обробки замовлень та інші;

– експертні системи;

– системи рівня управління (тактичні);

– системи рівня реалізації (стратегічні);

– по сфері застосування (медичні, геоінформаційні і т. п.);

– по архітектурі:

– локальні (настільні);

– розподілені (файл-серверні, клієнт-серверні)

– за масштабністю:

– персональні (індивідуальні);

– колективні;

– корпоративні.

Також інформаційні системи можуть поєднувати в собі кілька типів з даної класифікації, в такому випадку прийнято говорити про поділ інформаційної системи на підсистеми.

Інформаційна підсистема - частина інформаційної системи, що реалізує деяку частину функціоналу системи і пов'язана з іншими підсистемами.

Незважаючи на вжиті дорогі методи, функціонування комп'ютерних інформаційних систем виявило наявність слабких місць у захисті інформації. Неминучим наслідком стали постійно зростаючі витрати і зусилля для захисту інформації. Однак для того, щоб вжиті заходи виявилися ефективними, необхідно визначити, що таке загроза безпеці інформації, виявити можливі канали витоку інформації та шляхи несанкціонованого доступу до даних, що захищаються. Підкреслимо, що саме поняття «загроза» в різних ситуаціях часто трактується по-

різному. Наприклад, для підкреслено відкритої організації загроз конфіденційності може просто не існувати - вся інформація вважається загальнодоступною (ЗМІ); проте в більшості випадків нелегальний доступ є серйозною небезпекою. Іншими словами, загрози залежать від інтересів суб'єктів інформаційних відносин (і від того, який збиток є для них неприйнятним). Отже, реалізація загроз інформаційній безпеці полягає в порушенні конфіденційності, цілісності та доступності інформації. Зловмисник може ознайомитися з конфіденційною інформацією, модифікувати її, або навіть знищити, а також обмежити або блокувати доступ легального користувача до інформації. При цьому зловмисником може бути, як співробітник організації, так і стороння особа. Але, крім цього, цінність інформації може зменшитися через випадкові, ненавмисні помилки персоналу, а також непередбачуваних чинників. Класифікація інформаційних загроз представлені на рисунках 1.2, 1.3.



Рис.1.2 - Типи інформаційних загроз

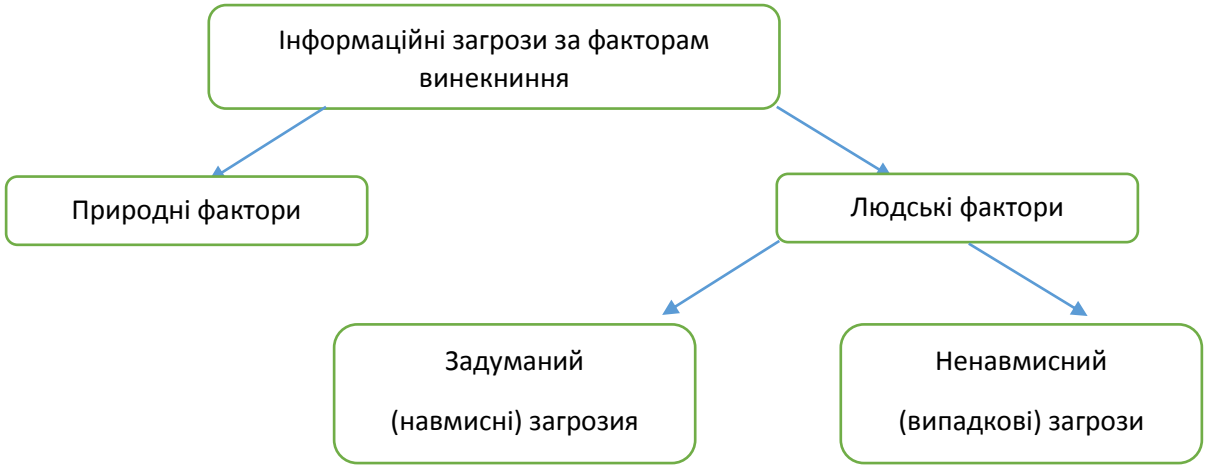


Рис. 1.3 - Класифікація інформаційних загроз за факторами виникнення

Класифікація інформаційних загроз за факторами виникнення

Інформаційні загрози можуть бути обумовлені:

* природними факторами (стихійні лиха — пожежа, повінь, ураган, блискавка та інші причини);

* людськими факторами. Останні, в свою чергу, поділяються на:

- загрози, що носять випадковий, ненавмисний характер. Це загрози, пов'язані з помилками процесу підготовки, обробки і передачі інформації (науково-технічна, комерційна, валютно-фінансова документація); загрози, пов'язані з помилками процесу проектування, розробки і виготовлення систем і їх компонент (будівлі, споруди, приміщення, комп'ютери, засоби зв'язку, операційні системи і ін.) з помилками в роботі апаратури; з помилками процесу підготовки та обробки інформації (помилки програмістів та користувачів через недостатню кваліфікацію та неякісне обслуговування, помилки операторів при підготовці, введенні та виведенні даних, коригуванні та обробці інформації);

- загрози, зумовлені навмисними, навмисними діями людей. Це загрози, пов'язані з передачею, спотворенням і знищенням наукових відкриттів, винаходів секретів виробництва, нових технологій але корисливим і іншим антигромадським мотивами (документація, креслення, описи відкриттів і винаходів та інші матеріали); підслуховуванням і передачею службових та інших науково-технічних і комерційних розмов. Це загрози, пов'язані з несанкціонованим доступом до ресурсів автоматизованої інформаційної системи (внесення технічних змін в засоби обчислювальної техніки і засоби зв'язку, підключення до засобів обчислювальної техніки і каналах зв'язку, розкрадання носіїв інформації).

Внутрішні загрози найчастіше визначаються соціально напруженістю і важким моральним кліматом.

Зовнішні загрози можуть визначатися зловмисними діями конкурентів, економічними умовами та іншими причинами (наприклад, стихійними лихами). Набуло широкого поширення промислове шпигунство, що завдає шкоди власнику комерційної таємниці, проводит незаконний збір, привласнення і передачу відомостей, що становлять комерційну таємницю, особою, не уповноваженою на це її власником. До основних загроз безпеки відносять:

* розкриття конфіденційної інформації;

* компрометація інформації;

* несанкціоноване використання інформаційних ресурсів;

* помилкове використання ресурсів; несанкціонований обмін інформацією;

* відмова від інформації;

* відмова від обслуговування.

Огляд шкідливих програм. Під шкідливими програмами надалі будемо розуміти такі програми, які прямо або побічно дезорганізують процес обробки інформації або сприяють витоку, або спотворенню інформації [1].

Нижче розглянемо деякі (найпоширеніші) види подібних програм:

"Троянський кінь" - програма, що виконує на додаток до основних (проектних і документованих) не описані в документації дії. Аналогія з давньогрецьким "троянським конем" таким чином, цілком виправдана. Небезпека "троянського коня" полягає в додатковому блоці команд, тим чи іншим чином вставленому в вихідну нешкідливу програму, яка потім пропонується (дарується, продається, підміняється) користувачем. Цей блок команд може спрацьовувати при настанні деякої умови (дати, часу і т.д., або по команді ззовні). Найбільш небезпечні дії "троянський кінь" може виконувати, якщо запустив її користувач, що володіє розширеним набором привілеїв. В цьому випадку зловмисник, який склав і впровадив «троянського коня», і сам цими привілеями не володіє, може виконати несанкціоновані привілейовані функції чужими руками. Або, наприклад, зловмисника дуже цікавлять набори даних користувача, який запустив таку

програму. Останній може навіть не володіти розширеним набором привілеїв - це не завадить виконанню несанкціонованих дій [2].

Вірус - це програма, яка може заражати інші програми шляхом включення в них своєї, можливо модифікованої, копії, причому остання зберігає здатність до подальшого розмноження. Своєю назвою комп'ютерні віруси зобов'язані певній схожості з вірусами біологічними:

* здібностями до саморозмноження;

* високій швидкості поширення;

* вибірковості уражуваних систем (кожен вірус вражає тільки певні системи або однорідні групи систем);

* наявності в більшості випадків певного інкубаційного періоду;

* здатності "заражати" ще незаражені системи;

* труднощі боротьби з вірусами і т. д.

Основні класи вірусів:

- Мережеві: поширюються по комп'ютерній мережі
- Файлові: впроваджуються в виконуються в файли
- Завантажувальні: впроваджуються в завантажувальний сектор диска (Boot-сектор)

Останнім часом до цих особливостей, характерним для вірусів комп'ютерних і біологічних, можна додати ще й швидкість появи модифікацій і нових поколінь вірусів, що можна пояснити ідеями і навичками зловмисників [3].

Програма, всередині якої знаходиться вірус, називається «враженою». Коли така програма починає роботу, то спочатку управління отримує вірус. Вірус знаходить і «заражає» інші програми, а також виконує якісь шкідливі дії. Процес зараження вірусом програмних файлів можна представити наступним чином. У зараженій програмі код останньої змінюється таким чином, щоб вірус отримав управління першим, до початку роботи програми - вірусоносія. При передачі управління вірусу він будь-яким способом знаходить нову програму і виконує

вставку власної копії в початок або додавання її в кінець цієї, зазвичай ще не зараженої, програми. Якщо вірус записується в кінець програми, то він коригує код програми з тим, щоб отримати управління першим. Після цього управління передається програмі-вірусоносію, і та нормально виконує свої функції. Більш витончені віруси можуть для отримання управління змінювати системні області накопичувача (наприклад, сектор каталогу), залишаючи довжину і вміст зараженого файлу без змін.

Завантажувальні віруси. Від файлових вірусів завантажувальні віруси відрізняються методом поширення. Вони вражають не програмні файли, а певні системні області носіїв (жорстких дисків). На включеному комп'ютері вони можуть тимчасово розташовуватися в оперативній пам'яті. Зазвичай враження відбувається при спробі завантаження комп'ютера з магнітного носія, системна область якого містить завантажувальний вірус. Далі цей комп'ютер сам стає джерелом поширення завантажувального вірусу.

Макровіруси. Це особливий різновид вірусів вражає документи, виконані в деяких прикладних програмах, що мають засоби для виконання так званих макрокоманд. Зокрема, до таких документів належать документи текстового процесора MicrosoftWord. Зараження відбувається при відкритті файлу документа у вікні програми, якщо в ній не відключена можливість виконання макросів [4].

"Хробак/ Черв'як" - програма, що поширюється через мережу і не залишає своєї копії на фізичному носії. "Черв'як" використовує механізми підтримки мережі для визначення вузла, який може бути заражений. Потім за допомогою тих же механізмів передає своє тіло або його частину на цей вузол і або активізується, або чекає для цього відповідних умов. Найбільш відомий представник цього класу-вірус Морріса (або, вірніше «"черв'як Морріса"), який вразив мережу Internet в 1988 р. Найбільш сприятливим середовищем поширення "хробака" є мережа, всі користувачі якої вважаються дружніми і довіряють один одному. Відсутність захисних механізмів якнайкраще сприяє уразливості мережі.

"Жадібні" програми - це програми, які при виконанні прагнуть монополізувати якийсь ресурс системи, не даючи іншим програмам можливості використовувати його. Доступ таких програм до ресурсів системи зазвичай призводить до порушення її доступності. Природно, така атака буде активним втручанням в роботу системи. Безпосередній атаці зазвичай піддаються ключові об'єкти системи: процесор, оперативна пам'ять, пристрої введення-виведення.

Тупикова ситуація виникає, коли «жадібна» програма нескінченна (наприклад, виконує свідомо нескінченний цикл). Однак у багатьох операційних системах існує можливість обмеження часу процесора, що використовується завданням.

Іншим прикладом «жадібної» програми є програма, яка захоплює занадто велику область оперативної пам'яті. В оперативній пам'яті послідовно розміщуються дані, наприклад, підкачувані з зовнішнього носія. Зрештою пам'ять може опинитися у володінні однієї програми, і виконання інших виявиться неможливим.

Шпигунські програми. Як випливає з назви, ці програми призначені для збору даних та надсилання їх третій стороні без вашого відома та згоди. Такі програми можуть відстежувати натискання клавіш (клавіатурні шпигуни), збирати конфіденційну інформацію (паролі, номери кредитних карт, PIN-коди і т.д.), відстежувати адреси електронної пошти в поштовій скриньці або особливості вашої роботи в Інтернеті. Крім того, шпигунські програми неминуче знижують продуктивність комп'ютера.

Загарбники паролів. Це програми спеціально призначені для крадіжки паролів. При спробі входу імітується введення імені і пароля, які пересилаються власнику програми-загарбника, після чого виводиться повідомлення про помилку введення і управління повертається операційній системі. Користувач, який думає, що допустив помилку при наборі пароля, повторює вхід і отримує доступ до системи. Однак його ім'я та пароль вже відомі власнику програми-загарбника.

Перехоплення пароля може здійснюватися і іншим способом - за допомогою впливу на програму, керуючу входом користувачів в систему і її набори даних [5].

Методика впливу шкідливих програм значною мірою залежить від організації обробки інформації в системі, розробленої політики безпеки, можливостей встановлених засобів захисту, а також сумлінності адміністратора і оператора. Для реалізації атаки існує два способи:

- по-перше, можна подолати систему захисту, тобто шляхом різних впливів на неї припинити її дії щодо себе або своїх програм. Це складно, трудомістко і не завжди можливо, зате ефективно;

- по-друге, можна поспостерігати за тим, що «погано лежить», тобто які набори даних, що представляють інтерес для зловмисника, відкриті для доступу через недогляд адміністратора. Такий доступ теж можна назвати несанкціонованим, його легко здійснити, але від нього легко і захиститися. До цього ж типу відноситься атаки з підбором пароля, оскільки здійснити такий підбір можливо лише в разі порушення правил складання паролів і використання в якості пароля людських імен, повторюваних символів та ін.

У переважній більшості випадків атаки стають можливими через непродуманий вибір засобів захисту, їх некоректної установки і настройки, поганого контролю роботи, а також при недбалому ставленні до захисту своїх власних даних.

Незаконне використання привілеїв. Зловмисники, які застосовують даний спосіб атаки, зазвичай використовують штатне програмне забезпечення (системне або прикладне), що функціонує в нештатному режимі.

Практично будь-яка захищена система містить засоби, що використовуються в надзвичайних ситуаціях, при збоях обладнання або засоби, які здатні функціонувати з порушенням існуючої політики безпеки. У деяких випадках користувач повинен мати можливість доступу до всіх наборів системи (наприклад, при раптовій перевірці).

Такі засоби необхідні, але вони можуть бути надзвичайно небезпечними. Зазвичай ці засоби використовуються адміністраторами, операторами, системними програмістами та іншими користувачами, які виконують спеціальні функції [6].

Для того щоб зменшити ризик від застосування таких засобів більшість систем захисту реалізує такі функції за допомогою набору привілеїв - для виконання певної функції потрібен певний привілей. У цьому випадку кожен користувач отримує свій набір привілеїв, звичайні користувачі - мінімальний, адміністратори - максимальний (відповідно до принципу мінімуму привілеїв). Природно, при таких умовах розширений набір привілеїв - заповітна мрія будь-якого зловмисника. Він дозволить йому здійснювати практично будь-які дії, причому, можливо, навіть в обхід всіх заходів контролю. Порушення, що здійснюються за допомогою незаконного використання привілеїв, є активним впливом, що здійснюються з метою доступу до якого-небудь об'єкту або системі в цілому.

