Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідуюча кафедри кібербезпеки
та захисту інформації
_____ Наталія. ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього ступеня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Засоби захисту операційних платформ з відкритим кодом від кібернетичних атак»

Виконавець: студент IV курсу, групи КБ-41

_____ **Микола ГРАСС**
(підпис) (ім'я прізвище)

	Прізвище, ініціали	**Підпис**
Керівник	Іван ПАРХОМЕНКО	

Нормоконтроль	Сергій ДАКОВ	

Київ 2022

Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідуюча кафедри кібербезпеки
та захисту інформації
_____ Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-41** (група) **Грассу Миколі Вікторовичу** (прізвище ім'я по-батькові)

Тема дипломної роботи «Засоби захисту операційних платформ з відкритим кодом від кібернетичних атак»

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №2 від 29.10.2021 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Структура операційних систем з відкритим кодом, архітектура операційних систем

Linux, стек технологій бля розробки сканера

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Структура операційних систем з відкритим кодом, засоби захисту ОС від шкідливого ПЗ, Програмна реалізація механізму захисту ОС від шкідливого ПЗ

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Автоматизація аналізу програмного забезпечення на шкідливий код в операційних системах сімейства Unix

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 01 листопада 2021 року

Завдання видав _____ (підпис) Іван ПАРХОМЕНКО (ініціали, прізвище)

Завдання прийняв до виконання _____ (підпис) Микола ГРАСС (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	01.11.2021 – 26.01.2022	*виконано*
2	Аналіз літератури	27.01.2022 – 20.02.2022	*виконано*
3	Розгляд структури ОС з відкритим кодом	21.02.2022 – 03.03.2022	*виконано*
4	Аналіз загроз від шкідливого ПЗ в ОС з відкритим кодом	04.03.2022 – 30.03.2022	*виконано*
5	Розгляд інструментів сканування файлів на шкідливе ПЗ	31.03.2022 – 17.04.2022	*виконано*
6	Вибір методу сканування	18.04.2022 – 07.05.2022	*виконано*
7	Програмна реалізація сканера	08.05.2022 – 26.05.2022	*виконано*
8	Оформлення пояснювальної записки	27.05.2022 – 08.06.2022	*виконано*
9	Підготовка до захисту	09.06.2022 – 14.06.2022	*виконано*

Завдання видав _____ (підпис) Іван ПАРХОМЕНКО (ініціали, прізвище)

Завдання прийняв до виконання _____ (підпис) Микола ГРАСС (ініціали, прізвище)

Термін подання дипломної роботи до ЕК 06 червня 2022 року

УДК 004.056.5

РЕФЕРАТ

Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків. Основний текст займає 59 сторінок, включає в себе зміст, вступ, три розділи дипломної роботи, висновки та список джерел. Крім того, робота містить додаток із загальною кількістю сторінок 5. У пояснювальній записці дипломної роботи міститься 5 рисунків.

Метою роботи є реалізація засобів та механізмів захисту операційних платформ з відкритим кодом від кібернетичних атак

Об'єктом дослідження є процес захисту операційних платформ з відкритим кодом

Предметом дослідження є засоби та механізми, які реалізують засоби та методи захисту операційних платформ з відкритим кодом

Методи дослідження дипломної роботи:

- аналіз літератури;
- аналіз документів;
- системний підхід;
- методи порівняння;
- структурний аналіз

Для досягнення зазначеної мети поставлено наступні завдання:

- дослідити структуру операційних систем з відкритим кодом
- проаналізувати засоби та механізми виявлення шкідливого програмного забезпечення
- розробити програмну реалізацію для виявлення шкідливого програмного забезпечення

Практичною цінністю отриманих результатів є програмна реалізація засобів захисту операційних платформ з відкритим кодом

Ключові слова: кібератака, захист інформації, виявлення шкідливого коду, сигнатурний аналіз, евристичний аналіз, сімейство Unix-подібних операційних систем

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПЗ	–	Програмне забезпечення
ОС	–	Оперційна Система
API	–	Aplication Programing Interface
OS	–	Operating System
FTP	–	File Tranfer Protocol
DOS	–	Denial of Service
SSH	–	Secure Shell
SCP	–	Secure Copy Protoco.l
RDP	–	Remote Desktop Protocol
REST	–	Representational State Transfer
HTTP	–	Hypertext Transfer Protocol
EDR	–	Endpoint Detection and Respose
AV	–	Antivirus
IP	–	Internet Protocol
IDS	–	Intrusion Detection System
HIDS	–	Host-based intrusion detection system
HIPS	–	Host-based Intrusion Prevention System

ЗМІСТ

ВСТУП

Актуальність: Сьогодні поширені операційні системи з відкритим вихідним кодом. Це пояснюється тим, що вони безкоштовні і на думку більшості, безпечні. Яскравим прикладом операційних систем з відкритим кодом є Unix-подібні системи. Великою проблемою нових ОС є те, що розробники копіюють базовий дистрибутив доповнюючи новими функціями, тому проблеми з безпекою старих ОС перекочуються в нові, що може призводити до успішних кібератак. Тому захист операційних систем з відкритим кодом є актуальним на сьогоднішній час.

Тому не малі випадки атаки на ресурси з цими операційними системами ,щодня більша частина організацій потерпає від кібератак, і тому наразі є дуже актуальне питання захисту операційних систем з відкритим кодом.

Метою роботи є реалізація засобів та механізмів захисту операційних платформ з відкритим кодом від кібернетичних атак

Для досягнення зазначеної мети поставлено наступні завдання:

- дослідити структуру операційних систем з відкритим кодом
- проаналізувати засоби та механізми виявлення шкідливого програмного забезпечення
- розробити програмну реалізацію для виявлення шкідливого програмного забезпечення

Методи дослідження дипломної роботи:

- аналіз літератури;
- аналіз документів;
- системний підхід;
- методи порівняння;
- структурний аналіз

Об'єктом дослідження є процес захисту операційних платформ з відкритим кодом

Предметом дослідження є засоби та механізми, які реалізують засоби та методи захисту операційних платформ з відкритим кодом

Практичною цінністю отриманих результатів є програмна реалізація засобів захисту операційних платформ з відкритим кодом

Ключові слова: кібератака, захист інформації, виявлення шкідливого коду, сигнатурний аналіз, евристичний аналіз, сімейство Unix-подібних операційних систем

РОЗДІЛ 1
АНАЛІЗ СТРУКТУРИ ОПЕРАЦІЙНИХ СИСТЕМ З ВІДКРИТИМ КОДОМ ТА МОЖЛИВИХ ЗАГРОЗ ДЛЯ НИХ

1.1 Основи операційних систем з відкритим кодом

Програмне забезпечення з відкритим кодом – це комп'ютерне програмне забезпечення або програми, де власники або власники авторських прав дозволяють користувачам або третім особам бачити, використовувати та надавати право змінювати вихідний код продукту.

Операційна система з відкритим вихідним кодом — це операційна система, в якій вихідний код є загальнодоступним і доступним для редагування. Загальновідомі операційні системи, такі як Microsoft Windows, Apple iOS і Mac OS, є закритою операційною системою.

Закриті операційні системи побудовані з численними кодами та складним програмуванням, і це називається вихідним кодом. Цей вихідний код зберігається в таємниці відповідними компаніями (власниками) і недоступний для третіх осіб.

Операційна система з відкритим кодом працює так само, як і закриті, єдина відмінність полягає в тому, що вихідний код або весь додаток може змінюватися користувачем. Немає різниці в продуктивності, але може бути відмінність у функціонуванні.

Наприклад, у пропріетарній (закритій) операційній системі інформація упаковується та зберігається. Те ж саме відбувається у відкритому коді. Але оскільки вихідний код є видимим для вас (користувача), ви можете зрозуміти процес і змінити спосіб обробки інформації.

Хоча перший є безпечним і простим, другий потребує певних технічних знань, але ви можете налаштувати та підвищити продуктивність. Різниця та плюси та мінуси обговорюються далі в статті.

Немає визначеного способу чи рамки для роботи операційної системи з відкритим вихідним кодом; його можна налаштувати відповідно до потреб користувача.

Більшість операційних систем з відкритим кодом базуються на Linux.

Ядро Linux створено Лінусом Торвальдсом. Він забезпечує основні функції, необхідні для операційної системи, такі як розподілення даних, обробка пам'яті та взаємодія з апаратним забезпеченням комп'ютера. Linux є відкритим вихідним кодом, багато розробників вивчали вихідний код і створили багато допоміжних плагінів і дистрибутивів для своїх потреб:

Деякі дистрибутиви зосереджуються на простоті та зручності використання, наприклад: Ubuntu, Linux Mint та Elementary OS;

Інші використовуються в сфері інформаційної безпеки або забезпенні анонімності, такі як Kali, Parrot, Tails.

Використання ОС з відкритим кодом мають як переваги та недоліки.

До переваг можна зазначити:

- Економічність – більшість ОС з відкритим кодом є безкоштовною. І деякі з них доступні за дуже дешевою ціною, ніж комерційні закриті продукти.
- Надійність та ефективність – більшість контролюються тисячами очей, оскільки вихідний код є відкритим. Тож якщо є якісь уразливості чи помилки, їх виправляють найкращі розробники у всьому світі
- Гнучкість - велика перевага в тому, що ви можете налаштувати його відповідно до ваших потреб. І є свобода творчості.

Щодо недоліків, можна вказати:

- Ризик безпеки – хоча помилки виявлені, існує ризик атак, оскільки вихідний код доступний зловмисникам.
- Складність – він не зручний у використанні, як закриті. Щоб користуватися цим програмним забезпеченням, потрібно мати мінімальні технічні знання
- Підтримка – якщо ви зіткнулися з проблемою, тоді не буде служби підтримки клієнтів, яка б могла вам допомогти.

Хоча Linux лежить в основі більшості операційних систем з відкритим кодом, існують ОС, побудовані на основі інших ядер.

Однією з примітних альтернатив є FreeBSD , безкоштовна операційна система, походження якої сягає операційної системи Berkeley Unix 1970-х років, але вона не побудована на ядрі Linux. Хоча Unix-подібна система не заснована на Linux, її можна використовувати з робочими столами Gnome 2 і KDE, макет яких буде знайомий давнім користувачам Linux, а також запускати багато програм, сумісних з Linux.

Можливо, найдивовижнішою альтернативою є ReactOS, незавершена ОС з відкритим кодом, схожа на Windows XP і призначена для заміни Windows, яка веде себе так само, як і старий улюблений. Будь-хто, хто думає спробувати ReactOS, повинен знати, що це альфа-програмне забезпечення, тому очікуйте багато помилок і відсутні функціональні можливості.

Для будь-яких користувачів комп'ютерів старої школи, які прагнуть командного рядка DOS, є також FreeDOS, операційна система з відкритим вихідним кодом, DOS-сумісна, яку можна використовувати для запуску застарілого програмного забезпечення – його виробники стверджують, що будь-яке програмне забезпечення, яке працює на MS- DOS також має працювати на FreeDOS.

Визначити, які з відкритих операційних систем є найпопулярнішими, непросто. На перший погляд, найпопулярнішою операційною системою з відкритим вихідним кодом є Android, яка заснована на модифікованій версії ядра Linux. Проблема в тому, що те, що багато людей вважають Android, не є повністю відкритим вихідним кодом. В основі ОС Android лежить Android Open Source Project, який відкритий, як випливає з назви, але побудований на основі програм із закритим кодом від Google.