Незаконне захоплення привілеїв можливо або при наявності помилок в самій системі захисту (що, наприклад, виявилося можливим в одній з версій операційної системи UNIX), або в разі недбалості при управлінні системою і привілеями зокрема (наприклад, при призначенні розширеного набору привілеїв всім підряд).

Атаки "салямі" найбільше характерні для систем, що обробляють грошові рахунки і, отже, для банків особливо актуальні. Принцип атак "салямі" побудований на тому факті, що при обробці рахунків використовуються цілі одиниці, а при обчисленні відсотків нерідко виходять дробові суми. Наприклад, 6,5% річних від $102,87 за 31 день складе $0,5495726. Банківська система може округлити цю суму до $0.55. Однак якщо користувач має доступ до банківських рахунків або програм їх обробки, він може округлити її в іншу сторону - до $0.54, а різницю в 1 цент записати на свій рахунок. Власник рахунку навряд чи її помітить, а якщо і зверне увагу, то спише її на похибки обробки і не додасть значення. Зловмисник же отримає прибуток в один цент, при обробці 10.000

рахунків в день. Його прибуток таким чином складе $1000, тобто близько $300 000 на рік.

Природно, такі атаки мають сенс лише в тих організаціях, де здійснюється не менше 5000 - 10000 транзакцій в день, інакше не має сенсу ризикувати, оскільки в разі виявлення злочинця просто визначити. Таким чином, атаки «салямі» небезпечні в основному для великих банків. Причинами атак "салямі" є, по-перше, похибки обчислень, що дозволяють трактувати правила округлення в ту чи іншу сторону, а по-друге, величезні обсяги обчислень, необхідні для обробки рахунків.

"Приховані канали" - шляхи передачі інформації між процесами системи, що порушують системну політику безпеки. У середовищі з поділом доступу до інформації користувач може не отримати дозвіл на обробку ого даних, проте може придумати для цього обхідні шляхи. Практично будь-яка дія в системі якимось чином зачіпає інші її елементи, які при цьому можуть змінювати свій стан. При достатній спостережливості і знанні цих зв'язків можна отримати прямий або опосередкований доступ до даних [7].

Наприклад, програміст банку не завжди має доступ до імен та залишків депозитних рахунків. Програміст системи, призначеної для обробки цінних паперів, може не мати доступ до пропозицій про покупку або продаж. Однак при створенні таких систем він може передбачити спосіб отримання цікавлять його відомостей.

«Маскарад». Під "маскарадом" розуміється виконання будь-яких дій одним користувачем від імені іншого користувача. При цьому такі дії іншому користувачеві можуть бути дозволені. Порушення полягає в присвоєнні прав і привілеїв. Мета "маскараду" - приховування будь-яких дій за ім'ям іншого користувача або присвоєння прав і привілеїв іншого користувача для доступу до його наборів даних або для використання його привілеїв. "Маскарад" - це спосіб активного порушення захисту системи, він є опосередкованим впливом, тобто

впливом, вчиненим з використанням можливостей інших користувачів. "Маскарадом" також називають передачу повідомлень в мережі від імені іншого користувача. Способи заміни ідентифікатора можуть бути різні, зазвичай вони визначаються помилками і особливостями мережевих протоколів. Проте на приймальному вузлі таке повідомлення буде сприйнято як коректне, що може привести до серйозних порушень роботи мережі. Особливо це стосується керуючих повідомлень, що змінюють конфігурацію мережі, або повідомлень, що ведуть до виконання привілейованих операцій. Найбільш небезпечний "маскарад" в банківських системах електронних платежів, де неправильна ідентифікація клієнта може призвести до величезних збитків. Особливо це стосується платежів за допомогою електронних банківських карт. Сам по собі метод ідентифікації за допомогою персонального ідентифікатора (PIN) досить надійний, порушення можуть відбуватися внаслідок помилок його використання. Це станеться, наприклад, в разі втрати кредитної картки, при використанні очевидного ідентифікатора (свого імені, ключового слова і т.д.). Тому клієнтам треба суворо дотримуватися всіх рекомендацій банку щодо виконання такого роду платежів.

"Збір сміття". Після закінчення роботи оброблювана інформація не завжди повністю видаляється з пам'яті. Частина даних може залишатися в оперативній пам'яті, на дисках і інших носіях. Дані зберігаються на носії до перезапису або знищення; при виконанні цих дій на звільненому просторі диска знаходяться їх залишки. Хоча прочитати такі дані важко, проте, використовуючи спеціальні програми та обладнання, все ж можливо. Такий процес прийнято називати «збиранням сміття». Він може привести до витоку важливої інформації.

«Люк». Люком називається не описана в документації на програмний продукт можливість роботи з цим програмним продуктом. Сутність використання люків полягає в тому, що при виконанні користувачем деяких не описаних в документації дій він отримує доступ до можливостей і даних, які в звичайних умовах для нього закриті (зокрема - вихід в привілейований режим).

1.2 Характеристики Веб-ресурсів

Документи в Інтернеті надаються у вигляді веб-сторінок, гіпертекстових файлів, пов'язаних з іншими документами через гіперпосилання. Структура веб-сторінки та всіх елементів, що входять до неї, визначається за допомогою коду мови розмітки, як правило, HTML, який аналізується та відображається веб-браузером.

Вміст сторінки може динамічно оновлюватися за допомогою JavaScript, мови сценаріїв, що виконується браузером. Код JavaScript може бути включений у веб-сторінку та керувати нею, змінюючи Document Object Model (DOM), деревоподібне представлення веб-сторінки. У роботі [1] дослідники представили узагальнену архітектуру будь-якого веб-додатку, звідки виявили всі шляхи проведення різних атак.

Веб-сторінки запитуються та передаються через протокол гіпертекстової передачі (HTTP), протокол «запит-відповідь» на основі парадигми клієнт-сервер. Браузер діє як клієнт і надсилає HTTP-запити на ресурси, розміщені на віддалених серверах. Сервери, в свою чергу, надають відповіді, що містять запитані ресурси, якщо вони доступні. Всі потоки HTTP-трафіку прозорі, тому протокол HTTP не гарантує конфіденційність і цілісність зв'язку. Для захисту обміну даними був розроблений протокол HTTP Secure (HTTPS), який передає звичайний HTTP-трафік в зашифрованому каналі TLS/SSL. І HTTP, і його захищений варіант HTTPS є протоколами «Stateless», тобто кожен запит обробляється сервером як незалежний від усіх інших. Однак деякі веб-програми повинні пам'ятати інформацію про попередні запити, наприклад, щоб відстежувати, чи дотримувався Користувач необхідних кроків для процедури оплати. Для цих цілей існують файли Cookie. Вони є найпоширенішим механізмом зберігання інформації про запит клієнта та впровадження сеансів в Інтернеті. Грубо кажучи, Cookie являє собою пару «ключ-значення», яка записується сервером в браузер клієнта і кожен

раз при спробі відкрити сторінку відповідного сайту браузер пересилає цей фрагмент даних веб-серверу в складі HTTP-запиту. Файли Cookie можуть або безпосередньо кодувати інформацію про стан, або, найчастіше, включати унікальний ідентифікатор сеансу, що дозволяє серверу ідентифікувати клієнта-запитувача та відновити відповідний стан сеансу при обробці декількох запитів одним і тим же клієнтом. На рис. 1.4 представлені компоненти будь-якого веб-додатку, описаного вище.

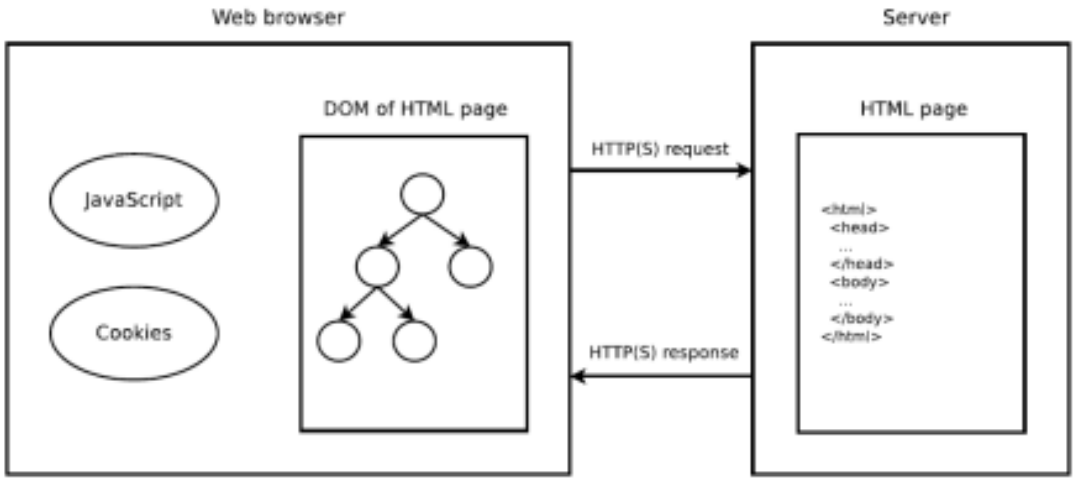


Рис. 1.4 - Узагальнена Архітектура веб-додатків

Вебдодаток - це клієнт-серверна програма, яка працює на веб-платформі. Вона є невід'ємною частиною сучасної веб-екосистеми, яка забезпечує динамічне надання інформації та послуг.

Веб-додаток складається з коду виконуваного, як на стороні сервера, так і на стороні клієнта. На стороні сервера, веб-додаток приймає вхідні дані від користувача через HTTP-запити і взаємодіє з локальною файловою системою, базою даних або іншими компонентами для доступу до даних і пошуку інформації. Після сервер повертає вихідні дані (тобто згенеровані Html-сторінки), які надсилаються клієнту через HTTP-відповіді. На клієнтській стороні

відображаються HTML-сторінки, а клієнтський код (тобто JavaScript), вбудований у відповіді HTTP, виконується веб-браузером.

Клієнтський код також може взаємодіяти з серверним кодом асинхронно, не заважаючи відображенню існуючої HTML-сторінки через AJAX і динамічно оновлювати сторінку. В даний час з метою сприяння швидкій розробці додатків розробляються деякі бібліотеки і платформи програмування (наприклад, Ruby on Rails і Django). Наприклад, більшість платформ веб-розробки надають функції управління сеансами, які розробники можуть безпосередньо використовувати для управління веб-сеансами.

Модель виконання веб-додатків кардинально відрізняється від традиційних додатків. Розробка веб-додатків також стикається з труднощами, з якими не доводиться стикатися при розробці традиційних додатків. По-перше, у відкритому веб-середовищі введені користувачем дані потенційно небезпечні і ніколи не можна їм довіряти. Перевірка даних, що вводяться, є важливою частиною веб-програми для виявлення та очищення ненадійних даних, що вводяться користувачем. Хоча перевірка вхідних даних є важливою функцією, необхідною для всіх типів додатків, її реалізація у веб-додатках є набагато складнішою через унікальні особливості технологій веброзробки.

По-друге, зв'язок між клієнтом і веб-додатком здійснюється за допомогою протоколу HTTP. Як результат, кілька входів від одного і того ж користувача будуть сприйматися як різні, незалежні користувачі веб-додатків. Для корекції таких дій веб-програма повинна використовувати управління сеансами для відображення веб-запитів від одного користувача до веб-сеансу. Оскільки результати обчислень з боку клієнта ніколи не можуть бути довіреними, потрібна додаткова перевірка цих результатів на стороні сервера. Ці унікальні можливості істотно ускладнюють логічну реалізацію веб-додатків, які повинні забезпечувати управління потоком між клієнтом і сервером і між різними модулями.

1.3 Комп'ютерні злочини

Для розуміння масштабів обсягу інформатизації суспільства наведемо такі цифри: зараз у світі налічується близько 3,5 млрд. людей, що мають доступ до Інтернету. Це половина всього населення Землі.

Мережею Інтернет в сукупності зі своїми внутрішніми інформаційними мережами користуються всі банки і корпорації, що акумулюють на своїх рахунках великі грошові кошти, ласий шматок для різного роду комп'ютерних шахраїв. Число злочинних посягань на майно, що здійснюються шляхом зловживання довірою або обманом, продовжує зростати.

Поняття "комп'ютерна злочинність" охоплює злочини, що здійснюються за допомогою комп'ютерів, інформаційно-обчислювальних систем і засобів телекомунікацій, або спрямовані проти них з корисливими або деякими іншими цілями.

Комп'ютерне злочин як кримінально-правове поняття - це передбачене кримінальним законом умисне порушення чужих прав та інтересів щодо автоматизованих систем обробки даних, вчинене на шкоду підлягає правової охорони прав та інтересів фізичних і юридичних осіб, суспільства і держави. У більш загальному вигляді способи здійснення таких злочинів можна класифікувати на:

* несанкціонований доступ;
* вірусна модифікація;
* перехоплення інформації;
* комбіноване використання.

Несанкціонований доступ включає:

* несанкціоноване підключення;
* несанкціоновану модифікацію;
* несанкціоноване блокування;
* несанкціоноване знищення.

До несанкціонованого підключення відносяться несанкціонований доступ до обчислювальних ресурсів, вплив на парольно-ключові системи, установка програмних продуктів. Варто звернути увагу на те, що існує безліч програм-«зломщиків», званих професійною мовою HACK-TOOLS (інструмент злому). Ці програми працюють за принципом простого перебору символів. Але вони стають малоефективними в комп'ютерних системах, що володіють програмою - "захисником" комп'ютерних портів, що веде автоматичний протокол звернень до комп'ютера і відключає абонента, якщо пароль невірний. Тому останнім часом злочинцями став активно використовуватися метод «інтелектуального перебору», заснований на підборі передбачуваного пароля, виходячи із заздалегідь визначених тематичних груп його приналежності. Цікаві результати експериментів, представлені фахівцями у формі таблиці 1.1.

Таблиця 1.1

Типологізація комп'ютерних злочинців

Тематичні групи паролів	% частоти вибору пароля людиною	% розкриття пароль
Імена, прізвища та похідні	22,2	54,5
Інтереси (хобі, спорт, музика)	9,5	29,2
Дати народження, знаки зодіаку свої і близьких; їх комбінація з першою групою	11,8	54,5
Адреса проживання, місце народження	4,7	55
Послідовність клавіш ПК, повтор символ	16,1	72,3
Номери документів (паспорт, пропуск, посвідчення і т. д.)	3.5	100
Інші	30,7	5,7

Особи, які вчиняють комп'ютерні злочини, можуть бути об'єднані в три великі групи:

* особи, які не пов'язані трудовими відносинами з організацією-жертвою, але мають деякі зв'язки з нею;

* співробітники організації, що займають відповідальні пости;

* співробітники-користувачі ЕОМ, які зловживають своїм становищем.

Наприклад, близько 90% зловживань у фінансовій сфері, пов'язаних з порушеннями в області інформаційної безпеки, відбувається при прямій або непрямій участі діючих або колишніх працівників банків. При цьому на злочинний шлях часто стають найкваліфікованіші, що володіють максимальними правами в автоматизованих системах категорії банківських службовців – системні адміністратори та інші співробітники служб автоматизації банків.

Таблиця 1.2

Типи та суб'єкти загроз

Тип загроз	Оператор	Керівник	Програміст	Інженер (технік)	Пользователь	Конкурент
Зміна кодів	+		+			
Копіювання файлів	+		+			
Знищення файлів	+	+	+		+	+
Присвоєння програм			+	+		+
Шпигунство	+	+	+			+
Установка			+	+		+
Саботаж	+		+	+		+
Продаж даних	+	+	+		+	
Злодійство		+	+		+	+

Суб'єктів комп'ютерних злочинів з точки зору професійної підготовленості прийнято поділяти на осіб, які вчиняють злочини:

а) "нетехнічні";

б) «технічні», що вимагають мінімуму спеціальних знань;

в) «високотехнічні», можливі за умови ґрунтовного володіння обчислювальною технікою.

Практика показує, що більшість злочинів категорії "А" здійснюють малознайомі з обчислювальною технікою службовці із середньою освітою. Однак цих людей відрізняють дві якості: вони мають доступ до комп'ютера і знають, які функції виконує він в їх організації. «Нетехнічні» злочини скоюються головним чином шляхом крадіжки пароля доступу до файлів інформації, що зберігається в машинній пам'яті. Володіючи паролем і певними навичками, можна увійти в засекречені файли, змінити їх зміст і т. п. Ці злочини досить прості для розслідування, і, посиливши захист системи, їх легко попередити.

"Технічні" злочини пов'язані з маніпуляціями програмами, які складені фахівцями. Змінити їх можуть лише особи, які мають відповідну кваліфікацію. Найбільші труднощі для правоохоронних органів представляють "високотехнічні" злочини.

Аналізуючи комп'ютерні злочини, можна встановити спільність «почерку» зловмисника (або організованої групи):

* реалізується несанкціонований доступ до автоматизованої інформаційної системи, до її апаратно-програмних засобів, модифікуються важливі записи;

* вносяться зміни в існуюче програмне забезпечення для створення спеціальних рахунків фізичних і юридичних осіб, розсилки фальшивих платіжних документів та ін.

* паралельно знищуються сліди комп'ютерного злочину шляхом модернізації бухгалтерських документів аналітичного та синтетичного обліку;

* здійснюється отримання готівкових коштів.

Небезпека, як правило, є всередині організації, а не поза нею. Комп'ютерному злочину сприяють такі передумови, як відсутність належної системи бухгалтерського обліку, комплексного захисту інформації та її контролю, слабка готовність персоналу, відсутність або слабка організація системи поділу доступу і т. п.

РОЗДІЛ 2 МЕТОДИ ТА АЛГОРИТМИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ

2.1 Існуючі методи захисту веб-ресурсів

Деякі засоби захисту виникають як відповідь на новий тип атак, в той час як інші розроблені через необхідність захищати існуючий додаток від агресивного середовища. Таким чином, засоби захисту від атак, як правило, розробляються для універсального застосування в різних додатках, тоді як засоби захисту конкретних додатків прагнуть мати можливість обробляти різні типи атак.

На додаток до цієї різниці, розробник захисту може також має мати уявлення про те, як повинен працювати захист. Різні розробники захисту можуть погодитися з тим, що деякі класи атак є успішними через неправильну перевірку вхідних даних, тому пропонується загальна категорія захисту «перевірка вхідних даних». Однак, незважаючи на те, що розробники захисту мають однакове уявлення, фактичні реалізації захисту можуть істотно відрізнятися.

Наприклад, на основі взаємозв'язку системи захисту з вихідним кодом додатку який підлягає захисту, методи захисту можуть бути згруповані за підходами «білий ящик» і «чорний ящик». Захист може або вимагати доступ до джерел для попереднього аналізу, або реалізувати захист тільки за допомогою зовнішніх засобів.

Підходи "білого ящика" включають в себе аналіз вихідного коду і, таким чином, специфічні для конкретних мов програмування і в основному застосовні для сценарних мов. І навпаки, підходи «чорного ящика» зазвичай не залежать від використовуваної мови програмування, оскільки замість цього аналізуються зовнішні дії програми (наприклад, пари HTTP-запит/відповідь, дії доступу до бази даних тощо). Концептуальна різниця між цими двома підходами полягає в тому, що у випадку аналізу вихідного коду розглядається намір розробника програми. Однак деякі аспекти наміру можуть бути пропущені у випадку зовнішнього спостереження.

Іншим поширеним прикладом є групування систем захисту на основі їх розташування. Хоча зворотний проксі - сервер може захистити від XSSатак, деякі атаки, такі як руйнівні атаки SQL-ін'єкцій, залишаються невиявленими, оскільки шкідливі дії будуть вже прийняті до того часу, коли згенеровані сторінки досягнуть проксі-сервера. Аналіз причини цього недоліку дозволяє зробити висновок про те, що для ефективної роботи проксі-методів необхідно вести спостереження за всіма вихідними точками додатки. Точки виходу включають усі комунікації із зовнішніми ресурсами, такими як спроба отримати доступ до бази даних, мережі або файлової системи. Подібне обмеження застосовується до методів захисту на основі журналів подій (журналів) [11]. Деякі з менш динамічних методів захисту включають аналіз файлів журналів додатків. Найбільша перевага методів зворотного проксі-захисту та захисту на основі журналів полягає в тому, що вони можуть бути легко розгорнуті в різних середовищах. Важливою перевагою методів на основі зворотного проксі-сервера є те, що проксі має більше контролю над переданими даними. Наприклад, такий проксі-сервер може змінювати або повністю видаляти передані дані, для того щоб запобігти атаці. Однак для методів, заснованих на моніторингу журналів, може бути занадто пізно щоб запобігти атаці до того часу, коли шкідлива дія буде доступна у файлі журналу. Таким чином, аналіз журналів частіше використовується для виявлення аномалій/вторгнень, а не для їх запобігання. В іншій частині розділу розглядаються питання захисту, згруповані за деякими загальними ознаками, використаним для класифікації за існуючими методами. На закінчення наводиться зведена таблиця з описом місця розглянутих методів захисту в пропонованій класифікації.

Методи "білого ящика". Опис підходу до захисту як методу "білого ящика" не є вельми інформативним і може бути використаним тільки для швидкого визначення загальної застосовності цього підходу. Тому слід надати додаткову інформацію, щоб розрізняти різні методи «білого ящика». У статті [12] автори

описали різні підходи «білої скриньки», а також розглянули недоліки цієї концепції. Наприклад, аналіз вихідного коду, який використовується, може працювати до виконання коду (статичний аналіз) або під час виконання (динамічний аналіз). Статичний аналіз передбачає, що виявлені проблеми повинні бути усунені до того, як додаток буде фактично використано у виробничому середовищі. Результати такого аналізу можна розглядати як рекомендації щодо усунення виявлених проблем. Хоча в деяких випадках виявлені проблеми можуть бути усунені автоматично, використання неавтоматичного статичного аналізу коду все ще може бути корисним на етапі розробки програми. Динамічний аналіз відстежує виконання коду програми під час роботи і може реагувати більш активно (наприклад, запобігати виконанню деяких гілок коду). Аналіз часу виконання також дозволяє аналізатору мати доступ до даних, наданих користувачем. Така гнучкість зазвичай досягається шляхом розширення існуючого інтерпретатора мови програмування або стандартних бібліотек. Існують також і гібридні підходи аналізу, які можуть автоматично усувати виявлені проблеми, залучаючи розробників до розгляду виявлених вразливостей і прийняття деяких рішень [13]. Інша особливість аналізу часу виконання полягає в тому, що наявність доступу до вихідного коду дозволяє створити модель передбачуваної поведінки програми. Потім така модель може бути використана на етапі прийняття рішення про вторгнення, щоб перевірити, чи очікувана поведінка програми. Підходи "білого ящика" також можуть бути зосереджені на конкретних аспектах застосування, таких як використовувані API (замість аналізу самого додатка), наближена структура веб-сторінок. Незалежно від деталей використовуваного аналізу, через необхідність доступу до джерел додатків використання методів «білого ящика» концептуально обмежене або програмним забезпеченням з відкритим вихідним кодом, або додатками, розробленими власними силами.

Методи «чорного ящика «аналогічно групі підходів» білого ящика, для проведення відмінності між різними підходами» чорного ящика потрібні додаткові деталі. Загальна відмінність між підходами полягає в предметі спостереження. Перегляд веб-програми у вигляді чорної скриньки означає, що підходи до захисту мають доступ лише до вхідних та вихідних даних програми. У роботі [14] розглянуто даний підхід і виділені його суттєві недоліки. Перевага впровадження чорних скриньок полягає в тому, що такі підходи можуть бути розгорнуті для будь-якої веб-програми (навіть якщо вихідний код недоступний). Тим не менш, більш глибоке розуміння того, як програма повинна поводитися, може бути неможливим з підходами «чорної скриньки», що потенційно може призвести до зниження швидкості виявлення атаки. Тому дані методи не будуть розглянуті в даному огляді. Далі будуть описані актуальні методи захисту від існуючих видів атак на веб-ресурси.

Захист від SQL-ін'єкцій. Одну з серйозних загроз для веб-додатків представляють так звані Sql-ін'єкції. Вони є однією з найпоширеніших атак на веб-додатки, що використовують базу даних, тому найбільша кількість методів захисту розроблена спеціально для обходу таких атак. Однак уразливості, пов'язані з SQL, традиційно пояснюється відсутністю належного захисту вхідних даних. Розглянемо наступний спрощений фрагмент коду PHP:

$mails = mysqli_query($con, “SELECT MailSubject FROM Mails WHERE MailDate > ‘”.$_GET["date"].”’”);

Залишаючи осторонь проблеми з автентифікацією, такий код може бути використаний для вилучення списку повідомлень електронної пошти, отриманих після вказаної дати. Небезпека цього коду полягає в, здавалося б, нешкідливому GET HTTP-запиті с параметром «date». Небажані наслідки можуть виникнути, якщо відвідувач веб-сайту підставить наступне значення для дати:

2022-01-01‘ OR 1=1

У цьому випадку буде створено наступний запит SQL:

SELECT MailSubject FROM Mails WHERE MailDate > ‘2022-01- 01’ OR 1=1

Це призведе до включення в список всіх об'єктів електронної пошти. Більш руйнівні результати можуть бути досягнуті за допомогою іншої структури для значення «date», наприклад, включення символу «крапка з комою» для виконання більш ніж однієї команди SQL.

Хоча це приклад небезпечного коду, подібні фрагменти коду часто зустрічаються з історичних причин. Спочатку ризики безпеки не були повністю зрозумілі, і формування SQL-запитів «на льоту» шляхом об'єднання рядків виглядало простим і природним. Завдяки вдосконаленню методів безпеки уникнути таких вразливостей стало можливим завдяки використанню функцій екранування, таких як mysql_real_escape_string(). Застосування таких функцій неявно виражає наміри розробника щодо ролі вхідних даних. Така функція убезпечить код і гарантує, що вхідні дані будуть використовуватися тільки в якості зумовленого типу даних.

Знешкодження вхідних даних-це добре відома техніка, яка може запобігти деяким атакам. Цей підхід працює найкраще, якщо його застосовувати належним чином на етапі розробки, оскільки розробники мають можливість застосовувати дане знешкодження перед кожним викликом функції, яка працює з базою даних. Однак, особливо для складних додатків, деякі виклики функцій, звернені до бази даних, можуть бути пропущені через нагляд розробника, що може призвести до вразливостей.

Таким чином, були запропоновані різні автоматизовані підходи. Популярним прикладом таких підходів є шаблонізація запитів. Ключовою ідеєю створення шаблонів є повне усунення небезпечної конкатенації рядків. Для цього завдання підготовки остаточного SQL-запиту делегується спеціалізованій функції. Призначені для користувача вхідні дані передаються в таку функцію, яка фактично «очищає» їх перед тим як сформувати остаточну рядок Sqlзапиту [16]. Популярним прикладом такого підходу є технологія підготовлених заяв

(пов'язаних змінних), доступна в PHP. Для застарілих додатків додатковою проблемою є ідентифікація викликів чутливих функцій. Така ідентифікація залежить від мови і може включати складний статичний аналіз.

Інша проблема, характерна як для старого, так і майбутнього програмного забезпечення, полягає в тому, щоб формально перевірити правильність дезінфікуючої функції. Для багатомовних веб-додатків навіть порядок дезінфекції стає важливим. Якщо зловмисник зможе передбачити (принаймні частково) форму згенерованого SQL-запиту, атаки SQL-ін'єкції завершаться успішно. Такі знання дозволяють надавати зловмисникові вхідні дані, що змінюють структуру спочатку очікуваного SQL-запиту. Загалом, атаки шляхом ін'єкцій є можливими, якщо зловмисник знає синтаксис мови (HTML, SQL тощо). В останні роки було розроблено багато інструментів для виявлення та запобігання атакам ін'єкцій SQL. Розглянемо найбільш відомі з них. AMNESIA є засобом виявлення і запобігання для атак SQL-ін'єкцій. Для цієї мети вона використовує статичний аналіз і моніторинг виконання запитів. Інструмент будує модель законних запитів для кожної точки доступу, тобто. коли SQL-запити видаються ядру бази даних, інструмент контролює додаток під час виконання, щоб переконатися, що всі генеровані запити відповідають статично створеній моделі. SQLRand використовує набір інструкцій рандомізації для виявлення та переривання запитів вбудованого коду, і кожне SQL ключове слово поєднується з випадковим цілим числом, щоб ввести в оману зловмисника. Запропонована методика запобігає атакам SQL-ін'єкцій у збережених процедурах шляхом комбінування статичного аналізу коду програми з перевіркою виконання. Але якщо генератор псевдовипадкових чисел, який використовується для рандомізації синтаксису SQL, скомпрометований, тоді весь метод захисту виявиться недієвим.

Захист від XSS-атак. Cross Site Scripting (XSS) атака — це зловмисне програмне вплив на браузер користувача з метою крадіжки даних або заподіяння іншої шкоди. Щоб не плутати поняття з таблицями стилів CSS (Cascading Style

Sheets), домовилися в скороченому позначенні замінювати перший символ буквою «X».

Атаки XSS стали можливими через те, що поточна розробка сучасних веб-додатків зазвичай пов'язана з одночасним змішуванням декількох мов. Це нагадує архітектуру Фон Неймана, де дані та код зберігаються в одній пам'яті. Наприклад, HTML і JavaScript зазвичай повертаються відвідувачеві сайту в одному потоці. У статті [17] автори описали випадки виникнення XSS. Недостатнє розділення контексту і може призвести до атак XSS. Атаки XSS успішні в тих випадках, коли зловмисник може підставити дані, які помилково інтерпретуються або з боку сервера, або за допомогою браузера відвідувача іншого веб-сайту.

Типовим прикладом є впровадження конструкції типу <script>...< / script > у форму введення тексту. Якщо надана форма неналежним чином використовується для формування, наприклад, наступних веб-сторінок, сценарій, наданий в тезі <script>, може бути виконаний будь-якими відвідувачами вебсайту.