Існує також Chromium OS, операційна система з відкритим кодом, розроблена для запуску веб-програм. Хоча сама по собі Chromium OS не широко поширена, вона є основою ОС Chrome, яка працює на комп'ютерах Chromebook, які особливо популярні на ринку освіти.

Крім того, операційні системи на базі Linux досі займають лише від одного до трьох відсотків ринку настільних ПК. Серед них є поєднання старих фаворитів, таких як Ubuntu і Debian, і новіших конкурентів, таких як Linux Mint і Elementary OS. Теорія стверджує, що, якщо багато людей уважно вивчають код, хтось, швидше за все, помітить помилки, діри в безпеці та шкідливі підпрограми в ОС.

Однак це не завжди працює на практиці, оскільки принаймні один гучний приклад серйозної помилки безпеки залишається в операційних системах з відкритим кодом роками, перш ніж бути помічений. Деякі видатні розробники стверджують, що сучасне програмне забезпечення настільки складне, що цілеспрямований експертний огляд кількох людей важливіший , ніж побіжний огляд багатьох.

Як згадувалось раніше існує багато операційних систем з відкритим кодом, орієнтованих на забезпечення високобезпечних машин, які також захищають конфіденційність користувача.

Tails — це ОС, орієнтована на конфіденційність, розроблена для тих, хто хоче максимально уникати відстеження. Tails можна запускати з DVD- або USB-накопичувача, і його розробники кажуть , що він «не залишить слідів на комп’ютері, який ви використовуєте», намагатиметься анонімізувати використання Інтернету через мережу Tor і шифруватиме ваші файли, електронні листи та миттєві повідомлення.

Іншим прикладом ОС з відкритим вихідним кодом, орієнтованої на безпеку, є Qubes , операційна система, яка підвищує безпеку, розділяючи різні види діяльності на ізольовані екземпляри, які називаються qubes, які не можуть впливати один на одного. Наприклад, у вас може бути один qube для відвідування ненадійних веб-сайтів, а інший — для здійснення онлайн-банкінгу.

1.2 Архітектура операційних систем на базі Linux

Як зазначалось в попередньому підрозділі - Linux є найвідомішою та найбільш використовуваною операційною системою з відкритим кодом. Як операційна

система, Linux — це програмне забезпечення, яке знаходиться під усім іншим програмним забезпеченням на комп’ютері, отримує запити від цих програм і передає ці запити на обладнання комп’ютера.

Від смартфонів до автомобілів, суперкомп’ютерів і побутової техніки, домашніх настільних комп’ютерів до корпоративних серверів — операційна система Linux є всюди.

Linux існує з середини 1990-х років і з тих пір досяг користувацької бази, яка охоплює весь світ. Linux насправді скрізь: він у ваших телефонах, термостатах, у ваших автомобілях, холодильниках, пристроях Roku і телевізорах. Він також керує більшою частиною Інтернету, усіма 500 найкращими суперкомп’ютерами світу та світовими фондовими біржами.

Але крім того, що Linux є найкращою платформою для запуску настільних комп’ютерів, серверів і вбудованих систем по всьому світу, Linux є однією з найнадійніших, безпечних і безтурботних операційних систем.

Як і Windows, iOS і Mac OS, Linux є операційною системою. Фактично, одна з найпопулярніших платформ на планеті, Android, працює на базі операційної системи Linux.

Операційна система — це програмне забезпечення, яке керує всіма апаратними ресурсами, пов’язаними з вашим настільним комп’ютером або ноутбуком. Простіше кажучи, операційна система керує зв’язком між вашим програмним забезпеченням і апаратним забезпеченням. Без операційної системи (ОС) програмне забезпечення не функціонує.

Операційна система Linux складається з кількох різних частин:

● Bootloader – програмне забезпечення, яке керує процесом завантаження вашого комп’ютера. Для більшості користувачів це буде просто заставка, яка спливає і в кінцевому підсумку зникає для завантаження операційної системи.

● Ядро – це єдина частина цілого, яка насправді називається «Linux». Ядро є ядром системи і керує процесором, пам'яттю та периферійними пристроями. Ядро є найнижчим рівнем ОС.

● Система ініціалізації – це підсистема, яка завантажує користувальницький простір і відповідає за керування демонами. Однією з найбільш широко використовуваних систем ініціалізації є systemd, яка також є однією з найбільш суперечливих. Це система ініціалізації, яка керує процесом завантаження після того, як початкове завантаження передається із завантажувача (тобто GRUB або GRand Unified Bootloader).

● Демони – це фонові служби (друк, звук, планування тощо), які запускаються під час завантаження або після входу на робочий стіл.

● Графічний сервер – це підсистема, яка відображає графіку на вашому моніторі. Його зазвичай називають X-сервером або просто X.

● Середовище робочого столу – це частина, з якою користувачі насправді взаємодіють. Є багато середовищ робочого столу на вибір (GNOME, Cinnamon, Mate, Pantheon, Enlightenment, KDE, Xfce тощо). Кожне середовище робочого столу містить вбудовані програми (наприклад, файлові менеджери, інструменти конфігурації, веб-браузери та ігри).

● Програми . Настільні середовища не пропонують повний набір програм. Так само, як Windows і macOS, Linux пропонує тисячі і тисячі високоякісних програм, які можна легко знайти та встановити. Більшість сучасних дистрибутивів Linux (докладніше про це нижче) включають інструменти, подібні до App Store, які централізують та спрощують встановлення програм. Наприклад, Ubuntu Linux має Центр програмного забезпечення Ubuntu (ребрендинг програмного забезпечення GNOME), який дозволяє швидко шукати серед тисяч програм і встановлювати їх з одного централізованого місця.

● Ядро - це невеликий спеціальний код, який є основним компонентом ОС Linux і безпосередньо взаємодіє з обладнанням. Це проміжний рівень між програмним та апаратним забезпеченням, який забезпечує низький рівень обслуговування компонентів режиму користувача. Він повністю розроблений на мові C та архітектурі файлової системи. Крім того, він має різні блоки, які керують різними операціями. .У цьому підручнику ми дізнаємося про архітектуру ядра Linux.

Ядро одночасно запускає ряд процесів і керує різними ресурсами. Він розглядається як менеджер ресурсів, коли в системі одночасно виконується кілька програм. У цьому випадку ядро є екземпляром, який спільно використовує доступні ресурси, такі як час процесора, дисковий простір, мережеві підключення тощо.

Архітектура ядра поділяється на 2 типи: Монолітне ядро та Модульні ядра

У традиційній архітектурі монолітного ядра всі основні системні послуги, такі як управління процесами та пам’яттю, обробка переривань тощо, були упаковані в один модуль у просторі ядра. Модуль - це об'єктний файл, у якому весь код може бути пов'язаний з ядром під час виконання. Традиційне монолітне ядро займає величезний простір, а рівень обслуговування дуже низький. Повторна компіляція займає кілька годин, якщо додається якась нова функція, оскільки всі служби приєднані до одного модуля.

Сучасна монолітна архітектура ядра складається з різних модулів, які можна динамічно завантажувати і вивантажувати. Таким чином, ремонтопридатність дуже проста, оскільки ядру потрібно подбати лише про завантажений модуль. Повторна компіляція не потрібна після додавання функції або деяких змін. Монолітне ядро швидше, ніж Модульні ядра.

Модульна архітектура ядра підтримує модульний підхід. Усі службові модулі не запускаються в просторі ядра в порівнянні з монолітним ядром. Керування драйверами пристрою, стеком протоколів, файловою системою тощо запускаються в просторі користувача. Це зменшує розмір коду ядра, а також підвищує безпеку.

Архітектура ядра розділена на дві основні частини:

Простір користувача. Усі програми та програми користувача виконуються в просторі користувача. Простір користувача не може отримати прямий доступ до пам’яті та обладнання. Він отримує доступ до обладнання через простір ядра. Процеси або програми, які виконуються в просторі користувача, отримують доступ лише до певної частини пам'яті за допомогою системного виклику. Завдяки повному захисту, збої в режимі користувача можна відновити.

Простір ядра. Усі програми ядра виконуються в просторі ядра. Простір ядра отримує доступ до всієї частини пам’яті та безпосередньо взаємодіє з обладнанням,

таким як оперативна пам’ять, жорсткий диск тощо. Він розділений на різні блоки та модулі, які керують усіма операціями (наприклад, керування файлами, управління пам’яттю, керування процесами тощо) у просторі ядра та додатками, що працюють у простір користувача. Простір ядра складається з інтерфейсу системного виклику, ядра (основний компонент Linux) і модуля пристрою.

Інтерфейс системного виклику є проміжним рівнем між простором користувача та простором ядра. Кожна програма, яка запускається в просторі користувача, може взаємодіяти з ядром через інтерфейс системного виклику. Наприклад, функція системного виклику для роботи з файлом: відкриття ( ), запис ( ), читання ( ) тощо.

Ядро не залежить від апаратного забезпечення. Це загальне для всіх апаратних процесорів, які підтримуються Linux. Ви можете запускати ядро на будь-якому процесорі, як-от Intel, ARM, Atmel тощо. Воно діє як менеджер ресурсів у просторі ядра та виконує керування процесами, файлами, пам’яттю, обробником переривань, плануванням процесу тощо. Це потужна структура, яка виконує всі види операцій.

Архітектура ядра дотримується модульного підходу. Кожен блок у ядрі (тобто управління файлами) є не що інше, як фрагмент коду, написаний мовою C. Кожен блок складається з потужної структури для обробки різноманітних операцій.

Бібліотека GNU C забезпечує механізм перемикання програми простору користувача на простір ядра.

Управління процесами — це керування різними процесами, які виконуються одночасно. Процес — це екземпляр «виконуваної програми», як-от відкриття файлу, доступ до диска, доступ до зовнішнього ресурсу (наприклад, принтер) тощо, і може бути створений та знищений. Керування процесом надає інформацію про те, що відбувається з процесом, і керує його пріоритетами, наприклад, яка адреса повинна бути призначена процесу, файл, виділений процесу, стан процесу (наприклад, запуск, очікування, зупинка) тощо. Я коротко поясню процес в іншому підручнику.

Управління пам'яттю є найважливішою частиною ядра, яка обробляє призначення адресного простору для процесу та програми. В основному управління пам’яттю призначає віртуальну пам’ять замість фізичної пам’яті, при цьому остання

є фактичним адресним простором в RAM. Не плутайте фізичну та віртуальну пам’ять. Призначення віртуальної адреси долає обмеження щодо призначення фізичної пам'яті. Перетворення фізичної адреси у віртуальну адресу виконується за допомогою MMU (модулю керування пам’яттю, який забезпечує захист від перешкод, спільне використання пам’яті та виділення віртуальної пам’яті. Фізичний адресний простір ділиться на деякий блок пам’яті, який називається фреймом, який містить кілька сторінок.

Файлова система Linux, як правило, є вбудованим рівнем операційної системи Linux, що використовується для керування даними сховища. Це допомагає впорядкувати файл на дисковому сховищі. Він керує назвою файлу, розміром файлу, датою створення та багато іншого про файл.

Якщо у нашій файловій системі є непідтримуваний формат файлу, ми можемо завантажити програмне забезпечення для роботи з ним.

Файлова система Linux має ієрархічну файлову структуру, оскільки містить кореневий каталог та його підкаталоги. До всіх інших каталогів можна отримати доступ з кореневого каталогу. Розділ зазвичай має лише одну файлову систему, але він може мати більше однієї файлової системи.