Атаки міжсайтового сценарію (XSS) зазвичай поділяються на два типи:

- активні (збережені);
- є пасивні (відбиті).

Активні XSS-атаки є постійними, так як шкідливо вводяться скрипти постійно зберігаються на стороні сервера, наприклад, в базі даних у вигляді повідомлення, відправленого на форум [18].

1) щоразу, коли веб-відвідувач отримує сторінку, що містить введений сценарій, браузер відвідувача починає виконувати шкідливий скрипт (якщо в браузері не вимкнено виконання активних сценаріїв).

2) Пасивні атаки можливі, навіть якщо введений скрипт спочатку відсутній на стороні сервера. Атаки можуть завершитися успішно, якщо сервер відповість сторінками, заснованими на введенні користувача. Якщо веб-сторінка не побудована належним чином, вхідні дані зловмисника можуть бути поміщені в неправильний контекст і згодом виконані на стороні клієнта.

Через те, що для пасивних атак корисне навантаження повертається до джерела, зловмисник повинен обдурити користувачів при відправці шкідливого скрипта на сервер. Така хитрість може бути досягнута, наприклад, шляхом надання жертві спеціально створеного посилання, наприклад, через електронну пошту.

Додатковий тип атак XSS включає атаки XSS DOM. Однак через те, що такі атаки можуть навіть не дійти до серверної частини, захист від цього специфічного типу атак виходить за рамки захисту веб-додатків. Хоча пропонується багато підходів до захисту від XSS, вони, як правило, базуються на дезінфекції даних користувачів. Оскільки атаки XSS націлені скоріше на клієнтів, захист від XSS може бути технічно розглянута за рамками захисту веб-додатків і в більшій мірі покладати відповідальність за безпеку на клієнта. Однак захист XSS не може бути реалізований виключно на клієнтському сайті, оскільки браузер не має попередніх знань про передбачувану структуру веб-сторінки. Тому самі веб-додатки можуть вибирати клієнтські Засоби захисту, які використовуються клієнтами. Два типи атак XSS, про які йде мова, стають можливими, оскільки браузери помилково вважають, що веб-сторінки, що генеруються сервером, повинні бути надійними, тому деякі дослідники припускають, що атаки XSS можна розглядати як проблему довіри, а не проблему невідповідності контексту.

Такі підходи, засновані на довірі, можуть включати, наприклад, підписання відомих довірених частин сторінки під час пропуску введених користувачем даних. Заглядаючи глибше, можна спостерігати, що методи, засновані на довірі, використовують модель з білим списком (певні частини позначені як надійні), тоді як підходи, засновані на дезінфекції, використовують модель з чорним списком (користувацькі входи позначені як ненадійні). Деякі методи захисту від XSS не застосовують ніяких обмежень на введення даних Користувачем, замість цього проводиться аналіз зворотного проксисервера для моніторингу вихідних HTTP-відповідей [19]. Такий підхід дозволяє проксі-серверу аналізувати повністю

згенеровану веб-сторінку і, виходячи з попередніх знань про передбачувану структуру веб-сторінки, або віддавати відповідь відвідувачеві веб-сайту, або повністю відмовлятися від відповіді при виявленні атаки. Такі підходи можуть здатися простішими у впровадженні через відсутність необхідності дезінфекції введених користувачем даних. Однак складність зводиться до отримання точного опису передбачуваної структури веб-сторінки.

Захист від вбудованих в SVG атак. Зростаюча складність сучасних веб-технологій потенційно може призвести до більших ризиків. Можливість включення складних та інтерактивних об'єктів, таких як SVG-зображення або Flash-вміст, призводить до більшої кількості шляхів атаки.

Враховуючи, що зображення SVG можуть містити код JavaScript, може бути небезпечно приймати та вставляти SVG з невідомих джерел. Одним із можливих заходів захисту може бути перевірка SVG проти схеми XML.

Однак, як показують автори, такої перевірки недостатньо, оскільки XML-схеми орієнтовані не на зміст, а на структуру XML-документа. Іншим підходом є очищення SVG, яке передбачає видалення всього потенційно шкідливого вмісту з файлу зображення, мінімізуючи Візуальні наслідки. Очищення досягається за рахунок білого списку дозволеного контенту. Даний підхід, схоже, працює досить успішно, тому його також можна розглядати на рівні HTML специфікації або на рівні браузера. Наприклад, поточна специфікація W3C HTML-5 чітко вказує на те, що інтерактивний вміст SVG не повинен виконуватися браузером. Можливим рішенням буде обмеження веб-браузерів від автоматичного виконання інтерактивного контенту.

Однак різноманітність браузерів, доступних на ринку (і, можливо, з міркувань сумісності), ускладнює реалізацію таких обмежень на практиці. Навіть якщо таке рішення буде всебічно прийнято, воно перенесе проблему в сферу соціальної інженерії, оскільки така атака все одно може бути успішною, якщо

користувачі будуть дозволяти вручну, обманним шляхом, виконання заблокованого інтерактивного контенту.

Моніторинг вхідних даних. Такі засоби захисту припускають, що спостереження за введенням даних Користувачем є достатнім для виявлення деяких типів атак. Вони вважають додаток "чорним ящиком" і зосереджені лише на вхідних даних.

Існує три типи часто використовуваних входів в HTTP-протоколі-це Параметри HTTP GET, параметри HTTP POST і HTTP заголовки. Інтуїція таких підходів полягає в тому, що деякі характеристики відрізняються для вхідних даних, що викликають атаки та вхідних даних, що надходять під час звичайного щоденного використання [20].

Таким чином, Користувальницькі входи постійно контролюються і аналізуються, якщо виявлений нестандартний вхід, тоді виникає тривога. Деякі підходи можуть додатково враховувати інші фактори, такі як структура DOM веб-сторінки. На відміну від підходів на основі зворотного проксі, моніторинг вхідних даних дозволяє виявляти потенційні атаки на більш ранньому ЕТАПІ, що дозволяє як виявляти, так і запобігати атакам. Ключова проблема для підходів моніторингу введення користувача полягає у виборі правильних характеристик для спостереження. Часто використовується кілька характеристик для підвищення рівня виявлення та зменшення помилкових спрацьовувань. У більшості випадків, немає ніякої очевидного, суворого відмінності між» звичайними «і» шкідливими " вхідними даними користувачів; таким чином, вибір характеристик, для спостереження — це, в якійсь мірі, припущення. Система захисту може бути розроблена для моніторингу багатьох факторів, сподіваючись, що неочевидна кореляція зі шкідливими вхідними даними буде виявлена автоматично. Характеристики спостереження не вибираються випадковим чином, а скоріше засновані на неявних припущеннях про значення характеристики, виведеному з попереднього досвіду. Якщо відомо, що характеристика відрізняється для деяких

атак, розумно включити цю характеристику до списку факторів, які слід враховувати. Проте такі припущення можуть бути не вірними або не достатніми для розгортання в реальному світі. Деякі приклади таких значущих вхідних характеристик включають вхідну довжину, розподіл вхідних символів, відсутність введення тощо.

Наприклад, аналіз довжини вхідних даних може бути виправданий тим фактом, що атаки переповнення буфера вимагають довших вхідних даних, щоб перевищити довжину виділеного буфера. Необхідність аналізу розподілу вхідних символів пояснюється тим фактом, що деякі типи атак (наприклад, SQL-ін'єкції) можуть включати використання рідкісних символів, які зазвичай не зустрічаються при повсякденному використанні. Для спрощення розгортання в різних системах, стадія навчання використовується для кожної програми, щоб дізнатися, що таке звичайний користувальницький введення. Система захисту в основному дізнається, яку інформацію користувачі зазвичай надають під час щоденного використання. Ця зібрана інформація згодом використовується на етапі моніторингу в режимі реального часу, щоб визначити, чи в даний час введення користувачем відрізняється від «звичайних» даних. Однак визначення порогового значення різниці для виникнення тривоги є нетривіальною проблемою, яка не має точної відповіді, що зазвичай призводить до ймовірнісних результатів виявлення. Загалом, такі підходи страждають від тих самих проблем, які притаманні підходам, заснованим на навчанні. Наприклад, забезпечення чистого навчального середовища, що охоплює всі сценарії використання та керує неточними результатами виявлення.

Захист від забруднення HTTP параметрів. Такі засоби захисту спрямовані на обхід або виявлення хоча б деяких типів атак з використанням параметрів HTTP. Атака на зараження параметрів HTTP можлива, оскільки специфікація протоколу HTTP визначає лише структуру вхідних параметрів (наданих користувачами), не

накладаючи достатніх обмежень на фактичні імена та значення параметрів. В [21] автори провели аналіз HTTP запитів і навели приклади захисту параметрів в них.

На нижчому рівні HTTP-запит, отриманий сервером, це просто рядок байтів, який повинен бути проаналізований програмним забезпеченням для створення зручної структури даних, придатної для використання веб-додатком. Така попередня підготовка зазвичай включає створення словникової структури, що містить передані параметри. Виникає проблема, якщо запит HTTP містить два або більше параметрів з однаковим іменем. Хоча таке виникнення не суперечить визначенню протоколу HTTP, структура даних словника не може містити один і той же ключ більше одного разу. Отже, залежно від реалізації, деякі параметри можуть бути повністю втрачені і ніколи не потраплять у веб-додаток. Зловмисники можуть формувати HTTP-запити, що мають кілька параметрів GET з тим же ім'ям, що призводить до неоднозначностей на стороні сервера [22]. Деякі серверні програми вибирають першу появу, а інші вибирають останню.

Атака на забруднення параметрів HTTP стає можливою, якщо функція дезінфекції застосовується лише до одного з переданих параметрів, тоді як це інша подія, яка фактично використовується для генерації відповіді. Проблема, пов'язана з атаками за допомогою параметрів http-протоколу, по суті є неправильним припущенням про те, як параметри HTTP GET або POST формуються в HTTP-запит і передаються на сервер. Наприклад, навіть якщо законний веб-браузер ніколи не створює HTTP-запит, що містить кілька параметрів з однаковим ім'ям, ніщо не заважає зловмиснику створювати подібні шкідливі запити вручну. Це означає, що веб-сервер не може встановлювати або приймати будь-які обмеження на отриманий Нттрзапит. Хоча можна створювати (або змінювати існуючі) веб-програми, щоб бути стійкими до атак забруднення HTTP, такі модифікації повинні сильно залежати від прикладного середовища (наприклад, приклад версії веб-сервера або використовуваної мови сценаріїв). Різноманітність рамок для веб-розробки, доступних на ринку, може ще більше ускладнити такі модифікації.

Загалом, атаки забруднення HTTP значною мірою залежать від помилок та невідповідностей, присутніх на іншому рівні, ніж сам веб-додаток.

Таким чином, універсальне рішення проблеми включатиме роз'яснення веб-стандартів (таких як процедури обробки HTTP GET / POST). Однак це все одно зажадає виправлення існуючих інструментів вебразработки і середовища виконання.

Ізоляція користувальницьких входів. Такий захист передбачає, що відокремлення вводу користувача від інших даних, що обробляються додатком, є достатнім для виявлення або обходу деяких типів атак. Хоча зосередження уваги на введенні даних користувачами є вигідним з точки зору раннього виявлення, відсутність розподілу контексту між різними видами даних (наприклад, введення користувачем та вміст, що генерується додатком) та кодом ускладнює аналіз введених даних користувачами.

Результат, створений веб-додатком, - це просто рядок байтів, який не містить семантичної чи структурної інформації. Насправді згенерований результат може бути не HTML-сторінкою, а стисненим zip-файлом, наприклад. Хоча у випадку, якщо результатом є веб-сторінка, синтаксичний аналіз може дозволити розділити різні структурні частини рядка, такі як JavaScript, HTML або CSS блоків. На жаль, жодна структурна інформація про джерело підрядка не зберігається ніде на самій створеній сторінці [23]. Оскільки довіра до повернених даних в першу чергу базується на джерелі даних, додавання метаінформації для відокремлення вводу користувача від виводу веб-програми може дозволити розрізнити ці дані при аналізі Згенерованого рядка (наприклад, веб-сторінки або запиту SQL). Без додаткового потоку даних Така метаінформація повинна зберігатися як частина самого рядка. Використання набору захисних символів для оточення користувацьких рядкових частин є природним підходом до розділення даних. Проблема такого підходу полягає в тому, що захисні Символи є частиною одного і того ж символьного домену. Якщо символи, використовувані для поділу,

відомі зловмисникові, він може вільно підставити такий же набір символів для того щоб отримати управління над перемиканням контексту.

Зробимо висновок, що при розробці різних веб-додатків використовуються різні інструментальні засоби і мови програмування. Зокрема, широко поширені динамічні мови програмування, такі як PHP, Python і т.п. Спостерігаючи загальну картину, можна відзначити що існують деякі подібності між різними, на перший погляд, типами атак. Наприклад, XSS-атаки за своєю основою дуже схожі на атаки SQL-ін'єкціями. Обидва варіанти можливі через невідповідності контексту.

Проблема полягає у відсутності розділення даних та (іноді декількох) потоків коду. Неправильне тлумачення контексту даних також можна пояснити різницею сприйняття між розробником та веб-браузером.

Приклад цієї різниці можна розглянути на деякому фрагменті коду вебсторінки:

 Welcome !

Браузер буде інтерпретувати даний фрагмент як код, який має чотири переходи контексту.

Таблиця 2.1

Представлення кода браузером

	HTML: розмітка
Welcome	1. Текст: повідомлення
	2. HTML: зображення
!	3. Текст: повідомлення
	3. Текст: повідомлення

Однак Розробник даного коду веб-сторінки передбачає, що фрагмент буде мати тільки два переходи контексту.

Таблиця 2.2

Представлення кода разробником

	HTML: розмітка
Welcome	1. Текст: повідомлення
	Текст: ім'я користувача
!	Текст: повідомлення
	2. HTML: розмітка

Першим передбачуваним переходом є зміна контексту з HTML розмітки на статичний текст, а другим — Контекстний перехід назад до HTML розмітки, після закінчення статичного тексту. Однак, якщо зловмиснику на веб-сторінці вдається ввести замість законного імені користувача, з'являються два непередбачених переходу. Наслідки можуть відрізнятися залежно від фактичного типу переходів контексту. Наприклад, несподіваний перехід зі статичного тексту в HTML можна, можливо, розглядати як XSS атаку. Незважаючи на кореневі подібності між атаками XSS та ін'єкціями SQL, може бути важко застосувати той самий захист від даних видів атак через технічні труднощі. Наприклад, виявлення XSS-атак може виконуватися шляхом аналізу HTTPответов, але для атак SQL-ін'єкціями необхідні інші джерела, такі як аналіз PHP коду або моніторинг запитів до бази даних під час роботи [24].

З огляду на деякі контекстні подібності в цих видах атак, можна сказати що, розробка уніфікованого підходу до захисту від них була б дуже корисна. Такі атаки на основі невідповідності контексту можливі через відсутність ізоляції даних користувачів. Хоча й існують деякі підходи забезпечують таку ізоляцію даних шляхом введення додаткового рівня абстракції, який обробляє дані, введені користувачем особливим чином, але на жаль, їх досить важко застосовувати до вже існуючих веб-додатків [25]. Ще одним обмежуючим фактором є необхідність вивчення нових методів програмування, що впроваджуються такими підходами.