Файлова система розроблена таким чином, щоб вона могла керувати та забезпечувати простір для енергонезалежних даних. Усі файлові системи вимагали простору імен, який є методологією імен і організаційною методологією. Простір імен визначає процес іменування, довжину імені файлу або підмножину символів, які можна використовувати для імені файлу. Він також визначає логічну структуру файлів у сегменті пам'яті, наприклад використання каталогів для організації конкретних файлів. Після опису простору імен необхідно визначити опис метаданих для цього конкретного файлу.

Структура даних повинна підтримувати ієрархічну структуру каталогів. Ця структура використовується для опису доступного та використовуваного дискового простору для певного блоку. Він також містить інші відомості про файли, такі як розмір файлу, дата і час створення, оновлення та останньої зміни.

Крім того, він зберігає додаткову інформацію про розділ диска, таку як розділи та томи.

Розширені дані та структури, які вони представляють, містять інформацію про файлову систему, що зберігається на диску; він відрізняється і не залежить від метаданих файлової системи.

У Linux файлова система створює деревовидну структуру. Усі файли впорядковані у вигляді дерева та його гілок. Найвищий каталог називається кореневим (/) . До всіх інших каталогів у Linux можна отримати доступ з кореневого каталогу.

Ключові особливості файлової системи Linux:

- Зазначення шляхів: Linux не використовує зворотну косу риску (\) для розділення компонентів; він використовує пряму косу риску (/) як альтернативу. Наприклад, як і в Windows, дані можуть зберігатися в C:\ My Documents\ Work, тоді як у Linux вони будуть зберігатися в /home/ My Document/ Work.
- Розділи, каталоги та диски: Linux не використовує літери дисків для організації диска, як це робить Windows. У Linux ми не можемо визначити, чи ми звертаємося до розділу, мережевого пристрою чи «звичайного» каталогу та диска.
- Чутливість до регістру: файлова система Linux чутлива до регістру. Він розрізняє імена файлів у нижньому та верхньому регістрі. Наприклад, існує різниця між test.txt і Test.txt в Linux. Це правило також застосовується до каталогів і команд Linux.
- Розширення файлів: у Linux файл може мати розширення «.txt», але необов’язково, щоб файл мав розширення. Під час роботи з Shell це створює деякі проблеми для новачків, щоб розрізняти файли та каталоги. Якщо ми використовуємо графічний файловий менеджер, він символізує файли та папки.
- Приховані файли: Linux розрізняє стандартні файли та приховані файли, в основному файли конфігурації приховані в ОС Linux. Зазвичай нам не потрібно відкривати або читати приховані файли. Приховані файли в Linux представлені крапкою (.) перед іменем файлу (наприклад, .ignore). Щоб отримати доступ до

файлів, нам потрібно змінити вигляд у файловому менеджері або використовувати певну команду в оболонці.

Коли ми встановлюємо операційну систему Linux, Linux пропонує багато файлових систем, таких як Ext, Ext2, Ext3, Ext4, JFS, ReiserFS, XFS, btrfs і swap

Файлова система Ext означає розширена файлова система . В першу чергу він був розроблений для ОС MINIX . Файлова система Ext є старішою версією і більше не використовується через деякі обмеження.

Ext2 — перша файлова система Linux, яка дозволяє керувати двома терабайтами даних. Ext3 розробляється через Ext2; це оновлена версія Ext2 і містить зворотну сумісність. Основним недоліком Ext3 є те, що він не підтримує сервери, оскільки ця файлова система не підтримує відновлення файлів і знімок диска.

Ext4 є найшвидшою файловою системою серед усіх файлових систем Ext. Це дуже сумісний варіант для SSD (твердотільний накопичувач) дисків, і це файлова система за замовчуванням у дистрибутиві Linux.

JFS означає журнальну файлову систе**му** , і вона розроблена IBM для AIX Unix . Це альтернатива файловій системі Ext. Його також можна використовувати замість Ext4, де потрібна стабільність з невеликими ресурсами. Це зручна файлова система, коли ЦП обмежена.

ReiserFS є альтернативою файловій системі Ext3. Він має покращену продуктивність і розширені функції. Раніше ReiserFS використовувалася як файлова система за замовчуванням у SUSE Linux, але пізніше вона змінила деякі політики, тому SUSE повернувся до Ext3. Ця файлова система динамічно підтримує розширення файлу, але має деякі недоліки в продуктивності.

Файлова система XFS розглядалася як високошвидкісна JFS, яка розроблена для паралельної обробки вводу-виводу. NASA все ще використовує цю файлову систему зі своїм сервером високої пам’яті (сервер 300+ терабайт).

Btrfs означає файлову систему дерева B. Він використовується для відмовостійкості, системи ремонту, веселого адміністрування, великої конфігурації сховища тощо. Це не найкращий костюм для виробничої системи.

Файлова система підкачки(swap) використовується для підкачки пам'яті в операційній системі Linux під час сну. Система, яка ніколи не переходить у режим глибокого сну, повинна мати простір підкачки, рівний розміру її оперативної пам’яті.

1.3 Загрози для серверів на базі операційних систем Linux

Безпека сервера так само важлива, як і безпека мережі, оскільки сервери часто містять велику кількість важливої інформації організації. Якщо сервер зламано, весь його вміст може стати доступним для злому для крадіжки або маніпулювання за бажанням. Найбільш поширеними загрозами для серверів є:

Невикористані сервіси та відкриті порти – Повна інсталяція Red Hat Enterprise Linux містить до 1200 пакетів програм і бібліотек. Однак більшість адміністраторів серверів не вибирають встановлення кожного окремого пакета в дистрибутиві, а замість цього встановлюють базову інсталяцію пакетів, включаючи кілька серверних програм. Поширеним явищем серед системних адміністраторів є встановлення операційної системи, не звертаючи уваги на те, які програми насправді встановлюються. Це може бути проблематичним, оскільки можуть бути встановлені непотрібні служби, налаштовані за умовчанням і, можливо, увімкнені. Це може призвести до запуску небажаних служб, таких як Telnet, DHCP або DNS, на сервері або робочій станції без усвідомлення адміністратора, що, у свою чергу, може спричинити небажаний трафік на сервер або навіть потенційний шлях до системи для злому.

Невиправлені служби – більшість серверних програм, які включені в стандартну інсталяцію, є надійними, ретельно перевіреними частинами програмного забезпечення. Оскільки їх код використовується у виробничих середовищах протягом багатьох років, їх код був ретельно вдосконалений, а багато помилок було знайдено та виправлено. Однак ідеального програмного забезпечення не існує, і завжди є місце для подальшого вдосконалення. Більше того, новітнє програмне забезпечення часто не так ретельно перевіряється, як можна було б очікувати, через

його нещодавнє прибуття в виробниче середовище або через те, що воно може бути не таким популярним, як інше серверне програмне забезпечення. Розробники та системні адміністратори часто знаходять у серверних програмах помилки, які можна використовувати, і публікують інформацію на веб-сайтах, пов'язаних із відстеженням помилок та безпеці, як-от список розсилки Bugtraq або веб-сайт групи реагування на надзвичайні ситуації з комп'ютером (CERT).

Неуважне адміністрування – адміністратори, які не можуть виправити свої системи, є однією з найбільших загроз для безпеки сервера. За даними Інституту системного адміністрування мережі та безпеки (SANS), основна причина вразливості комп'ютерної безпеки полягає в тому, щоб призначати непідготовлених людей для забезпечення безпеки і не надають ні навчання, ні часу, щоб зробити можливим виконання роботи. Це стосується як недосвідчених адміністраторів, так і надмірно самовпевнених або вмотивованих адміністраторів.

Деякі адміністратори не можуть виправити свої сервери та робочі станції, в той час як інші не можуть переглядати повідомлення журналу від ядра системи або мережевого трафіку. Ще одна поширена помилка полягає в тому, щоб залишити без змін стандартні паролі або ключі до служб. Наприклад, деякі бази даних мають паролі адміністрування за замовчуванням, оскільки розробники баз даних припускають, що системний адміністратор змінює ці паролі відразу після встановлення. Якщо адміністратор бази даних не зможе змінити цей пароль, навіть недосвідчений зломщик може використовувати широко відомий пароль за замовчуванням, щоб отримати адміністративні привілеї бази даних. Це лише кілька прикладів того, як неуважне адміністрування може призвести до зламаних серверів.

Незахищені служби – навіть найпильніша організація може стати жертвою вразливостей, якщо мережеві послуги, які вони вибирають, є небезпечними. Наприклад, існує багато сервісів, розроблених з припущенням, що вони використовуються в надійних мережах; однак, це припущення зазнає невдачі, як тільки послуга стає доступною через Інтернет, яка сама по собі є ненадійною.

Однією з категорій незахищених мережевих служб є ті, які вимагають незашифрованих імен користувачів і паролів для аутентифікації. Telnet і FTP є

двома такими службами. Якщо програмне забезпечення для перехоплення пакетів контролює трафік між віддаленим користувачем, імена користувачів і паролі такої служби можуть бути легко перехоплені.

По суті, такі служби також можуть легше стати жертвою того, що галузь безпеки називає атакою «людина посередині». У цьому типі атаки зломщик перенаправляє мережевий трафік, обманюючи зламаний сервер імен у мережі, щоб він вказував на його комп'ютер, а не на призначений сервер. Як тільки хтось відкриває віддалений сеанс на сервері, машина зловмисника діє як невидимий канал, тихо сидить між віддаленим сервісом і нічого не підозрюючим користувачем, який захоплює інформацію. Таким чином, зломщик може збирати адміністративні паролі та необроблені дані без усвідомлення цього сервером або користувачем.

Інша категорія незахищених служб включає мережеві файлові системи та інформаційні послуги, такі як NFS або NIS, які розроблені спеціально для використання локальної мережі, але, на жаль, розширені, щоб включити глобальні мережі (для віддалених користувачів). NFS за замовчуванням не має жодних механізмів автентифікації або безпеки, налаштованих для запобігання зломщику монтувати загальний ресурс NFS і отримати доступ до всього, що в ньому міститься. NIS також має життєво важливу інформацію, яку повинен знати кожен комп'ютер у мережі, включаючи паролі та дозволи на файли, у базі даних ACSII або DBM (на основі ASCII). Зломщик, який отримує доступ до цієї бази даних, може отримати доступ до кожного облікового запису користувача в мережі, включаючи обліковий запис адміністратора.

Ще одною на перший погляд незначною загрозою для серверів на базі ОС Linux є шкідливе ПЗ, на відмінну від Windows. Так як Linux є відкритою операційною системою, і щоб ПЗ отримало доступ до ядра чи інших важливих файлів потрібен доступ до root, тому написання вірусі для Linux є досить складною справою, тобто не дуже поширеною. Але випадки які все ставались наносять колосальні збитки, через поширеність використання Linux в великих компаніях критичних інфраструктурах, в хмарі. Такі атаки є цілеспрямовані, тобто більш небезпечні. Тому захист від цього типу атак є досить актуальним в наш час.

За даними ФБР , у 2020 році кібератаки завдали збитків на суму майже 4,2 мільярда доларів. Ця цифра на 20% більше, ніж у попередньому році. У останніх звітах Verizon та Європол також стверджують, що цифрові атаки процвітають.