Якщо дивитися більш широко, то ізоляцію даних користувача можна розглядати не тільки для веб-додатків, але і для настільних або інших типів, а також мережевих додатків. Наприклад, атаки переповнення стека завершуються успішно, коли надані Користувачем дані неправильно трактуються як код, а потім виконуються. Звичайна захист від атак переповнення стека передбачає використання специфічної для процесора технології, що дозволяє відзначати певні області пам'яті як нездійсненні. Такі додаткові заходи безпеки можуть бути ефективними в деяких випадках, однак використання інтерпретованих мов програмування, які зазвичай зустрічаються у веб-розробці, робить такий захист непридатним. Наміри розробника є важливою частиною розробки програми. Помилки програмування, які зазвичай зустрічаються в додатках, по суті є невідповідністю між намірами розробника та фактичною поведінкою Програми (якщо розробник спеціально не робить помилок).

На підставі проведеного огляду можна зробити висновок, що подальший розвиток методів автоматичного виявлення вразливостей і захисту від них вебпріложеній піде по шляху з'єднання можливостей різних методів з тим, щоб стало можливим: ет охопити максимально широкий клас вразливостей; ето контролювати повноту виявлення вразливостей, щоб за підсумками автоматичного аналізу можна було б дати гарантію відсутності вразливостей заданих класів.

2.2 Аналітичний огляд розподілених атак на відмову

Атака типу «відмова в обслуговуванні» (DoS) - це спроба завдати шкоди, зробивши недоступною цільову систему для звичайних кінцевих користувачів, наприклад веб-сайт або веб-додаток. Зазвичай зловмисники генерують велику кількість пакетів або запитів, які зрештою перевантажують роботу цільової системи. Для здійснення атаки типу «розподілена відмова в обслуговуванні» (DDoS) зловмисник використовує безліч зламаних або контрольованих джерел [1, 2].

На початку розвитку мережевих технологій, DDoS був інструментом тестування пропускної здатності. Але досить швидко зловмисники знайшли можливість використання даної атаки в своїх корисних цілях. Атака типу «відмова в обслуговуванні» є в наборі інструментів зловмисників уже протягом двадцяти років, і вона досі зростає в популярності і розвивається. Зараз даний тип атаки використовуються для отримання вигоди різними способами: шантаж, припинення функціонування основних вузлів мережі; припинення працездатності веб-сервісів; в бізнес або політичній конкуренції тощо. Вплив може варіюватися від незначного роздратування адміністратора мережі та незначних порушених сервісів до повного припинення функціонування цілих веб-сайтів, додатків або навіть цілого бізнесу, який працює в автономному режимі. Це у свою чергу може призвести до втрати репутації або значних фінансових втрат. Також даний тип атак може бути відволікаючим фактором, для здійснення складніших атак, для викрадення конференційної інформації. Можна виділити основні мотиви зловмисників які використовують DDoS [8]:

- Конкуренція.

На даний момент досить популярною є послуга проведення DDoS на замовлення. Тобто, при конкуренції, фірма, яка не погоджується з конкурентами, 7 просто формує завдання для хакера паралізувати систему, з якою працюють

конкуренти, або паралізувати роботу зовнішні або внутрішніх ресурсів конкурентоспроможної компанії. В результаті чого, організовується DDoS атака на визначений термін і з визначеною силою.

● Шахрайство.

Хакери самостійно організовують DDoS атаки з власної мережі заражених комп'ютерів, які дозволяють блокувати роботу невеликих та середніх системи. Якщо користувач не встановив захист від DDoS атаки, то хакер може повною мірою паралізувати роботу системи, і шантажувати на певну суму для розблокування. Часто, звичайні користувачі, погоджуються на умови хакерів. Це обумовлене тим, що просто цінність даних вища, ніж сума, вказана хакером.

Зі сторони мережі DDoS виглядає як різке збільшення трафіку. В загалом виділяється дві методики, які найбільш часто використовуються зловмисниками:

● руйнівні атаки – призначені для того щоб сегмент мережі став цілком недоступним: зависає, знищується, видаляється операційна система. Цей тип атаки стає доступним через вразливості, які є в програмному забезпеченні;

● атака на ресурси системи – в цій атаці велика кількість пустих або неправильних запитів відправляється на вузол мережі або веб-додатку, що тягне за собою значне зменшення пропускної здатності мережі або продуктивності системи та мережевого обладнання.

Основні види DDoS атак [8]:

1. Атака на перевищення ліміту системних ресурсів. 8 Дії зловмисників націлені на вичерпання системних ресурсів такі як оперативна і фізична пам'ять, процесорний час тощо.

2. Недостатня перевірка даних користувача. Недостатня перевірка даних користувача може призвести до нескінченного або тривалого циклу їх обробки, що призводить до підвищеного та тривалого споживання процесорних ресурсів або виділенню великих обсягів пам'яті, аж до її вичерпання.

3. Атаки другого роду. Це атаки, які націлені на ділянки мережі в яких неправильно ввімкнено або налаштовано системний захист, тим самим, це призводить до недоступності конкретних ресурсів.

4. HTTP-флуд. Нападник відсилає невеликі http-пакети, які заставляють сервер у свою чергу відсилати пакети-відповіді, розміри яких значно більші. Тим самим зловмисник має великі шанси наситити смугу пропускання жертви безглуздими пакетами що викликає відмову в роботі сервісу. Для того, щоб відповідні пакети не входили в опис заборони в обслуговуванні, нападник замінює свій мережевий адрес на легітимні у вузлу мережі.

5. ICMP-флуд (Smurf-атака). Цей тип атаки є одним з найнебезпечніших. У ньому по широкомовній адресі зловмисник відправляє підроблений ICMP- пакет, в якому адреса відправника, змінюється на адресу жертви. Усі вузли, які отримали такий пакет, присилають відповідь на цей запит. Для такого виду атаки зазвичай використовують велику мережу, щоб у комп'ютера-жертви не було жодних шансів. Таким чином, запит, відправлений через мережу в 1000 комп'ютерів буде посилений в 1000 разів.

6. UDP флуд (атака Fraggle).

Цей тип атаки є аналогом ICMP флуду, але замість ICMP пакетів використовуються UDP пакети. На сьомий порт жертви відправляються echo- 9 команди по широкомовному запиту. Після чого підміняється ip-адреса зловмисника на ip-адресу жертви, яка отримує безліч повідомлень у відповідь, що призводить до насичення смуги пропускання і відмови в обслуговуванні жертви.

7. SYN-флуд.

Цей вид атаки ґрунтований на спробі запуску великого числа одночасних TCP- з'єднань через відправлення SYN-пакету з зворотним адресом якого не існує. Після декількох спроб відіслати у відповідь ACK-пакет на недоступну адресу більшість операційних систем вставляє невстановлене з'єднання в чергу. І тільки після n-ої спроби закривають з'єднання. Оскільки потік ACK-пакетів дуже

великий, незабаром черга виявляється переповнене, і ядро дає відмову на спробу відкрити нове з’єднання. Схема даної атаки показан.

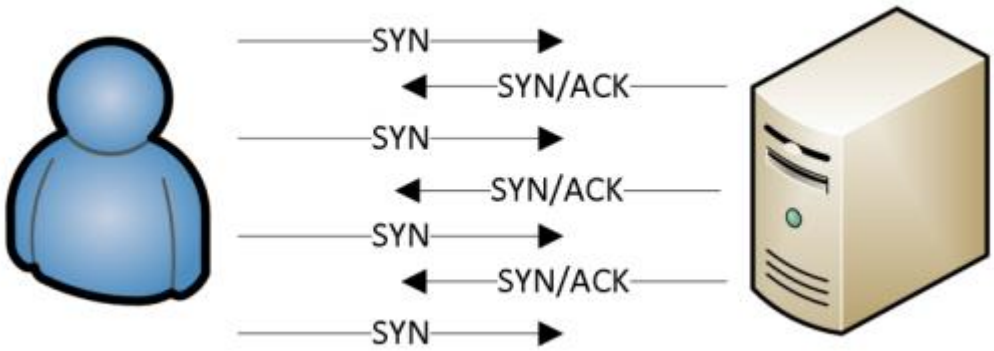


Рис. 2.1 - SYN-флуд

8. Відправлення важких «пакетів». Зловмисник посилає пакети серверу, які не насичують смугу пропускання, а витрачають увесь його процесорний час. Відповідно, в системі може пройти збій і легальні користувачі не зможуть отримати доступ до необхідних ресурсів.

9. Переповнювання сервера лог-файлами.

При неправильній ротації лог-файлів і неправильно встановленій системі квотування зловмисник може відправляти великі за об'ємом пакети, які незабаром займуть усе вільне місце на жорсткому диску сервера.

10. Помилки програмного коду. Досвідчені організатори DDoS атак, повністю розібравшись в структурі жертви, пишуть програми-експлоїти, які дозволяють атакувати складні системи комерційних підприємств і організацій. В основному, це помилки в програмному коді, які дозволяють виконувати неприпустимій інструкції або винятковій ситуації, яка може привести до аварійного завершення служби.

11. Недоліки в програмному коді. Зловмисники шукають помилки в програмному коді яких-небудь програм або операційної системи й примушують їх обробляти виняткові ситуації, які вони обробляти не вміють, що призводить до падіння ядра або краху усієї системи в цілому. Зараз, здебільшого використовується комбінований з декількох видів DDoS атак, шо робить протидію проти них складнішою. За даними статистики [9] найпопулярніший тип атаки HTTP-Flood (80%).

Зараз кіберзлочинці мають безліч технології для проведення цієї атак, які постійно розвиваються. У 54% дана атака комп'ютери ботнету посилають данні конкретній сторінці веб-ресурсу; наступне місце 22% атака на поля авторизації; на третьому місці 13% атака спрямована на велику кількість завантаження файлів з веб-ресурсу. І лише в одному випадку з 10 проводяться сладніші атаки, коли зловмисники намагаються замаскувати дії ботів під поведінку справжніх користувачів.

Друге місце (10%) за популярністю це атака типу UDP-Flood. Зловмисники, які здійснюють такі атаки, покладаються на величезну кількість пакетів, тобто генерують за допомогою ботнету велику за кількістю пакетів, але невеликі за розміром (до 64 байта) пустих пакетів.

На наступних місцях атаки типу SYN-Flood а ICMP-Flood відповідно. Описана статистика зображена на діаграмі.

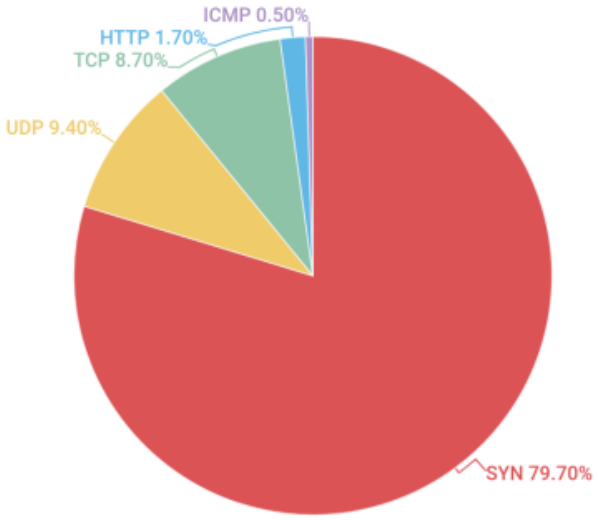


Рис. 2.2 – Типи DDoS атак за популярністю

Зазвичай мережа зловмисника, з якої відбувається DDoS атака, має три рівні (рис 2.3). Назва такої мережі «кластер DDoS». Верхній рівень кластера складається с декількох комп’ютерів, з яких відбувається початок атаки та надалі іде координація дій інших учасників. Другий рівень – складається з десятків комп'ютери, які вже надають сигнали про атаку на наступний рівень кластера. На третьому рівні — зазвичай DDoS-зловмисники покладаються на ботнети — колекції мережі інфікованих зловмисними програмами, які централізовано контролюються. Ці заражені кінцеві точки, як правило, є комп'ютерами та серверами, але все частіше є IoT та мобільними пристроями. Зловмисники збирають ці системи шляхом виявлення вразливих систем, якими вони можуть потенційно заразитися за допомогою фішинг-атак, зловмисних атак та інших методів масового зараження. Все частіше зловмисники також орендують цих ботнетів у тих, хто їх побудував [5].

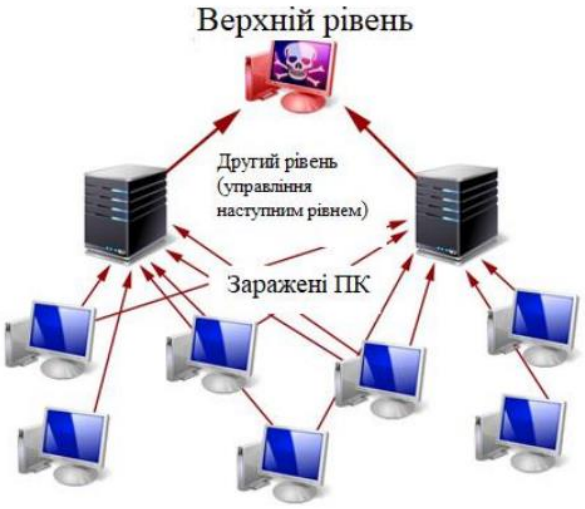


Рис. 2.3 – Структура ботнет мережі

Залучення до ботнету зазвичай відбувається через встановлення на комп'ютер жертви програму, яка не виявляється користувачем в щоденній роботі. Найчастіше зловмисники здійснюють ці дії через:

• зараження комп'ютера через вразливість (помилки в браузерах, поштових клієнтів, програмах перегляду документів, зображень, відео);

• незнання або необачність користувачів, коли небезпечна програма маскується під безпечне;

• несанкціонованого доступу до комп'ютера користувача;

• повний перебір пароля (брутфорс) для отримання доступу прав адміністратора мережі, переважно використовується в локальних мережах.

Через стрімкий розвиток та ріст ботнет мереж за останні роки, складність захисту від зловмисників зростає. Це пояснюється тип що, через використання цієї технології, знаходження організаторів атаки стає досить складно. Через це, ці напади стають більш популярні, та зазвичай проводять їх через орендовані ботнети. Очікується, що ця тенденція продовжиться. 13 У багатьох країнах реалізація DDoS -атаки є злочином, але досить рідко виходить впіймати організаторів, а ще рідше виконавців. В наші дні організацією DDoS атак

займаються не одна людина, а добре організовані зловмисна групи. Через їх організаційну та географічну віддаленість, їм вдається вдало протидіяти правоохоронним органам.

2.3 Алгоритм захисту від DDoS атак заснований на IP фільтрування

Даний алгоритм є практичною схемою для захисту від розподілених атак типу відмова в обслуговуванні (DDoS) атак, заснованих на фільтрації IP-адрес джерел [11].

В цьому методі граничний маршрутизатор зберігає історію всіх правомірних адрес IP, які раніше з'являлися в мережі. Коли граничний маршрутизатор перевантажений, ця історія використовується для рішення, чи слід приймати пакети від конкретного IP. На відміну від інших алгоритмів для захисту від DDoS атак, цей метод працює добре при високорівневій розподіленій DDoS атаці, тобто з великої кількості джерел.