Найпоширенішою загрозою у цих звітах є вид шкідливого ПЗ – програми-вимагачі. Будучи однією з найнебезпечніших цифрових загроз , вимагачі – це шкідливі програми , які шифрують файли та системи. Весь доступ заблоковано, поки користувач не заплатить викуп. Це складна загроза, яка багато років еволюціонувала, адаптуючись до різних ситуацій, платформ та операційних систем.

Яскравим прикладом використання вимагачів є RansomEXX (або Defrat777) це одна з найпоширеніших атак програм-вимагачів проти Linux 2020 та 2021 роках це програмне забезпечення-вимагач атакувало декілька високоякісних цілей, зокрема:

- Урядова мережа Бразилії.
- Департамент транспорту Техасу (TxDOT).
- Konica Minolta.
- IPG Photonics.
- Tyler Technologies.

RansomEXX — це 64-розрядний двійковий файл ELF на основі C, скомпільований з колекцією компіляторів GNU (GCC). Програма-вимагач керується людьми, тому учасникам загроз потрібен час, щоб зламати мережу, вкрасти облікові дані та поширити на пристрої. Після активації програма-вимагач генерує 256-бітний ключ, який шифрує файли в межах досяжності блочного шифру AES. Загальнодоступний RSA-4096 шифрує ключ AES, але атака також включає потік, який щосекунди повторно шифрує ключ AES.

На відміну від більшості троянських програм, RansomEXX не має:

- C&C зв'язок (C2)
- Припинення запущених процесів.
- Антианалітичні прийоми та пастки.

RansomEXX – це дуже цілеспрямована атака. Кожен зразок зловмисного програмного забезпечення містить твердо закодовану назву організації-жертви. І

зашифроване розширення файлу, і адреса електронної пошти для зв’язку зі зловмисниками використовують ім’я жертви.

Ще одним з найпоширеніших програм-вимагачів є Tycoon

Перші випадки використання цього програмного забезпечення-вимагача відбулися наприкінці 2019 року, коли хакери переслідували:

- Організації вищої освіти.
- Компанії в галузі програмного забезпечення.
- Малий і середній бізнес.

Корисне навантаження Tycoon — це заминований ZIP-архів із шкідливим компонентом Java Runtime Environment (JRE). Хакери компілюють програму-вимагач у файл образу Java, щоб приховати небезпеку.

Як правило, хакери Tycoon зламують систему через незахищений порт протоколу віддаленого робочого столу (RDP). Потрапивши всередину, зловмисники компілюють код в образ Java і створюють власну збірку JRE. Потім зловмисники виконують об’єкт Java за допомогою сценарію оболонки, шифруючи систему та залишаючи конфігураційний файл із записом про викуп.

Tycoon скремблує кожен файл за допомогою іншого ключа AES перед подальшим кодуванням даних за допомогою рівня RSA-1024. Як правило, у жертви є 60-годинне вікно, щоб заплатити біткойни в обмін на ключ дешифрування. ОС Linux і Windows уразливі до атак Tycoon.

Висновки за розділом 1

В першому розділі було проаналізовано структуру операційних систем з відкритим кодом, було також розглянуто їх види, більш детально розглянуто ОС Linux, Архітектуру її ядра, файлову систему, та принципи роботи. Також було проаналізовано можливі загрози для серверів на базі ОС Linux.

Операційна система з відкритим вихідним кодом — це операційна система, в якій вихідний код є загальнодоступним і доступним для редагування.

Найпопулярнішою є операційна система на базі Linux, яка використовується в більшості організаціях і компаніях, на якій зберігаються та оброблюються важливі дані організації тому є пріорітетною для атак.

Одним з найпоширеніших методів атак для серверів з ОС з відкритим кодом є використання вірусів, троянів, черв'яків та іншого шкідлого ПЗ. Крім того,було розглянуто одні з найбільш страшних таких випадків, атак на організації з використанням в шкоду яку вони принесли.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ МЕТОДІВ ЗАХИСТУ СЕРВЕРІВ НА БАЗІ ОПЕРАЦІЙНИХ СИСТЕМ LINUX

2.1 Методи захисту серверів на базі ОС Linux

Однією з перших речей, які ви повинні зробити після розгортання нового хмарного сервера, є переконатися, що він залишиться в безпеці. Linux пропонує безліч варіантів, які допоможуть запобігти несанкціонованому доступу та посилити безпеку системи. Деякі методи зможуть забезпечити захист серверів організації від несанкціонованого доступу і кібератак:

Шифрування трафіку. При підключенні до хмарного сервера весь трафік буде проходити через загальнодоступну мережу, яку будь-хто може підслухати, якщо ви не вжити заходів для захисту свого зв’язку. Краще за все уникати використання будь-яких незашифрованих протоколів передачі, таких як Telnet і FTP, або всього, що може надсилати паролі чи іншу конфіденційну інформацію у вигляді простого тексту. Замість цього краще використовувати SSH (Secure Shell), SCP (Secure Copy), SFTP (SSH File Transfer Protocol) або rsync для всіх потреб віддаленого керування та передачі файлів.

Протокол SSH пропонує безпечний зашифрований канал через загальнодоступну мережу, що дозволяє віддаленому входу та іншим мережевим службам працювати безпечно. Найбільш часто використовуваною реалізацією цього протоколу є OpenSSH, який включено в більшість операційних систем на базі Unix, таких як більшість дистрибутивів Linux та OS X, у середовищі Windows клієнт PuTTY SSH є популярною альтернативою. Перегляньте нашу статтю про підключення до сервера , щоб дізнатися більше.

Secure Copy або SCP — це вбудована функція OpenSSH, яка дозволяє просто передавати файли через зашифроване мережеве з’єднання. SCP використовує SSH

для передачі даних і забезпечує таку ж аутентифікацію та рівень безпеки, що й SSH. Нижче наведено два приклади копії одного файлу на віддалений сервер та з нього.

SFTP — це ще одна утиліта командного рядка, включена в OpenSSH, і за замовчуванням має встановлюватися в більшості операційних систем Unix. Як і SCP, він використовує SSH для безпечної передачі файлів через незахищену мережу. Користувачі Windows можуть отримати ту саму функціональність за допомогою WinSCP (Windows Secure Copy), який, як випливає з назви, реалізує функції SCP, а також SFTP.

rsync — ще одна утиліта, яка зазвичай зустрічається в системах Unix. Він пропонує передачу файлів через зашифровані канали, щоб синхронізувати копії файлу на двох комп'ютерах. Програма використовує SSH для встановлення початкового з'єднання між двома системами, а потім викликає rsync на віддаленому хості, щоб визначити, які частини файлу, що синхронізується, потрібно скопіювати.

Політики безпеки облікового запису користувача. Після першого входу на нещодавно розгорнутий хмарний сервер, створення нового облікового запису користувача для себе та ввімкнення контролю доступу sudo — це кілька важливих завдань, з яких слід почати. Sudo, що означає «суперкористувач робить», дозволяє виконувати дії, які в іншому випадку вимагали б облікового запису root. Це дозволяє уникнути щоденного входу в систему як root, замість цього використовувати привілеї sudo для виконання команд рівня root, коли це потрібно.

Використання sudo вважається гарною практикою для безпеки, і зазвичай воно встановлюється в більшості дистрибутивів Linux за замовчуванням. Щоб отримати максимальну віддачу від того, що пропонує sudo, і налаштувати безпечний доступ користувачів, дотримуйтесь нашого посібника з керування безпекою облікового запису користувача Linux.

Моніторинг аутентифікації входу. Реальність сучасного Інтернету така, що безпека сервера буде перевірена зловмисниками, рано чи пізно, сподіваючись знайти погано захищений вхід. Якщо ваш сервер працює навіть день, швидше за все, у вас уже були невдалі спроби входу з IP-адрес, відмінних від вашої. Більшість

дистрибутивів Linux зберігають журнали для аутентифікації з моменту їх першого завантаження. Різні системи можуть зберігати журнали під різними іменами

Незважаючи на те, що ваш хмарний сервер все ще має бути захищеним завдяки реалізаціям безпеки Linux за замовчуванням, ви не повинні бути спокійними і сподіватися, що так і залишиться. Є кілька потужних інструментів для зменшення кількості невдалих спроб входу та захисту від простого підбору пароля.

Fail2ban — це одна з таких систем запобігання вторгненням, яка працює разом із системою керування пакетами або брандмауером, встановленими на вашому сервері. Він зазвичай використовується для блокування спроб підключення після певної кількості невдалих спроб, фактично даючи користувачеві тайм-аут, перш ніж йому буде дозволено спробувати знову.

Використання SSH-ключі замість паролів. Паролі є стандартним способом аутентифікації майже для всього, і, хоча вони є надійними, їх часто можна вгадати за допомогою грубого примусу або списків словників, просто спробувавши кілька варіантів поширених паролів. Надійні та складні для вгадування паролі можуть знову стати проблемою для запам’ятовування та легко вводяться неправильно.

Інший варіант — використовувати SSH-ключі для аутентифікації, генеруючи пару довгих, практично неможливо зламати, кодів ключів. За допомогою цих ключів так званий відкритий ключ можна безпечно передати на ваш сервер, зберігаючи при цьому закритий ключ на власному комп’ютері.

Відкритий ключ можна використовувати лише для ідентифікації користувача, який має приватну частину пари.

Закритий ключ необхідно зберігати в безпеці, забезпечуючи доступ до нього лише у вас.

Налаштування фаєрволу.Загальні рішення для захисту будь-якого мережевого комп’ютера – це встановлення обмежень, до яких дозволені з’єднання. Це можна зробити за допомогою брандмауера, системи безпеки мережі, яка відстежує та контролює вхідний і вихідний мережевий трафік на основі заздалегідь визначених правил безпеки.

Панель керування UpCloud пропонує простий у налаштуванні брандмауер, який виступає як захист першої лінії для захисту вашого хмарного сервера. Брандмауер UpCloud працює спеціально на сервері, але ви можете копіювати налаштування брандмауера між вашими серверами. У вас також є можливість налаштувати брандмауер за допомогою одного з готових налаштувань, доступних у налаштуваннях правил брандмауера. Готові правила є простою відправною точкою для подальшого налаштування

Іншим варіантом на сервері Linux є використання вбудованого рішення під назвою iptables, яке входить до більшості дистрибутивів. У CentOS та інших варіантах Red Hat iptables часто постачається з деякими попередньо налаштованими правилами, тоді як сервери Ubuntu та Debian не застосовують жодних обмежень за замовчуванням.

Оновлення системи**.**Регулярно перевіряйте наявність оновлень на вашому сервері Linux. Нові вразливості час від часу виявляються і часто швидко усуваються. Переконайтеся, що ваш хмарний сервер має найновіші виправлення, щоб ваша система була актуальною та безпечною.

Мінімізування вразливих місць.Важливою частиною захисту хмарного сервера є не залишати відкритими будь-які непотрібні мережеві служби, які прослуховують вхідні з’єднання. У нещодавно розгорнутій системі Linux зазвичай відкритий лише порт SSH 22. Слід перевірити власний сервер шляхом сканування відкритих портів за допомогою мережевого інструмента під назвою Nmap.

Будь-які інші послуги, відкриті для загальнодоступної мережі, слід приділити пильну увагу. Переконайтеся, що ви знаєте, які служби ми використовуємо та наскільки безпечні їхні методи підключення. Слід вимкнути усі служби, які нам не потрібні.