Основний принцип цього алгоритму, який дає можливість використовувати в захисті від DDoS атак полягає в тому, що трафік DDoS атаки прагне використовувати підроблені випадкові адреси джерел, щоб замаскувати свою істинну адресу. Таким чином, залишається, розрізнити легальну IP-адресу від випадкової злочинної IP-адреси.

Алгоритм використовує механізм під назвою History-based IP Filtering (HIF), граничний маршрутизатор приймає пакети згідно з попередньо побудованої бази даних IP-адрес. База даних IP-адрес має бути заснована на основі попередньої історії маршрутизатора [11]. Існує декілька евристичних методів, щоб зробити базу даних IP-адрес точної і надійної.

Схема складається з трьох частин. Вона є дуже надійною в захисті від розподілених атак типу відмова в обслуговуванні. Цей механізм не потребує глобальну співпрацю з іншими веб-ресурсами. Дана методика може бути

застосована до широкої різноманітності видів трафіку, і не вимагає особливого налаштування вручну.

На ранній стадії DDoS атак, трафік обмежується за певним типом для прикладу, Smurf-атаки використовують ICMP пакети. Таким чином, можна легко блокувати атаку трафіку згідно з типу протоколу. На жаль, останні дослідження показують, що інструменти DDoS атак постійно розвиваються, та здатні створювати пакети з підробленими випадковими адресами джерела, вихідного порту і навіть генерувати корисне навантаження пакетів [13]. Це значно ускладнює процес фільтрації та вимагає розширеного трафіку методики моделювання. Цей метод дає практичне розв’язання цієї проблеми, відрізняючи хороші та погані пакети порівнюючи їх з попередньою історією. На відміну від алгоритмів фільтрації IP-джерела, які намагаються охарактеризувати атакуючий трафік, можна схарактеризувати замість цього звичайний трафік. Ця методика відноситься до типу виявлення аномалій. Дослідження показали, що більшість всіх IP-адрес нормального трафіку відправляли запити раніше. На основі статистичного аналізу інтернет-трафіку, зібрані в мережі середнього розміру, було виявлено, що велика частина IP-адрес, що входять в інтернет-трафік постійно з'являються на основі щоденних спостережень [10]. З цього можна зробити висновок, що надійна особливість ідентифікації нормального трафіку це те, що він регулярно з'являвся на веб-сайті.

Фундаментальною проблемою безпеки для IP-мереж є те, що немає схеми аутентифікації джерела IP пакета. Якщо маршрутизатор або веб-сервер мав би можливість визначити, які адреси є законними, проблема аутентифікації могла бути вирішена. Щоб відрізняти хороші пакети від поганих пакетів, маршрутизатор або веб-сервер повинен вчитися з історії попередніх здійснюваних підключень до мережі. Даний метод полягає в тому, щоб зробити запис всіх IP-адрес попередніх успішних мережевих з'єднань, та зібрати базу даних IP-адрес. Тоді, коли наша мережа або веб-сайт відчувають високий рівень скупчення, ми можемо

відмовитися від пакетів чию адресу джерела не з'являється в нашій базі даних IP-адрес. У разі DDoS атаки, це означає, що ми маємо високу точність фільтрації пакетів атаки.

В основному є дві ключові ролі техніки HIF. Перша полягає в розробці правил розрізнення законних пакетів від шкідливих пакетів шляхом перевірки IPадреса вхідного пакета в базі даних IP-адрес (IP Address Database (IAD)). Друге, як використовувати хеш способи побудови ефективних пошук IP таблиці. Нехай весь трафік з однією IP-адресою джерела, характеризується як одна IP-адреса.

Нехай Si = {si 1, si 2, s i 3, si 4 … si n,}, описує колекцію всіх законних IPадрес, які з'явились у мережі на дату i, де | Si | = ni. Нехай Fk={f1, f2, f3, f4, …, fm,} описує колекцію всіх частих законних IP-адрес від дати l до дати k де | Fk |=m. цьому розділі ми надаємо два альтернативні правила для визначення «частих». Нехай A={a1, a2, a3, a4, …, an } описує IP-адреса, які використаються в розподіленій атаці на відмову в службі. Оскільки існує стабільна група IP-адрес, які регулярно відвідують мережу, а DDoS атаки використовують випадкові підроблені IPадреси, як ми проаналізували в попередньому розділі, для спостереження трафіку k днів наступне співвідношення: $|S1 \cup S2 \cup S3 \ldots \cup Sk| < \sum k\ i{=}1\ ni \ll |A|$. Очевидно, що ($F\ k \subseteq S1 \cup S2 \cup S3 \ldots \cup Sk$). Ми прагнемо проаналізувати розподіл IP-адреси в межах $S1 \cup S2 \cup S3 \ldots \cup Sk$ та використання статистичного підходу для розробки порогу для визначення частого збору користувачів F. Таким чином $Pnormal = |F \cap Si|/|Si|$ являє собою відсоток нормальних IP-потоків, прийнятих на дату j (j> k) і $Pddos = |F \cap A|/|A|$ представляє відсоток допущених IP-адрес атаки. В ідеалі ми хочемо, щоб $Pnormal$ був l, а $Pddos$ був 0.

Визначаємо базу даних IP-адрес (IAD) як цілу колекцію IP-адрес, що з'явились у попередньому періоді експертизи, наприклад, один місяць. Для

фільтрування IP на основі історії (HIF) дуже важливо отримати точну та ефективну базу даних IP-адрес (IAD). Існує два різні правила, щоб визначити, що часто є IP-адресою. Перше правило вважає IP-адресу частою залежно від кількості днів, коли вона з'явилася. Нехай p1(d) представляють колекцію унікальних IP-адрес, які з'явилися щонайменше D дні.

Нехай f1(d) являє собою відсоток хорошого трафіку, який проходить при використанні p1(d) як IAD. Друге правило - кількість пакетів на IP-адреси. Нехай p2(d) являє собою сукупність унікальних IP-адрес, що мають принаймні пакети u. Нехай f2(u) являє собою відсоток хорошого трафіку, який проходить при використанні p2(u) в якості IAD.

На практиці ми хочемо щоб |p1(d)| і | p2(u)| невеликими, щоб зменшити потребу в пам'яті для збереження IAD, і ми хочемо f1(d) і f2(d) бути великими, щоб захистити законний трафік. У цьому алгоритмі залучені два параметри проектування: кількість днів (d) та кількість пакетів на IP-адресу (u). Ми можемо налаштуват«Транспортний протокои параметри відповідно до різних мережевих умов та отримати більш точну та ефективну IAD, комбінуючи всі ці два правила, які можуть бути показані як:

$$Fc = p1(d) \cap p2(u) \quad (2.2)$$

Після того, як ми заповнили IAD, важливо швидко отримувати Процес пошуку IP-адреси, особливо коли |F| великий. Для цього можна використовувати хеш функцію для створення єдиної хеш-таблиці.

Міжмережеві екрани (МЕ) - невід'ємний елемент забезпечення мережевої безпеки. Застосування МЕ при побудові систем захисту інформації є не тільки частою практикою, а й обов'язковою вимогою ряду нормативних документів в області інформаційної безпеки. Політика міжмережевого екранування (визначення того, які мережеві потоки повинні бути пропущені, а які заблоковані) реалізується в МЕ набором правил фільтрації мережевого трафіку.

Структура представлення наборів правил фільтрації переважної більшості сучасних МЕ є лінійний список. Для кожного пакета, що проходить через МЕ здійснюється послідовний перебір правил з даного лінійного списку. Після знаходження першого правила фільтрації, яке повністю задовольняє відповідний мережевий пакет, перебір наступних в списку правил зупиняється. Таким чином, чим вище за списком буде стояти відповідне правило фільтрації, тим швидше МЕ прийме рішення про обробку мережевого пакета. Очевидно, що чим вище за списком будуть розташовуватися найбільш часто спрацьовані правила фільтрації, тим менше ресурсів МЕ буде витрачатися на послідовний перебір лінійного списку правил. У свою чергу, від інтенсивності використання обчислювальних ресурсів МЕ багато в чому залежить його пропускна здатність.

Запропонований алгоритм впорядкування правил фільтрації мережевого трафіку, дозволяє впорядковувати правила фільтрації в наборах правил за частотою їх спрацьовування. Мета помістити найбільш часто використовувані правила на вершину списку, щоб мінімізувати витрачання ресурсів системи на послідовну перевірку великої кількості правил. Шляхом оптимізації даного процесу можлива істотна економія машинного часу, і зниження навантаження на пристрої, що здійснює функції брандмауера.

Основу алгоритму становить технологія пасивного аналізу мережевого трафіку. Вона дозволяє побудувати статистичну модель трафіку який проходить через МЕ і на її основі розрахувати частоту спрацьовування кожного правила фільтрації з набору правил. Блок-схема алгоритму представлена на рис. 2.4. На вхід алгоритму подається мережевий трафік, який перетворюється в масив мережевих пакетів P, з якого потім відновлюється інформація про TCP з’єднання або UDP-з’єднання. Ця інформація зберігається в окремому масиві даних C. Для кожного з'єднання в даному масиві зберігається наступна інформація: IP-адреса клієнта, номер порту клієнта, IP-адреса сервера, номер порту сервера, який використовується протокол транспортного рівня і кількість переданих в рамках

з'єднання мережевих пакетів. На основі масиву C формується зведений масив мережевих з'єднань C* шляхом об'єднання записів з однаковими значеннями полів «IP-адреса клієнта», «IP-адреса сервера», «№ порту сервера» і «Транспортний протокол». Даний масив потім сортується за спаданням значення поля «Кількість пакетів».

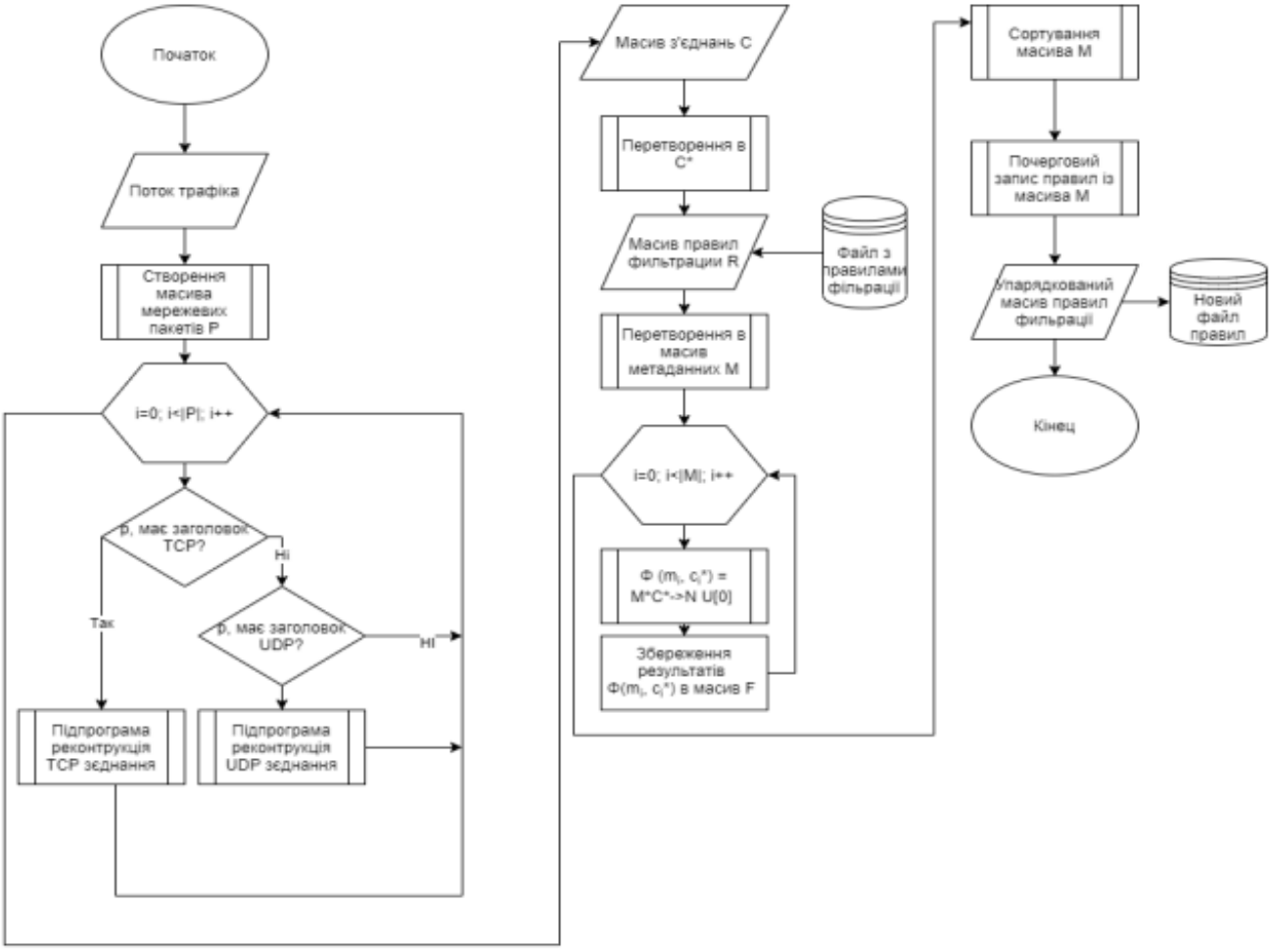


Рис. 2.4 - Блок-схема алгоритму впорядкування правил фільтрації на основі статистичного аналізу

Ще одним вхідним параметром алгоритму є масив правил фільтрації брандмауера R, який необхідно впорядкувати. Значення даного масиву передаються в функцію перетворення в метадані, де здійснюється формування

масиву метаданих M. Для кожного правила з R в масив M зберігається наступна інформація: порядковий номер, IP-адреса клієнта, IP-адреса сервера, який використовується протокол транспортного рівня, номер порту сервера, відповідне текстове представлення відповідного правила фільтрації.

Далі масив метаданих M і зведений масив мережевих з'єднань C* надходять на вхід функції розрахунку значення Ф(m, ci *). В даній функції для кожного правила яке є належить до M на підставі зведеного масиву мережевих з'єднань C* здійснюється розрахунок функції Ф, значенням якої є невід'ємне ціле число, яке дорівнює кількості спрацьовувань відповідного правила:

$rj \in R: Ф: M \times C * \rightarrow N \cup \{0\}$

На основі значень функції Ф процедура сортування здійснює пошук такої перестановки елементів масиву M, при якій відповідні їм значення функції Ф розташовуються в порядку зменшення: $Ф(mi) \geq Ф(mj) \geq ... \geq Ф(mk)$. А на основі даної перестановки вже здійснюється формування впорядкованого масиву правил R.

Таким чином, застосування алгоритму впорядкування правил фільтрації міжмережевих екранів в наборах правил фільтрації дозволяє знизити навантаження на апаратну складову міжмережевих екранів і тим самим підвищити їх пропускну здатність. При цьому ефект від упорядкування правил буде найбільш помітний на високонавантажених міжмережевих екранах з великою кількістю використовуваних правил фільтрації (тисячі і десятки тисяч правил).

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦЯ

3.1 Програмні засоби для розробки

Існують різні способи побудови власних комплексів протидії від DDoS атак. У цій роботі були застосовані ефективні методи захисту DDoS з використанням iptables. Попри те, що можна багато зробити з iptables для блокування DDoS атак, не існує способів для апаратних брандмауерів, щоб виявити та зупинити великі потоки DDoS. Ми охоплюватимемо лише захист від атак на основі TCP, оскільки більшість атак на основі UDP - це посилені атаки відбиття, які вичерпують мережеву інтерфейсу карту будь-якого загального сервера. Єдиний підхід щодо пом'якшення, який має сенс проти цих типів атак, - це блокувати їх або в основній мережі або навіть в мережі оператора. Якщо такі атаки зможуть дістатися до сервера, проти цих багато габаритних атак не можна зробити нічого, окрім переміщення до нової захищеної від DDoS мережі.

DDoS атаки є складними. Існує багато різних типів DDoS, і майже неможливо підтримувати захист проти всіх них. Але ми можемо пом’якшити майже будь-яку атаку DDoS на основі TCP.

Важливу роль при налаштуванні безпеки веб-серверу відіграють інструментальні засоби. Для створення брандмауера, та блокування небажаних IP було вирішено використовувати Iptables.

Iptables - це додаток/програма, яка дозволяє налаштовувати міжмережевий екран наданий ядром Linux, в якому користувач має можливість додавати або видаляти відповідні його вимогам правила безпеки. Iptables використовується для IPv4, ip6tables - для IPv6. Зазвичай, правила iptables налаштовує системним адміністратором. Важливо: щоб застосовувати правила iptables, потрібні rootправа [7]. Iptables можна використовувати для фільтрації певних пакетів, блокування джерел або портів призначення та IP-адрес, пересилання пакетів через NAT та

безліч інших речей. Найчастіше він використовується для блокування портів призначення та вихідних IP-адрес [7].

Серед інструментальних засобів для створення скриптів на веб-сервері найбільш популярні такі мови програмування, як PHP, Bash, Python, які підтримуються майже всіма хостинг-провайдерів.

Мова програмування PHP була обрана як інструментальний засіб для розробки. Вибір пояснюється тим що він досить простий у використанні і дозволяє швидко і ефективно писати різного роду скрипти, і на відміну від Bash, підходить для досить складних завдань. Php - скриптова мова програмування, яка була розроблена для динамічної генерації HTML-сторінок на стороні вебсервера. PHP є однією з найпопулярніших мов, використовуються у сфері веброзробок. PHP — проект відкритого програмного забезпечення [14].

PHP ще зручний і тим, що він дозволяє просто додавати і налаштовувати вхідні параметри для скриптів. Також PHP зручний для роботи з стандартними потоками введення \ виводу Linux, для цього використовуються константи STDIN, STDOUT, STDERR. По суті, ці потоки нічим не відрізняються від файлових потоків і працювати з ними так само просто. В якості засобу діагностики було обрано tcpdump. Ця утилита допоможе виявити практично будь-які уразливості в роботі сайту веб-сервера. А саме своєчасне виявлення вразливостей може запобігти більшості DDoS атак, і тоді вам навіть не потрібно витрачати ресурси на захист від них. tcpdump - це потужна утиліта unix, що дозволяє перехоплювати і аналізувати мережевий трафік, що проходить через мережеві інтерфейси.

3.2 Конфігурації веб-сервера для ефективного протидії DDoS атакам

Поширена помилка при налаштуванні веб-сервера полягає в тому, що люди не використовують оптимізовані настройки ядра, для більш ефективної пом’якшення наслідків від DDoS атак. Необхідно звернути більшу увагу на такі налаштування ядра, як:

• час, що проводиться сокетом в TCP-фазі FIN-WAIT-2;

• розмір приймального буфера сокетів TCP;

• не допускати підозрілий трафік;

• ті ж значення для інших буферів.

За замовчуванням ОС Debian та інші ОС не в змозі підтримувати величезну кількість з'єднань, які генерують ботнети. Необхідно внести зміни в налаштування ядра, щоб зміцнити захист стеку протоколів TCP/IP. Приклад такої конфігурації наведено в додатку.

Необхідно просто помістити данні налаштування в свій файл /etc/sysctl.conf і застосувати налаштування за допомогою команди sysctl –p. Ці налаштування sysctl.conf допомагають досягти максимальної продуктивності сервера від DDoS, а також збільшити ефективність правил iptables, які були використані у цій роботі.

Актуальні правила анти-DDoS для iptables:

● Блокувати недійсні пакети iptables -t mangle –L 1 PREROUTING -m conntrack -ctstate INVALID -j DROP. Це правило блокує всі пакети, які не є SYN-пакетом і не належать до встановленого TCP-з'єднання.

● Блокувати нові пакети, які не є SYN iptables -t mangle –L 1 PREROUTING -p tcp ! --syn -m conntrack -ctstate NEW -j DROP.

Це правило блокує всі нові пакети (не належать до встановленого з'єднання) і не використовують прапор SYN.

● Блокувати нечасті значення MSS iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW -m tcpmss ! --mss 536:65535 -j DROP.

Це правило блокує нові пакети (лише SYN-пакети можуть бути новими пакетами згідно з двома попередніми правилами), які використовують значення TCP MSS, яке не є загальним.

● Блокувати пакети з прапорами Bogus TCP. Одне з таких правил: iptables -t mangle –L 1 PREROUTING -p tcp -tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP.

Такі правила блокують пакети, які використовують помилкові прапорці TCP. Тобто, TCP-прапорці, які легітимні пакети не використовуватимуть

● Блокувати пакети з приватних підмереж (підроблення).

Один з прикладів такого правила:

sudo iptables -t mangle –L 1 PREROUTING -s 192.168.0.0/16 -j DROP

Ці правила блокують підроблені пакети, що походять з приватних (локальних) підмереж. У інтерфейсі загальнодоступної мережі зазвичай не хочеться отримувати пакети від приватних IP-адрес.

Цей набір правил блокують багато DDoS атак на основі TCP з дуже високою швидкістю пакетів. Додаткові правила захисту які необхідно використовувати [3]:

1. Не застосовувати Windows Server.

На практиці що при DDOS атаці, сайти, що працюють на Windows, поводяться гірше за все. Це пов’язано з тим, що при великій кількості з'єднань сервер перестає відповідати через специфічного пристрою мережевого стека. Намагайтеся використовувати в якості ОС для вашого сервера Linux. Подальші методи захисту від DDoS в цій статті в більшій мірі застосовні саме до цієї ОС.

2. Намагатися не застосовувати Apache.

Веб-сервера Apache вразливий до атак типу DDOS. Це пов’язано з його фундаментальною структурою. Через це зробити його більш безпечним стає складно. Для зменшення впливу від атак цього типу, використовується багато додаткових ресурсів, але набагато простіше буде використовувати інший HTTPсервер.

3. Код 444.

Для того, щоб реалізувати атаку веб-сайт, зловмисники можуть використати його пошукову систему – яка є однією з слабких ланок сайту. Безліч запитів до внутрішнього пошуку може миттєво змусити сайт відмовити в обслуговуванні інших користувачі. Для захисту від такої атаки просто вимкніть функцію пошуку до тих пір, поки не будете впевнені в його безпеці. Підтримка nginx

нестандартного коду 444 дозволяє повністю перекрити з'єднання без будь-якої відповідної віддачі. Лімітувати ресурси в веб-сервері nginx Ресурси які використовуються мають не перевищувати наявні. Часто зловмисники вичерпують ресурс оперативної пам’яті. Щоб цього не сталося треба встановити ліміти на розмір заголовків, та обмежити всі наявні буфери які доступні клієнтам, та на сервері загалом, це дозволяє значно збільшити стійкість від DDoS атак.

3.3 Короткий опис програмної реалізації

Програму для захисту від DDOS атак було реалізовано за допомогою мови програмування PHP у вигляді проекту ddosprotect.php.

Таблиця 3.1

Основні функції ddosprotect.php

Назва процедури	Короткий опис
check_ddos_second_modul	Застосування активного захисту від DDoS атаки використовуючи метод фільтрування по IP
check_ddos_first_modul	Застосування активного захисту від DDoS атаки використовуючи утиліту tcpdump
help_dock	Виведення на екран інформацію щодо використання, та додаткові правила захисту
activate_rules_iptablel	Застосування правил iptable для пасивного захисту

Для початку, нам необхідно дізнатися у користувача деяку інформацію, яка нам потрібна для захисту. Цю інформацію можна передати в якості параметрів, які ми передаємо при виклику скрипта. У PHP є глобальні змінні $argc і $argv. Перша містить кількість вхідних параметрів, а друга - масив з вхідними параметрами. Таким чином, обробка вхідних змінних зводиться просто до обробки масиву.

В залежності від введених параметрів програма буде запускати необхідні модулі:

-f (--first-modul) – запускає перший модуль програми;

-s (-- second-modul) – запускає другий модуль програми;

-h (--how-protected) – виводить на екран інформацію як користуватися; програмою, та додаткову інформацію для адміністратора, щодо захисту від DDoS;

-a (--activate) – активувати набір правил для iptable; За активації правил iptables відповідає наступна функція:

activate_rules_iptablel(){

```
 $ip_list = system($command);
$command = './iptablesrules.sh'; }
```

Всі правила знаходяться в окремому файлі iptablesrules.sh

Перший із запропонованих методів, призначений для захисту від атаки з декількох десятків або сотенних комп'ютерів, частими запросами на веб-сервері. Принцип дії захищених від DDoS атаки - блокується IP-адреса, частування обраного з тим, що більше відомо, чим зазвичай потрібно працювати з вашим сайтом. Захист може бути вбудований в php-код сторінки сайту. Використовується таблиця MySQL для зберігання тимчасових даних.

Отже, створюємо таблицю access_log. Нам потрібно тільки два поля, для зберігання IP-адреси і часу доступу:

Таблиця 3.2

Опис MySQL таблиці

Поле	Пояснення
IP (varchar (15))	IP-адрес користувача
enter_time (int(11))	Запис часу останнього доступу до сервера з одного IP.

Тепер пишемо функцію для визначення занадто активних IP-адрес:

```
function check_ddos_first_modul(){
 $rec_limit = 4; // $time_limit = 1; //
$ip = $_SERVER['REMOTE_ADDR'];
// Підключення до БД:
 if (!$db_connection = mysql_pconnect('localhost', 'root', '')){
die();
}else{
if(!mysql_select_db('ddos_test', $db_connection)){
 die();
}
```

```
}
mysql_query('DELETE FROM access_log WHERE enter_time < '.(time() - ($time_limit * 2)), $db_connection);
mysql_query('INSERT INTO access_log (ip, enter_time) VALUES ('".$ip."', '.time().')', $db_connection);
 if ($result = mysql_query('SELECT enter_time FROM access_log WHERE ip="'.$ip.'" ORDER BY enter_time DESC LIMIT '.$rec_limit)){
while($row = mysql_fetch_row($result)){
 $result_array[] = $row;
}
 }
$rec_count = count($result_array);
if ($rec_count == $rec_limit){
$first_time = $result_array[$rec_count - 1][0];
$last_time = $result_array[0][0];
 if (($last_time - $first_time) < $time_limit){
iptables_ban_ip($ip);
exit();
 } }}
```

Для цього прикладу було створена база даних з ім'ям ddos_test. Якщо вже використовується MySQL на своєму сайті - можете використовувати існуючий обліковий запис.

Таким чином, при кожному зверненні до сторінки сайту в таблицю записується час звернення і IP-адреса клієнта. При цьому кожен раз робиться вибірка з $rec_limit останніх звернень з даного IP, і якщо час між першим і останнім зверненнями в цій вибірці менше значення змінної $time_limit (в секундах) - викликається функція iptables_ban_ip, і даний IP блокується файерволом. В даному прикладі, якщо з однієї IP буде більш, ніж 4 звернення в

секунду - IP блокується. Сама таблиця тимчасових даних постійно очищається від зайвих даних і містить тільки останні звернення.

Функція для запису особливо настирливих IP-адрес в блек-лист брандмауера:

```
function iptables_ban_ip($ip){
$command = 'sudo iptables -A INPUT -s '.$ip.' -j DROP';
system($command);
}
```

Інший метод більш ефективний якщо DDOS атака вже йде. Цей метод використовує tcpdump, для виявлення IP від яких йде не бажаний трафік.

Наприклад, на сервері відключені логи, йде легка ddos атака, ви хочете швидко подивитися масштабність або переконатися, що це ddos атака, а не dos або може це взагалі ніяк не пов'язане із зовнішнім світом, для цього можна використовувати дану утиліту Відкривши tcpdump можна спостерігати список підключень до 80-у порту, чим більше повторних підключень з однакових хостів тим ймовірніше ми зіткнулися з DoS або DDoS атакою. Змінивши порт можна перевірити чи є атака на FTP, SSH або інші сервіси які містяться на сервері. Додавши ключ -n виводяться IP адреси, ключ -c вказує на кількість пакетів для виводу, ключ –w здійснює запис в файл.

```
tcpdump -v -n -w attack.log dst port 80 -c 250
```

Далі потрібно проаналізувати дані через tcpdump, за допомогою grep здійснюємо парсинг результатів, та сортуємо результат наступною командою:

```
tcpdump -nr attack.log |awk '{print $3}' |grep -oE '[0-9]{1,}\.[0-9]{1,}\.[0-9]{1,}\.[0-9]{1,}' |sort |uniq -c |sort –rn > bad_ip.txt
```

Результатом буде - дві колонки, перша кількість підключень, в другій IP. Чим більше підключений для одного IP тім більше шансів, що це бот.

```
tcpdump -nr attack.log |awk '{print $3}' |grep -oE '[0-9]{1,}\.[0-9]{1,}\.[0-9]{1,}\.[0- 9]{1,}' |sort |uniq |sort -rn > bad_ip.txt
```

Результат парсинга можна перенаправити в файл, а потім використовуючи вже створену функцію iptables_ban_ip, заблокувати всі ip зі списку. Оформляємо у вигляді PHP функції:

```
function check_ddos_second_modul(){
$command = 'tcpdump -v -n -w attack.log dst port 80 -c 250'
$ip_list = system($command);
$command = ' tcpdump -nr attack.log |awk '{print $3}' |grep -oE '[0-9]{1,}\.[0-9]{1,}\.[0-9]{1,}\.[0-9]{1,}' |sort |uniq |sort -rn > bad_ip.txt ';
$ip_list = system($command);
$ddos_ip = fopen("/ddos_ip/bad_ip.txt ", "r");
while (!feof($ddos_ip)) {
$buffer_ip = fgets($handle, 20);
iptables_ban_ip($buffer_ip)
}
 }
```

Повний програмний код, розробленої системи захисту від DDoS атак, наведено в Додатку.

3.4 Тестування захисту від DDoS атак

Оцінка ефективності створеного додатка комплексного захисту від DDOS атак, за допомогою експерименту, метою якого буде порівняння працездатності веб-сервера, що знаходиться під DDoS атакою, з підключеним захистом і без. Для цього ми організуємо два стрес-тесту з однаковими параметрами атаки на заздалегідь підготовлений веб-сервер в VPC на захищений і незахищений IPадрес. Ступінь працездатності програми із захисту від DDoS атак на фільтрацію небажаного трафіку ми оцінимо за допомогою метрик завантаження процесора і мережевого пристрою.