Сканування на наявність шкідливих програм.Системи Linux, як правило, менш імовірно заражаються шкідливим програмним забезпеченням, оскільки перевірка з відкритим кодом і різноманітні конфігурації кінцевих користувачів ускладнюють пошук і використання вразливостей. Вашим основним захистом має бути превентивна спроба зупинити несанкціонований доступ, але це не може бути

єдиним заходом безпеки. Хоча ви можете не думати, що у системі щось є незвичайним, шкідлива програма може працювати непомітно протягом тривалого часу, перш ніж викликати тривожний трафік або пошкодження системи. Тому важливо регулярно сканувати свій хмарний сервер на наявність шкідливих програм, щоб переконатися, що він не заражений.

Багато продуктів безпеки покладаються на сигнатури файлів для виявлення шкідливих програм та інших шкідливих файлів. Ця техніка передбачає читання або сканування файлу та тестування, щоб перевірити, чи відповідає файл набору заздалегідь визначених атрибутів. Ці атрибути відомі як «підпис» зловмисного програмного забезпечення. Сигнатури зловмисного програмного забезпечення, які можуть зустрічатися в різних форматах, створюються постачальниками та дослідниками безпеки. Набори підписів збираються в базах даних, деякі з яких можуть бути загальнодоступними та спільними, а інші містяться у власних базах даних, призначених виключно для певного постачальника.

Щоб створити підпис для конкретного файлу зловмисного програмного забезпечення або сімейства файлів, аналітику з безпеки потрібен один або кілька (чим більше, тим краще) зразків файлу для роботи. Такі зразки можуть бути зібрані «в дикій природі» із заражених комп’ютерів, отримані з даркнету та інших місць, де автори шкідливих програм торгують своєю роботою, або зі спільних сховищ шкідливих програм, де дослідники безпеки (а в деяких випадках і громадськість) можуть поділитися відомими файлами зловмисного програмного забезпечення. Деякі популярні сховища зловмисного програмного забезпечення, доступні спеціалістам із безпеки, включають VirusTotal , Malpedia та MalShare

MalShare є одним із кількох сховищ шкідливих програм, доступних для дослідників

Як тільки постачальник має набір або «корпус» файлів для роботи, він починає досліджувати файли на предмет загальних характеристик. Ці характеристики можуть включати такі фактори, як розмір файлу, імпортовані чи експортовані функції, байти даних у певних позиціях («зміщення»), хеші , рядки для друку тощо.

Процес генерації підписів можна автоматизувати, але спочатку це часто робиться вручну фахівцями-аналітиками та інженерами зі зловмисного програмного забезпечення, особливо коли виявлено абсолютно нове сімейство шкідливих програм.

Незважаючи на те, що існує багато різних форматів для створення підписів, одним із найпопулярніших форматів, широко використовуваних сьогодні, є YARA , який дозволяє аналітикам шкідливих програм створювати підписи на основі текстових і двійкових шаблонів. Наприклад, на наступному зображенні зліва показано фрагмент коду відомого сімейства шкідливих програм, який розповсюджує APT загроза OceanLotus , а праворуч — підпис YARA для його виявлення.

Виявлення на основі сигнатур дає ряд переваг перед простим збігом хешування файлів . По-перше, за допомогою сигнатури, яка відповідає загальним рисам серед зразків, аналітики шкідливого програмного забезпечення можуть націлюватися на цілі сімейства шкідливих програм, а не лише на один зразок.

По-друге, сигнатури дуже універсальні і їх можна використовувати для виявлення багатьох видів зловмисного програмного забезпечення на основі файлів. Підписи можуть легко включати або виключати різні типи файлів, будь то сценарії оболонки, файли Python, файли Windows PE, файли Linux ELF або файли Mach-O macOS. Та сама база даних зловмисного програмного забезпечення і навіть те саме правило, якщо б було доречно, потенційно могли б сканувати та відповідати підпису майже в будь-якому типі файлів.

По-третє, формати сигнатур, такі як YARA, є дуже потужними і пропонують аналітикам шкідливих програм як широкий спектр логіки для визначення шкідливої поведінки, так і відносно простий формат, який легко написати та перевірити. Більше того, оскільки підписи засновані на тексті, одна база даних може містити багато тисяч, навіть мільйони підписів, але сама по собі не буде надмірно великою.

Спільний формат підпису, як-от YARA, також легко поширюватися між дослідниками та каналами даних розвідки загроз, забезпечуючи широке виявлення відомих шкідливих програм і захист від відомих загроз якомога більшої кількості користувачів комп’ютерів.

такі дослідники шкідливого програмного забезпечення, як SentinelLabs, регулярно публікують звіти про загрози, що містять правила YARA, які можуть використовуватися іншими постачальниками, підприємствами та навіть окремими особами, щоб допомогти їм покращити власні зусилля щодо виявлення.

Навіть коли постачальники використовують власні формати підписів, зазвичай не викликає проблем перекласти підпис із загальнодоступного формату, такого як YARA, у формат, що стосується постачальника, оскільки більшість форматів на основі підпису мають подібні можливості.

Виявлення на основі сигнатур було стандартом для більшості продуктів безпеки протягом багатьох років і продовжує відігравати важливу роль у боротьбі з відомим зловмисним програмним забезпеченням на основі файлів, але сьогодні передове рішення не може покладатися лише або навіть переважно на підписи файлів для виявлення. Деякі з причин цього пов’язані з тим, як суб’єкти загроз пристосувалися до ухилення від виявлення сигнатур, а деякі пов’язані з недоліками, властивими методу сканування файлу на наявність певних атрибутів.

Перший серйозний недолік використання сигнатур для виявлення зловмисного програмного забезпечення полягає в тому, що підписи можуть бути записані лише після того , як зразок зловмисного програмного забезпечення вже побачено. Це означає, що будь-яке рішення, яке покладається виключно на підписи, завжди буде на крок позаду останніх атак.

Друга велика проблема полягає в тому, що сьогодні унікальні зразки шкідливих програм створюються з такою швидкістю , що створення достатньо ефективних підписів не є реальною метою. Це є однією з причин, чому так багато рішень на основі сигнатур не вловлюють відомі шкідливі програми.

Окрім різноманітних шкідливих програм, іншим типом шкідливого програмного забезпечення, на який слід звернути увагу, є руткіти, які представляють собою сукупність програм, призначених для отримання доступу до комп’ютера або частин його ОС, які зазвичай обмежені, але водночас приховують свою присутність. Руткіти часто використовуються зловмисниками після отримання root-доступу до цільової системи. Незважаючи на те, що руткіти намагаються замаскувати своє

існування, існують інструменти, створені спеціально для виявлення відомих варіантів руткітів.

Впровадження системи виявлення вторгнень.Перевірка системи за допомогою сканерів зловмисного програмного забезпечення та інших подібних програм усе ще в основному є запланованими завданнями, що виконуються час від часу. Це дає будь-якому зловмисному програмному забезпеченню час між скануваннями, щоб вони залишалися непоміченими, можливо, навіть протягом тривалого періоду часу. Рішення для простою між очищенням зловмисного програмного забезпечення — налаштувати систему виявлення вторгнень (IDS), яка постійно стежить за вашим хмарним сервером та його мережевим трафіком.

Система запобігання вторгненню хоста (HIPS) новіша, ніж HIDS , з основною відмінністю в тому, що HIPS може вживати заходів для пом’якшення виявленої загрози. Наприклад, розгортання HIPS може виявити, що хост сканується портом, і заблокувати весь трафік з хоста, який здійснює сканування. HIPS часто контролює стан пам’яті, ядра та мережі, файли журналів і виконання процесу. HIPS також захищає від переповнення буфера.

Перевага запобігання вторгненню полягає в тому, що вам не потрібно чекати відповіді співробітника служби безпеки, перш ніж вживати заходів для підтримки цілісності хоста. Цей підхід може виявитися корисним, особливо тому, що останні дослідження показують, як уразливі системи можна скомпрометувати за лічені хвилини. HIPS часто базується як на ознаках, так і на аномалії. На відміну від систем на основі сигнатур, які можуть захистити лише від відомих поганих сигнатур, HIPS на основі аномалій намагається відрізнити нормальну поведінку від аномальної. Ця можливість допомагає, коли загроза або не має відомого підпису, або база даних сигнатур ще не оновлена.

Недоліками HIPS є те, що прийнята відповідь може зробити хост непотрібним або, можливо, вплинути на доступність критичного ресурсу. Одна справа, якщо IDS видає хибнопозитивний результат, але хибнопозитивний результат із якоюсь рефлексивною дією може бути гіршим.

Як і HIDS, функціональність HIPS може значно відрізнятися для кожного продукту. Хорошим прикладом успішного та ефективного HIPS є Immunix, система запобігання вторгненням (IPS) для Linux. По суті, Immunix бере контроль доступу, який зазвичай застосовується на рівні користувача, і застосовує його до програм. Адміністратори можуть налаштувати профілі, які точно вказують, що можуть виконувати певні виконувані файли, а що ні. Оскільки велика кількість атак передбачає зловживання програмним забезпеченням, цей метод запобігання вторгненню може бути дуже ефективним. У цьому випадку напади можна зупинити до їх появи.

Snort є популярним вибором для мережевої системи виявлення вторгнень (NIDS), вона з відкритим вихідним кодом, активно розвивається і досить легка, щоб її можна було встановити навіть на найменших хмарних серверах.

Іншим типом системи виявлення вторгнень є хостова система (HIDS), яка аналізує поведінку системи та стан конфігурації для виявлення потенційних порушень безпеки, компромісів, модифікацій критичних системних файлів, поширених руткітів і шкідливих процесів.

OSSEC є хорошим прикладом HIDS з відкритим кодом, який виконує аналіз журналів, перевірку цілісності файлів, моніторинг політики, виявлення руткітів, попередження в режимі реального часу та активну відповідь. OSSEC доступний для більшості операційних систем, включаючи найпоширеніші дистрибутиви Linux. Його призначено для налаштування на основі сервер-клієнт, де на критично важливих системах встановлюються дуже легкі клієнти, які потім надсилають свої звіти на сервер OSSEC для аналізу. Це ідеально підходить для користувачів із кількома хмарними серверами для централізованого моніторингу безпеки.

2.2 Засоби захисту від шкідливого програмного забезпечення в операційних системах Linux

Для виявлення такого виду загроз використовують Антивірусне програмне забезпечення.

Антивірусне програмне забезпечення працює в основному з використанням методів сигнатури та поведінки. При виявленні на основі сигнатур, коли дослідники виявляють будь-яке нове зловмисне програмне забезпечення, його підпис файлу витягується та додається до бази даних антивірусу. Під час перевірки безпеки антивірус перевіряє подібні сигнатури у файлах, які він сканує, і якщо знайдено будь-яку відповідність, це виявляється як зловмисне програмне забезпечення. При виявленні поведінки алгоритм або намір зловмисної програми розуміється, і вся підозріла поведінка додається до бази даних антивіруса. Під час виконання файлу антивірус стежить за виконуваною програмою і намагається знайти поведінку, яка відповідає базі даних. Якщо знайдено будь-яку відповідність, це виявляється як зловмисне програмне забезпечення. Крім цих двох методів, також використовуються інші методи, такі як виявлення пісочниці, методи аналізу даних, евристичний аналіз, виявлення руткітів і захист у реальному часі.

Sandboxing використовує віртуальне середовище для запуску підозрілої програми та ізоляції її від реального середовища. Потім він перевіряє поведінку програми та порівнює її з базою даних; якщо знайдено будь-який збіг, його позначають як зловмисне програмне забезпечення. Цей метод безпечний і захищає систему від будь-яких атак зловмисного програмного забезпечення.

Технологія аналізу даних використовує алгоритми машинного навчання для класифікації шкідливого програмного забезпечення на основі його поведінки.

Евристичний аналіз – це метод, який використовується для визначення жанру вірусу або шкідливого програмного забезпечення. Зазвичай зловмисники можуть створити варіанти вірусу, і може бути сімейство подібних вірусів, що утворюють жанр. Простіше визначити ознаку жанру, ніж виявити конкретний вірус. Евристичний аналіз створює сигнатурний жанр різних сімейств вірусів на основі їх поведінки, який зберігається в базі даних антивіруса. Коли поведінка програми відповідає жанру вірусу, вона позначається як вірус, що належить до цього сімейства.

Виявлення руткітів використовується, зокрема, для шкідливих програм під назвою руткіт, які можуть взяти адміністративний контроль над комп’ютером.

Руткіт може підробити антивірус, відключивши його, а також іноді дуже важко видалити. Антивірус у режимі реального часу сканує кожен процес, що виконується у фоновому режимі, у режимі реального часу. Цей захист дуже важливий і запобігає активації небажаного шкідливого програмного забезпечення.

Який би метод не використовувався антивірусним програмним забезпеченням, найскладнішим є запобігання будь-якому хибнопозитивному виявлення. Багато програм можуть вести себе як зловмисне програмне забезпечення або мати підпис файлу, який виглядає як зловмисне програмне забезпечення, але ніколи не завдає шкоди. Але багато антивірусних або антивірусних програм іноді виявляють такі програми як шкідливі.

Розглянемо декілька рішень для коп'ютерів на базі ОС Linux від шкідливого програмного забезпечення:

WebTitan — це фільтр веб-контенту на основі DNS, який блокує зловмисне програмне забезпечення, спроби фішингу та програм-вимагачів, а також забезпечує контроль веб-контенту для підприємств, навчальних закладів та державних провайдерів Wi-Fi.

NordLayer використовує загальний захист кінцевих точок, доступний у кожній основній операційній системі, обмежуючи потенційні ризики чутливого до людських помилок фішингу, шкідливої активності та застарілих випадків виправлення за допомогою ThreatBlock, виявлення зламаних/рутованих пристроїв та підвищення надійності сервера за допомогою спеціального DNS.

ESET Endpoint Security — це рішення для моніторингу мережі, яке допомагає підприємствам керувати процесами для виявлення загроз, блокування цілеспрямованих атак, запобігання злому даних і забезпечення захисту від програм-вимагачів. Це дозволяє користувачам відстежувати поведінку шкідливих процесів і розкривати сегменти пам'яті.

Ninja Protect забезпечує безпрецедентну видимість і контроль над кінцевими точками, надійне виправлення, посилення кінцевих точок, резервне копіювання та можливості захисту даних разом з провідними в галузі можливостями AV і EDR

наступного покоління від Bitdefender, Ninja Protect є цілісним рішенням для захисту кінцевих точок.

Splunk Enterprise — це хмарна платформа, призначена для допомоги підприємствам у управлінні великими даними та аналізі машинних даних. Основні функції включають візуалізацію даних, показники ефективності, збір даних, пошук у реальному часі, індексацію, відстеження KPI, звітування та моніторинг.

Automox — це хмарне рішення для кібергігієни та управління виправленнями, яке допомагає підприємствам оптимізувати захист кінцевих точок і мінімізувати кіберзагрози на всіх пристроях. Централізована платформа надає користувачам огляд пристроїв, які потребують схвалення виправлення, оновлення системи або усунення несправностей.

Heimdal Ransomware Encryption Protection — це інноваційний модуль на 100% без сигнатур, який забезпечує провідне на ринку виявлення атак програм-вимагачів, а також усунення у разі шифрування. Він ефективний як проти безфайлових, так і на основі файлів, тихо захищаючи всі цифрові активи.

Найбільш популярнішим рішенням для виявлення шкідливого ПЗ в Linux є ClamAV.

ClamAV пропонує хороший захист від шкідливих програм з відкритим кодом для Linux. Головна перевага цього антивірусу є його безкоштовність, але крім цього

Він має вірусну базу даних з відкритим вихідним кодом і може використовуватися для дослідження або вдосконалення, також у нього можливе сканування поштового шлюзу, також він має багатопотоковий віртуальний сканер від шкідливих програм.

ClamAV працює в основному зі службою виявлення вірусів, вірусною базою даних і інструментом freshclam. Інструмент freshclam допомагає ClamAV оновлювати свою базу даних, і його можна налаштувати за допомогою файлу freshclam.conf. Цей інструмент можна налаштувати на роботу в інтерактивному (на вимогу з командного рядка) режимі або демонному режимі (тихо у фоновому режимі). Він підтримує оновлення за сценарієм (замість передачі всього файлу CVD під час кожного оновлення, він лише передає відмінності між останньою та

поточною базою даних за допомогою спеціального сценарію), перевірку версії бази даних через DNS, проксі-сервери (з автентифікацією), цифрові підписи та різні сценарії помилок.

База вірусів оновлюється за посиланням http://database.clamav.net шляхом завантаження файлів .cvd, а оновлення можна автоматизувати, налаштувавши інструмент freshclam. Служба виявлення вірусів підключена до вірусної бази разом із механізмом захисту ClamAV. Різні програми з графічним інтерфейсом, такі як WinClam і ClamTk, можна використовувати разом із механізмом виявлення, але за замовчуванням ClamAV також можна використовувати за допомогою утиліти командного рядка.

Особливості ClamAV включають:

- Сканер командного рядка.
- Інтерфейс Milter для Sendmail.
- Розширений засіб оновлення баз даних з підтримкою оновлень за сценарієм та цифрових підписів.
- База вірусів оновлюється кілька разів протягом дня. Вбудована підтримка всіх стандартних форматів поштових файлів.
- Вбудована підтримка різних форматів архівів, включаючи Zip, RAR, Dmg, Tar, Gzip, Bzip2, OLE2, Cabinet, CHM, BinHex, SIS та інші.
- Вбудована підтримка виконуваних файлів ELF і портативних виконуваних файлів, упакованих з UPX, FSG, Petite, NsPack, wwpack32, MEW, Upack та обфусцованих за допомогою SUE, Y0da Cryptor та інших.
- Вбудована підтримка популярних форматів документів, включаючи файли MS Office і MacOffice, HTML, Flash, RTF і PDF.

Гарним рішенням для аналізу ПО на шкідливий код є так звані пісочниці.

Пісочниця — це ізольоване середовище тестування, яке дозволяє користувачам запускати програми або відкривати файли, не впливаючи на програму, систему чи платформу, на якій вони працюють.

Розробники програмного забезпечення використовують пісочниці для тестування нового програмного коду. Фахівці з кібербезпеки використовують

пісочниці для тестування потенційно шкідливого програмного забезпечення. Без пісочниці програмне забезпечення чи програми могли б мати потенційно необмежений доступ до всіх даних користувача та системних ресурсів у мережі.

Пісочниці також використовуються для безпечного виконання шкідливого коду, щоб уникнути шкоди хост-пристрою, мережі або іншим підключеним пристроям. Використання пісочниці для виявлення зловмисного програмного забезпечення пропонує додатковий рівень захисту від загроз безпеці, таких як приховані атаки та експлойти, які використовують уразливості нульового дня.

Оскільки зловмисне програмне забезпечення стає все більш складним, моніторинг підозрілої поведінки для виявлення зловмисного програмного забезпечення стає все складнішим. Багато загроз за останні роки використовували передові обфускації , які можуть уникнути виявлення кінцевими точками та продуктами безпеки мережі.

Sandboxing захищає критичну інфраструктуру організації від підозрілого коду, оскільки вона працює в окремій системі. Це також дозволяє ІТ тестувати шкідливий код в ізольованому середовищі тестування, щоб зрозуміти, як він працює, а також швидше виявляти схожі атаки зловмисного програмного забезпечення.

Загалом, пісочниця використовується для перевірки підозрілих програм, які можуть містити віруси чи інше шкідливе програмне забезпечення, не дозволяючи програмному забезпеченню завдати шкоди хост-пристроям.

Використання пісочниці для тестування змін програмного забезпечення перед їх запуском означає, що під час і після тестування буде менше проблем, оскільки середовище тестування повністю відокремлено від виробничого середовища.

Пісочниця також чудово підходить для поміщення на карантин загроз нульового дня, які використовують незаявлені вразливості. Хоча немає гарантії, що пісочниця зупинить загрози нульового дня , вона пропонує додатковий рівень безпеки, відокремлюючи загрози від решти мережі. Коли загрози та віруси поміщені на карантин, експерти з кібербезпеки можуть вивчати їх, щоб виявити закономірності, допомагаючи запобігти майбутнім атакам та виявити інші вразливості мережі.

Пісочниця також доповнює інші програми безпеки, включаючи моніторинг поведінки та вірусні програми. Він забезпечує додатковий захист від певних штамів шкідливих програм, які антивірусна програма може не виявити. Більш просунуті зловмисне програмне забезпечення може перевірити, чи працює воно в пісочниці, перш ніж виконувати.

Висновки за 2 розділом

В другому розділі було розглянуто методи захисту серверів на базі ОС Linux, а саме:

Шифрування трафіку. При підключенні до хмарного сервера весь трафік буде проходити через загальнодоступну мережу, яку будь-хто може підслухати, якщо ви не вжити заходів для захисту свого зв’язку.

Політики безпеки облікового запису користувача. Після першого входу на нещодавно розгорнутий хмарний сервер, створення нового облікового запису користувача для себе та ввімкнення контролю доступу sudo — це кілька важливих завдань, з яких слід почати.

Впровадження системи виявлення вторгнень.Перевірка системи за допомогою сканерів зловмисного програмного забезпечення та інших подібних програм усе ще в основному є запланованими завданнями, що виконуються час від часу.

Також було розглянуто популярні рішення захисту ОС Linux від шкідливого ПЗ, різні антивіруси, та комплексні рішення в яких входить сканування системи на вміст шкідливих файлів. Б

Крім того було розглянуто сервіси для детальнішого аналізу шкідливого ПЗ – пісочниці

Пісочниця — це ізольоване середовище тестування, яке дозволяє користувачам запускати програми або відкривати файли, не впливаючи на програму, систему чи платформу, на якій вони працюють.

РОЗДІЛ 3
ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗАЦІЇ АНАЛІЗУ ШКІДЛИВОГО КОДУ

3.1 Можливості VirusTotal

VirusTotal одна з найпопуляніших служб, що перевіряє можливі шкідливі файли на віруси, хробаки та трояни, що полегшує процес виявлення шкідливого ПЗ.

VirusTotal перевіряє елементи за допомогою понад 70 антивірусних сканерів і служб блокування URL-адрес/доменів, а також безлічі інструментів для вилучення сигналів із досліджуваного вмісту. Будь-який користувач може вибрати файл зі свого комп’ютера за допомогою свого браузера та надіслати його до VirusTotal. VirusTotal пропонує ряд методів подання файлів, включаючи основний загальнодоступний веб-інтерфейс, програми для завантаження на робочий стіл, розширення для браузера та програмний API. Веб-інтерфейс має найвищий пріоритет сканування серед загальнодоступних методів подання. Подання можуть бути написані будь-якою мовою програмування за допомогою відкритого API на основі HTTP.