Для тестування використаний наступний веб-сервер:

• Простий сайт на WordPress, розгорнутий за допомогою стандартного LAMPстека;

• Віртуальний сервер в VPC з 1 vCPU і 1.7 GB оперативної пам'яті на Debian. Були використані наступні інструменти моніторингу:

• Утиліта Netdata для перегляда відомостей про систему в реальному часі; Та наступний інструмент для проведення стрес-тесту:

• IP Stresser, що надає можливість створення тесту з об'ємом атаки до 3 Гбіт/с. Як середовище для розміщення веб-сервера було обрано віртуальне приватне хмара Google Cloud за можливість швидкого створення віртуальної машини і підключення публічної підмережі. Далі створимо сервер всередині проекту, використовуючи в якості джерела для встановлення системи готовий образ Debian 9. Для встановлення WordPress було обрано готовий пакетний стек Bitnami WordPress для хмарної платформи Google Cloud, через можливість швидкого та простого встановлення. Встановити можна через інтерфейс Google Cloud. На рисунку 3.1 зображенні основні характеристики створеної віртуальної машини, та мережевого інтерфейсу. Після встановлення wordpress перевіряємо працездатність веб-сайту. Як бачимо сайт повністю функціонує. Проаналізувавши стандартні правила міжмережевого екрана, зроблено висновок, що система не зможе протистояти DDoS атаці, оскільки відсутні правила для фільтрації потенційно злочинного трафіка).

Перейдемо до стрес-тесту системи на незахищений IP-адрес. Конфігурації атаки показані на рисунку 3.1. В якості адреси цільового хоста був призначений адрес новоствореного сайту, який при реальному використанні, без додаткових налаштувань та механізмів захисту, буде обслуговувати весь трафік, що йде на нього, він не буде фільтруватися і призведе до відмови працездатності системи.

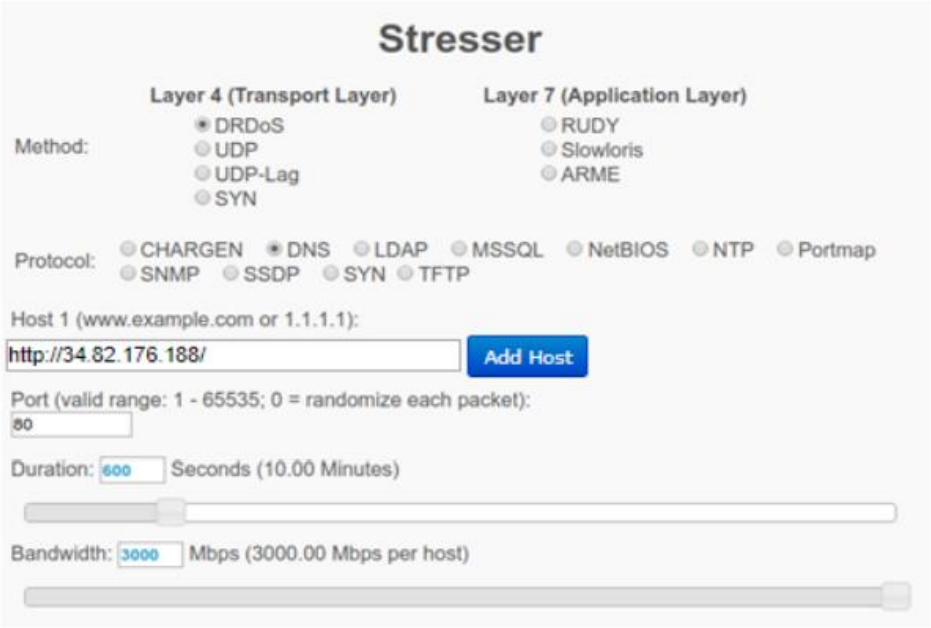


Рис. 3.1 – Конфігурації стрес-тесту

На рисунку 3.2 видно, що майже миттєво відбувається перевантаження процесора і об'єм прийнятого трафіку досягає до 3Гбіт/с.

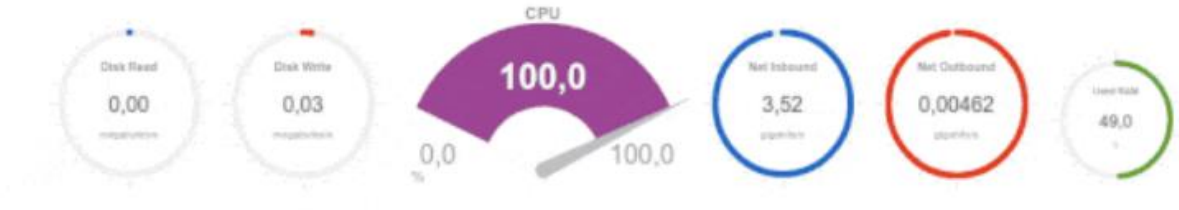


Рис. 3.2 - Метрика веб-сайту без захисту

Резюмуючи отримані результати, очевидно, що веб-сервер повністю прийняв на себе весь обсяг тестової атаки, а в разі збільшення навантаження стався б відмова працездатності програми. Далі ми протестуємо доступність веб-сервера з підключеним захистом від DDoS атак. Для цього підключимося через SSH до нашого сервера та включимо розроблений захист, використовуючи

команду ddosprotect.php –a (рис. 3.3). В результаті ми бачимо список новий список правил iptables (рис. 3.4).

```
progres_drumer@wordpress-1-vm:~$ ^C
progres_drumer@wordpress-1-vm:~$ iptables -L -n -v
-bash: iptables: command not found
progres_drumer@wordpress-1-vm:~$ sudo /sbin/iptables -L -n -v
Chain INPUT (policy ACCEPT 755K packets, 502M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 758K packets, 375M bytes)
 pkts bytes target     prot opt in     out     source               destination
progres_drumer@wordpress-1-vm:~$ ./ddosprotect.php -a
```

Рис. 3.3 - Список правил iptables до включення захисту та застосування захисту

```
:PREROUTING ACCEPT [161:12980]
:INPUT ACCEPT [161:12980]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [79:11948]
:POSTROUTING ACCEPT [79:11948]
-A PREROUTING -m conntrack --ctstate INVALID -j DROP
-A PREROUTING -p tcp -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -m conntrack --ctstate NEW -j DROP
-A PREROUTING -p tcp -m conntrack --ctstate NEW -m tcpmss ! --mss 536:65535 -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,RST FIN,RST -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,ACK FIN -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags ACK,URG URG -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,ACK FIN -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags PSH,ACK PSH -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,PSH,ACK,URG -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,PSH,URG -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,PSH,URG -j DROP
-A PREROUTING -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,ACK,URG -j DROP
-A PREROUTING -s 224.0.0.0/3 -j DROP
-A PREROUTING -s 169.254.0.0/16 -j DROP
-A PREROUTING -s 172.16.0.0/12 -j DROP
-A PREROUTING -s 192.0.2.0/24 -j DROP
-A PREROUTING -s 192.168.0.0/16 -j DROP
-A PREROUTING -s 10.0.0.0/8 -j DROP
-A PREROUTING -s 0.0.0.0/8 -j DROP
-A PREROUTING -s 240.0.0.0/5 -j DROP
-A PREROUTING -s 127.0.0.0/8 ! -i lo -j DROP
-A PREROUTING -p icmp -j DROP
-A PREROUTING -f -j DROP
COMMIT
```

Рис. 3.4 - Результат роботи команди ./ddos_protect.php –a

Стрес-тест на захищену IP-адресу, використовуючи повністю аналогічні параметри для атаки. Видно, що зареєстрований обсяг вхідного трафіку становить близько 120 Мбіт/с, а навантаження на процесор становить близько 20%.



Рис. 3.5 – Метрика веб-сайту з використанням захисту

Таким чином можна сказати, що використання подібного технічнопрограмного рішення із захисту від DDoS атак забезпечує певний рівень безпеки веб-сервера.

ВИСНОВОК

У роботі розроблена комплексна система захисту інформації веб-ресурсу.

Під час виконання роботи були проаналізовані основні види і типи загроз з точки зору інформаційної безпеки. Зазначено небезпеки і вразливості для розповсюджених веб-ресурсів.

Проведено огляд сучасних тенденцій в розвитку DDoS атак і атак іншого типу, та методів боротьби з ними;

Виявлено, що методи зловмисників постійно розвиваються, а їх послуги стають більш популярними.

Аналіз способів захисту показав, що тільки комплексне рішення може захистити та мінімізувати наслідки від DDoS атаки, а недооцінка важливості захисту в перспективі може нанести критичні збитки фізичним особам і підприємствам;

У роботі була реалізована система комплексного захисту від DDoS атак з використанням міжмережевого екрана для фільтрації небажаного трафіка, та додаткових методів захисту від DDoS атак.

Були сформовані ефективні конфігурації сервера та додаткові правила використання які можуть значно підвищити ефективність протидії системи DDoS атакам.

Розроблена система реалізована за допомогою мови програмування PHP та утиліт Iptables і Tcpdump, які вже превстановленні майже на всіх серверах на базі Linux. Працездатність перевірялась шляхом реалізації тестової атаки на веб-сайт. Далі планується з метою підвищення ефективності розробленої системи розширити методи розпізнавання шкідливого трафіка та поліпшити методи протидії DDoS атакам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DDoS Attacks Explained: Causes, Effects, and How to Protect Your Site/ Rachel McCollin - 2020. — https://kinsta.com/blog/what-is-a-ddos-attack/

2. DDoS attacks in Q3 2019/ Oleg Kupreev, Ekaterina Badovskaya, Alexander Gutnikov - November 11, 2019 — https://securelist.com/ddos-report-q3-2019/94958/

3. PHP Manual / Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, Jakub Vrana - 2020-05-26 — https://www.php.net/manual/en/

4. Habr. Доступные методы борьбы с DDoS-атаками для владельцев vds/ dedicated серверов с Linux / NetAngels - 2012. — https://habr.com/company/netangels/blog/149302/

5. Iptables Tutorial 1.1.19 / Oskar Andreasson – 2020. — https://www.frozentux.net/documents/iptables-tutorial/

6. Uwagbole S. O., Buchanan W. J., Fan L. (2017) Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention. In IFIP/IEEE Symposium on Integrated Network and Service Management 2017 (IM),

7. Кібербезпека : сучасні технології захисту. Навчальний посібник для студентів вищих навчальних закладів. / С. Е. Остапов, С. П. Євсеєв, О.Г. Король. – Львів: «Новий Світ2000», 2020 . – 678 с

8. Шило С.Г. Інформаційні системи та технології : навч. посіб. / С.Г. Шило, Г.В. Щербак, К.В. Огурцова. – Х. : ХНЕУ, 2013. – 219 с.

9. Potvin J. Genetic algorithms for the traveling salesman problem. Annals of Operations Research, 1996, 63, 339-370.

10. K. Vijayalakshmi and A. A. Leema "“Extenuating Web Vulnerability with a Detection and Protection Mechanism for a Secure Web Access” In Proc" 4th

International Conference on Signal Processing Communication and Networking (ICSCN) 2017.

11. G. Saride J. B. Aaron and J. Bose "“Secure Web Push System” In Proc" 8th International Conference on Communication Systems and Networks (COMSNETS) pp. 1-2 2016.

12. M. Casado P. Cao A. Akella and N. Provos "“Flow-Cookies: Using Bandwidth Amplification to Defend Against DDoS Flooding Attacks” In Proc" 14th IEEE International Workshop on Quality of Service pp. 286-287 2006.

13. J. Karro and J. Wang "“Protecting Web Servers from Security Holes in Server-Side Includes” In Proc" 14th Annual Computer Security Applications Conference (Cat. No.98EX217) 1998.

14. M. Huang "A Protection System for Web Server Resources" Proc. 10th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP) pp. 131-134 2013.

15. M. S. A. Eid and H. Aida "“Securely Hiding the Real Servers from DDoS Floods” In Proc" 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT’10) pp. 165-168 2010.

16. P. Pandiaraja and J. Manikandan "“Web Proxy based Detection and Protection Mechanisms against Client Based HTTP Attacks” In Proc" International Conference on Circuits Power and Computing Technologies (ICCPCT) 2015.

17. Y. Yuan and H. K. Ng "DATCONS: Protecting Web-Based QoS from DDoS Attacks" Proc. IEEE/IFIP Network Operations and Management Symposium NOMS 2006 2006.

18. M. H. Kamarudin C. Maple T. Watson and N. S. Safa "A LogitBoost-Based Algorithm for Detecting Known and Unknown Web Attacks" IEEE Access vol. 5 pp. 26190-26200 2017.

19. R. Williams E. McMahon S. Samtani M. Patton and H. Chen "Identifying Vulnerabilities of Consumer Internet of Things (IoT) Devices: A Scalable Approach"

Proc. IEEE International Conference on Intelligence and Security Informatics (ISI) pp. 179-181 2017.

20. Y. Seralathan T. T. Oh S. Jadhav J. Myers J. P. Jeong Y. H. Kim et al. "IoT Security Vulnerability: A Case Study of a Web Camera" Proc. The 20th International Conference on Advanced Communication Technology (ICACT) pp. 172-177 2018.

21. S. Sivabalan and P. J. Radcliffe "Detecting IoT Zombie Attacks on Web Servers" Proc. The 27th International Telecommunication Networks and Applications Conference (ITNAC) 2017.

22. S. N. Han G. M. Lee N. Crespi N. V. Luong K. Heo M. Brut et al. "“DPWSim: A Devices Profile for Web Services (DPWS) Simulator” In Proc" IEEE Internet of Things Journal vol. 2 no. 3 pp. 221-229 2015.

23. A. K. Das M. Wazid A. R. Yannam J. J. P. C. Rodrigues and Y. Park "Provably Secure ECC-Based Device Access Control and Key Agreement Protocol for IoT Environment" IEEE Access vol. 7 pp. 55382-55397 2019.

24. D. Hwang J. Choi and K. Kim "“Dynamic Access Control Scheme for IoT Devices using Blockchain” In Proc" International Conference on Information and Communication Technology Convergence (ICTC) pp. 713-715 2018.

25. P. Wang Y. Yue W. Sun and J. Liu "“An Attribute-Based Distributed Access Control for Blockchain-enabled IoT” In Proc" 2019 International Conference on Wireless and Mobile Computing Networking and Communications (WiMob) pp. 1-6 2019.

26. S. Ding J. Cao C. Li K. Fan and H. Li "A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT" IEEE Access vol. 7 pp. 38431-38441 2019.

27. S. Chung J. H. Kim and Y. Kim "Pragmatic Approach Using OAuth Mechanism for IoT Device Authorization in Cloud" Proc. International Conference on Advances in Computing Communication Control and Networking (ICACCCN) pp. 1-4 2018.

28. A. Adireddy U. Gottapu and A. P. Aravamudhan "Usercentric Federation of Access to Internet-of-Things(IoT) Devices: A valet key for IoT devices" Proc. 2016 International Conference on Circuits Controls Communications and Computing (I4C) 2016.