Як і у випадку з файлами, URL-адреси можна надіслати кількома різними способами, включаючи веб-сторінку VirusTotal, розширення браузера та API.

Після подання файлу або URL-адреси основні результати передаються заявнику, а також партнерам, які досліджують, які використовують результати для покращення власних систем. Як результат, надсилаючи файли, URL-адреси, домени тощо до VirusTotal, ви сприяєте підвищенню глобального рівня IT-безпеки.

Цей основний аналіз також є основою для кількох інших функцій, включаючи VirusTotal Community: мережу, яка дозволяє користувачам коментувати файли та URL-адреси та обмінюватися нотатками один з одним. VirusTotal може бути корисним для виявлення шкідливого вмісту, а також для виявлення помилкових спрацьовувань — звичайних і нешкідливих елементів, виявлених як шкідливі одним або кількома сканерами.

Зведені дані VirusTotal є результатом багатьох різних антивірусних систем, сканерів веб-сайтів, інструментів аналізу файлів і URL-адрес, а також внесків користувачів. Інструменти характеристики файлів і URL-адрес, які ми об’єднуємо, охоплюють широкий спектр цілей: евристичні механізми, відомі погані підписи, вилучення метаданих, ідентифікація шкідливих сигналів тощо.

Звіти про сканування, створені VirusTotal, надаються загальнодоступній спільноті VirusTotal. Користувачі можуть залишати коментарі та голосувати за те, чи є певний вміст шкідливим. Таким чином, користувачі допомагають поглибити колективне розуміння спільноти потенційно шкідливого вмісту та виявляють помилкові результати (тобто нешкідливі елементи, виявлені одним або кількома сканерами як шкідливі).

Вміст надісланих файлів або сторінок також може надаватися преміальним клієнтам VirusTotal. Корпус файлів, створений у VirusTotal, надає фахівцям із кібербезпеки та розробникам продуктів безпеки цінну інформацію про поведінку нових кіберзагроз та шкідливого програмного забезпечення. Через нашу комерційну пропозицію преміальних послуг VirusTotal надає кваліфікованим клієнтам і антивірусним партнерам інструменти для виконання складних пошуків на основі критеріїв для виявлення та доступу до зразків шкідливих файлів для подальшого вивчення. Це допомагає організаціям виявляти й аналізувати нові загрози та створювати нові засоби пом’якшення й захисту.

Сигнатури зловмисного програмного забезпечення часто оновлюються компанією VirusTotal, оскільки вони розповсюджуються антивірусними компаніями, що гарантує, що наша служба використовує найновіші набори сигнатур.

Сканування веб-сайтів у деяких випадках виконується шляхом запиту до баз даних постачальників, які спільно використовували VirusTotal і які зберігаються в наших приміщеннях, а в інших випадках за допомогою API-запитів до рішення антивірусної компанії. Таким чином, як тільки певний учасник заблокує URL-адресу, це негайно відображається у вердиктах користувачів.

VirusTotal також має API версії 3,що є стандартним і рекомендованим способом програмної взаємодії з VirusTotal. Цей API відповідає REST і має

передбачувані, орієнтовані на ресурси URL-адреси. Він використовує JSON для запитів і відповідей, включаючи помилки.

Незважаючи на те, що багато кінцевих точок і функцій, наданих API VirusTotal, є вільно доступними для всіх зареєстрованих користувачів, існує також Premium API,що має ряд приємних доповнень. Перевагами Premium API над стандартою є:

● Premium API не має обмежень щодо кількості запитів чи денної надбавки, ліміти регулюються ліцензією

● Premium API повертає більше контексту загроз і відкриває розширені кінцеві точки та функціональні можливості пошуку загроз і виявлення шкідливих програм.

● Premium API регулюється угодою про рівень обслуговування, яка гарантує готовність даних.

● Premium API підтримує такі популярні випадки використання :

● автоматичної телеметрії безпеки (сповіщення/події**)** , що веде до сортування попереджень, оркестрування на основі розвідки загроз і активного пошуку загроз.

● Повна характеристика будь-якого типу кампанії загроз, яку можна спостерігати: файли, хеші, URL-адреси, домени, IP-адреси, сертифікати SSL тощо.

● Інтеграція з SIEM, SOAR, EDR або AV для цілей збагачення, а не виявлення. Досягнення єдиного скляного прозріння.

● Впровадження програмних робочих процесів для допомоги в реагуванні на інциденти, що призводить до швидших, точніших і впевненіших рішень.

● Завантаження файлів для подальшого вивчення та розтину в автономному режимі , у внутрішніх пісочницях, конвеєрах аналізу тощо.

● Автоматичне надходження сповіщень про правила YARA для створення користувацьких каналів загроз, покриття «сліпих зон» стека безпеки та отримання IoC, які можна використовувати для проактивної нейтралізації кампаній загроз.

Для автоматизації процесу перевірки ПЗ на шкідливий код, було використано стандартну версію VirusTotal API.

3.2 Програмна реалізація утиліти для сканування шкідливого ПЗ

В даній дипломній роботі після аналізу теоритичних відомостей з попередніх розділів було написано програму для аналізу файлів на шкідливий код. Це було зроблено за допомогою публічного API від VirusTotal.

Програма рекурсивно переглядає файли, обчислює їх хеш SHA256 і з перевіряє збіги з хеш в базі VirusTotal.

Розглянемо цікаві функції програми

```
class simpleFile:
    def calculate_hash(self, file_name):
        sha256_hash = hashlib.sha256()
        with open(file_name,'rb') as f:
            # Read and update hash string value in blocks of 4K to avoid buffer overflow
            for byte_block in iter(lambda: f.read(4096),b""):
                sha256_hash.update(byte_block)
        return(sha256_hash.hexdigest())
    def __init__(self, file_name):
        self.file_name = file_name
        self.hash = self.calculate_hash(file_name)
    def get_hash(self):
        return(self.hash)
    def get_file_name(self):
        return(self.file_name)
```

В класі simpleFile функція def caclculate_hash() обчислює хеш для подальшої перевірки на збіги.

```
def add_virustotal_result(self, result):
        self.vt_result = result
        json_data = json.loads(json.dumps(result))
        try:
            if json_data['results']['response_code'] == 1:
```

```
            self.total_scanners = json_data['results']['total']
            self.positives = json_data['results']['positives']
            self.scan_date = json_data['results']['scan_date']
        except KeyError:
            print("Received unexpected response from VirusTotal:")
            print(result)
            sys.exit(f"\nReceived invalid response from VirusTotal. Did you enter a valid VT
API Key in the config file?")
```

В функції def add_virustotal_result() йде процес конвертації результатів збігів з Virustotal в json формат для аналізу співпадінь.

```
  def get_virustotal_result(self):
        return(self.vt_result)
    def is_malicious(self):
        return(self.count_alerting_scanners() / self.count_total_scanners() >=
self.ALERTING_LEVEL)
    def count_total_scanners(self):
        # number of AV scanners that were used to check this file
        return(self.total_scanners)
    def count_alerting_scanners(self):
        # number of AV scanners that reported the file as malicious
        return(self.positives)
```

Функції:

- def_virustotal_result() – повертає значення результатів сканування
- def is_malicious() – повертає значення параметру alerting level, який як описувалось вище, відповідає за поріг який необхідно перевищити, щоб розпізнати файл як шкідливий
- def count_total_scanners() – повертає значення кількості сканерів з VirusTotal
- def count_alerting_scanners() – повертає значення сканерів які, ідентифікували файл як шкідливий.

```
class entityHandler:
    def __init__(self):
        self.hash_dict = {}
    def add_file(self, file, alerting_level):
        new_file = simpleFile(file)
        existing_duplicates = self.hash_dict.get(new_file.get_hash())
        if existing_duplicates is not None:
            existing_duplicates.add_file_name(new_file.get_file_name())
        else:
            self.hash_dict.update({new_file.get_hash():observedEntity(new_file,
alerting_level)})
    def get_entities(self):
        return(self.hash_dict.items())
    def count_entities(self):
        return(len(self.hash_dict))
    def retrieve_virustotal_results(self):
        if entity_handler.count_entities() <= 4:
            waiting_time = 0
        else:
            waiting_time = 15
        i = 0
        for hash, observed_entity in self.get_entities():
            i+=1
            print(f'Processing {i} out of {self.count_entities()}...')
            observed_entity.add_virustotal_result(vt.get_file_report(hash))
            time.sleep(waiting_time)
```

Класс entityHandler додає об’єкти, які використовуватимуться для аналізу шкідливого пз або оновлює інформацію чинних.

Функція def add_file() – первіряє , чи були вже оброблені інші файли з таким же хешем.

Функція def get_entities() – повертає ітерацію всіх об’єктів, , які використовуватимуться для аналізу шкідливого пз, щоб їх можна було перевірити

Функція def count_entities() - кількість об’єктів (тобто файлів з унікальним хешем)

Функція def retrieve_virustotal_results() – проводить процес аналізу файлів, шляхом порівняння, отриманих даних з базою VirusTotal.

3.3 Тестовий приклад

Дану програму, звісно, покращити, наприклад, використовувати з іншими сервісами для аналізу шкідливого. Також можна використовувати, для сканування папок з завантаженнями, чи системи в цілому, додавши скрипт в автозавантаження і додати графік проведення сканування.

Проведемо тестове сканування.

Для тесту програми було використано базу з вірусами theZoo.

theZoo – це проект, створений для того, щоб зробити можливість аналізу шкідливих програм відкритими та доступними для громадськості. Оскільки, майже всі версії зловмисного програмного забезпечення дуже важко знайти в такий спосіб, який дозволяє аналізувати, команда проєкту вирішили зібрати їх усі доступним і безпечним способом.

Підготуємо директорію для тесту (Рис.3.1).

```
root@vultr:~/test/Linux.Chapros.A# ls
'linux-chapros_ E022DE72CCE8129BD5AC8A0675996318'   the_zeus_binary_chapros
```

Рисунок 3.1 – Вміст Директорії для тестування програми

В даній директорії попередньо було створено 2 файли: config та virus.txt – це пусті файли та скопійовано шкідливі файли з бази theZoo. Також в даній директорії міститься, директорія Linux.Chapros.A, для перевірки рекурсивного сканування(Рис. 3.2).

```
root@vultr:~/test/Linux.Chapros.A# ls
'linux-chapros_ E022DE72CCE8129BD5AC8A0675996318'   the_zeus_binary_chapros
 the_injected_iFrame_java-cve-2012-1723
root@vultr:~/test/Linux.Chapros.A#
```

Рисунок 3.2 – Вміст директорії Linux.Chapros

Перед запуском програми, також було створено файл конфігурації ‘root\config.yaml’ з параметрами необхідними для коректної роботи програми(Рис.3.3)

Рисунок 3.3 – Вміст файлу конфігурації root\config.yaml

В файлі конфігурації можемо побачити:

file_path – директорія в якій проводитиметься перевірка

recursive – параметр, так званого режиму рекурсивної перевірки, тобто перевірки всіх підпапок, та файлів в них, які знаходяться в директорії в якій буде проводитись сканування

alerting_level - парметр відповідає за поріг який необхідно перевищити, щоб розпізнати файл як шкідливий, в даному випадку значення параметру 0.1 – це означає, що потрібно хоча б 10% усіх механізмів повідомили, що файл є шкідливим.

api_key – особистий ключ API VirusTotal.

Запустимо програму.(Рис. 3.4)

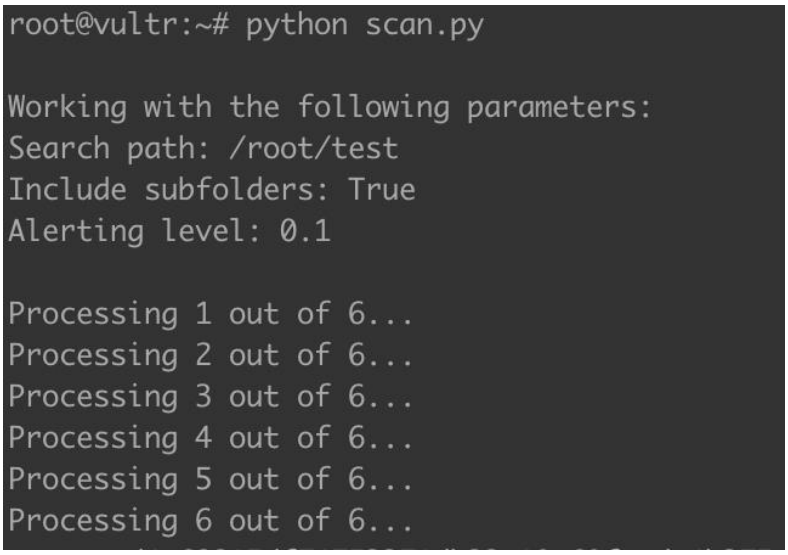

Рисунок 3.4 – Процес сканування файлів директорії /root/test

Після запуску можна побачити з якими параметрами його було запущено:

Search path, відповідає параметру file_path в файлі конфігурації root\config.yaml – директорія в якій проходить перевірка

Include subfolders, відповідає параметру recursive в файлі конфігурації root\config.yaml – параметр відповідає за рекурсивне сканування

Alerting level(alerting_level) – парметр відповідає за поріг який необхідно перевищити, щоб розпізнати файл як шкідливий

Після цього йде процес, аналізу кожного файлу в теці /root/test/ та в допоміжних теках, що знаходяться в цій директорії

Далі можемо побачити хеш потенційно шкідливих файлів та шлях до них, також вкінці видно, що з 6 перевірених файлів, 4 можуть загрожувати вашому серверу чи ПК.

Рисунок 3.5 – Хеш та шлях до знайдених потенційно шкідливих файлів

Висновки за розділом 3

В третьому розділі було розглянуто можливості сервісу для аналізу шкідливого ПЗ VirusTotal та на основі його API побудовано програму для сканування та виявлення шкідливого ПЗ в системі.

Було протестовано програму на сервері на базі операційної системи Linux з використанням, бази з вірусами theZoo.

theZoo – це проект, створений для того, щоб зробити можливість аналізу шкідливих програм відкритими та доступними для громадськості. Оскільки ми з’ясували, що майже всі версії зловмисного програмного забезпечення дуже важко

знайти в такий спосіб, який дозволяє аналізувати, ми вирішили зібрати їх усі для вас доступним і безпечним способом.

ВИСНОВКИ

В дипломній роботі було реалізовано програму для сканування та аналізу потенційно шкідливих файлів. Програма Використовувала API популярного сервісу для виявлення шкідливого ПЗ VirusTotal

В першій частині дипломної роботи була розглянута, струтура операційних систем з відкритим кодом, їх види та більш детально було розглянуто ОС Linux її архітектуру, файлову систему та види. Також було розглянуто різні дистрибутиви та їх призначення.

Найпопулярнішою є операційна система на базі Linux, яка використовується в більшості організаціях і компаніях, на якій зберігаються та оброблюються важливі дані організації тому є пріорітетною для атак.

Одним з найпоширеніших методів атак для серверів з ОС з відкритим кодом є використання вірусів, троянів, черв’яків та іншого шкідлого ПЗ. Крім того,було розглянуто одні з найбільш страшних таких випадків, атак на організації з використанням в шкоду яку вони принесли.

В другій частині дипломної роботи були розглянуті методи та засоби захисту серверів на базі ОС Linux.

Шифрування трафіку. При підключенні до хмарного сервера весь трафік буде проходити через загальнодоступну мережу, яку будь-хто може підслухати, якщо ви не вжити заходів для захисту свого зв’язку.

Політики безпеки облікового запису користувача. Після першого входу на нещодавно розгорнутий хмарний сервер, створення нового облікового запису користувача для себе та ввімкнення контролю доступу sudo — це кілька важливих завдань, з яких слід почати.

Впровадження системи виявлення вторгнень.Перевірка системи за допомогою сканерів зловмисного програмного забезпечення та інших подібних програм усе ще в основному є запланованими завданнями, що виконуються час від часу.

Також було розглянуто популярні рішення захисту ОС Linux від шкідливого ПЗ, різні антивіруси, та комплексні рішення в яких входить сканування системи на вміст шкідливих файлів. Б

Крім того було розглянуто сервіси для детальнішого аналізу шкідливого ПЗ – пісочниці

Пісочниця — це ізольоване середовище тестування, яке дозволяє користувачам запускати програми або відкривати файли, не впливаючи на програму, систему чи платформу, на якій вони працюють.

В третій частині дипломної роботи, було розроблено програму для сканування та аналізу на файлів на шкідливий код за допомогую API сервісу VirusTotal, та протестовано на VPS сервері.

Виходячи із поставленої мети дипломної роботи були виконані наступні завдання:

- дослідити структуру операційних систем з відкритим кодом
- проаналізувати засоби та механізми виявлення шкідливого програмного забезпечення
- розробити програмну реалізацію для виявлення виявлення шкідливого програмного забезпечення

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What are open-source operating systems? Everything you need to know [Електронний ресурс]:https://www.zdnet.com/article/what-are-open-source-operating-systems-everything-you-need-to-know/
2. Open-Source Operating System [Електронний ресурс]: https://www.javatpoint.com/open-source-operating-system
3. What is Linux? [Електронний ресурс]: https://www.linux.com/what-is-linux/
4. Operating System - Linux [Електронний ресурс]: https://www.tutorialspoint.com/operating_system/os_linux.htm
5. What is Linux Operating System? Introduction to Linux OS [Електронний ресурс]: https://www.guru99.com/introduction-linux.html
6. Kernel Architecture Of Linux [Електронний ресурс]: https://www.engineersgarage.com/kernel-architecture-of-linux-part-7-15/
7. Best Antivirus Software for Linux in 2021 [Електронний ресурс]: https://linuxhint.com/best-antivirus-software-linux/
8. Linux Ransomware: Famous Attacks and How to Protect Your System [Електронний ресурс]: https://phoenixnap.com/blog/linux-ransomware
9. FBI’s report points out that cybercrime is on the rise, with a focus on phishing and BEC [Електронний ресурс]: https://gatefy.com/blog/key-points-fbi-internet-crime-report-ic3-2020/
10. 8 Best Open Source Operating System 2022 [Електронний ресурс]: https://wethegeek.com/8-best-open-source-operating-system/
11. 10 Best Free and Open Source Operating System Examples [Електронний ресурс]: https://www.techjockey.com/blog/open-source-operating-system-example
12. 7 steps to securing your Linux server [Електронний ресурс]: https://opensource.com/article/19/10/linux-server-security
13. 40 Linux Server Hardening Security Tips [Електронний ресурс]: https://www.cyberciti.biz/tips/linux-security.html

14. How Secure Is Linux? [Електронний ресурс]: https://linuxsecurity.com/features/how-secure-is-linux

15. What is a Sandbox? [Електронний ресурс]: https://www.proofpoint.com/us/threat-reference/sandbox

16. VirusTotal API v3 Overview [Електронний ресурс]: https://developers.virustotal.com/reference/overview

17. The 8 Best Free Anti-Virus Programs for Linux [Електронний ресурс]: https://www.tecmint.com/best-antivirus-programs-for-linux/

18. Linux Security and Service Protection Methods [Електронний ресурс]: https://documentation.suse.com/sles/12-SP5/html/SLES-all/cha-security-protection.html

19. Host Intrusion Prevention System [Електронний ресурс]: https://www.sciencedirect.com/topics/computer-science/host-intrusion-prevention-system

20. Virus Signature [Електронний ресурс]: https://www.sciencedirect.com/topics/computer-science/virus-signature

ДОДАТКИ

ДОДАТОК А

Код програми для сканування ПЗ

```
import yaml
import sys
import json
import hashlib
import glob
import os
import time
from virus_total_apis import PublicApi as VirusTotalPublicApi
import argparse

class simpleFile:

    def calculate_hash(self, file_name):
        sha256_hash = hashlib.sha256()
        with open(file_name,'rb') as f:
            for byte_block in iter(lambda: f.read(4096),b""):
                sha256_hash.update(byte_block)

        return(sha256_hash.hexdigest())

    def __init__(self, file_name):
        self.file_name = file_name
        self.hash = self.calculate_hash(file_name)

    def get_hash(self):
        return(self.hash)

    def get_file_name(self):
        return(self.file_name)

class observedEntity:

    def __init__(self, file, alerting_level):
        self.files = []
        self.files.append(file.get_file_name())
        self.hash = file.get_hash()
        self.isMalicious = False
```

```
        self.vt_result = ''
        self.positives = 0
        self.total_scanners = 1
        self.ALERTING_LEVEL = alerting_level

    def add_file_name(self, file_name):
        self.files.append(file_name)

    def get_file_names(self):
        return(self.files)

    def get_hash(self):
        return(self.hash)

    def add_virustotal_result(self, result):
        self.vt_result = result

        json_data = json.loads(json.dumps(result))
        try:
            if json_data['results']['response_code'] == 1:
                self.total_scanners = json_data['results']['total']
                self.positives = json_data['results']['positives']
                self.scan_date = json_data['results']['scan_date']
        except KeyError:
            print("Received unexpected response from VirusTotal:")
            print(result)
            sys.exit(f"\nReceived invalid response from VirusTotal. Did you enter a valid VT API Key in the config file?")


    def get_virustotal_result(self):
        return(self.vt_result)

    def is_malicious(self):
        return(self.count_alerting_scanners() / self.count_total_scanners() >= self.ALERTING_LEVEL)

    def count_total_scanners(self):
        return(self.total_scanners)

    def count_alerting_scanners(self):
        return(self.positives)
```

```
class entityHandler:

    def __init__(self):
        self.hash_dict = {}

    def add_file(self, file, alerting_level):
        new_file = simpleFile(file)
        existing_duplicates = self.hash_dict.get(new_file.get_hash())
        if existing_duplicates is not None:
            existing_duplicates.add_file_name(new_file.get_file_name())
        else:
            self.hash_dict.update({new_file.get_hash():observedEntity(new_file,
alerting_level)})

    def get_entities(self):
        return(self.hash_dict.items())

    def count_entities(self):

        return(len(self.hash_dict))

    def retrieve_virustotal_results(self):

        if entity_handler.count_entities() <= 4:
            waiting_time = 0
        else:
            waiting_time = 15

        i = 0
        for hash, observed_entity in self.get_entities():
            i+=1
            print(f'Processing {i} out of {self.count_entities()}...')
            observed_entity.add_virustotal_result(vt.get_file_report(hash))

            time.sleep(waiting_time)


CONFIG_FILE= str(os.path.dirname(os.path.abspath(__file__)))+'\\config.yaml'
try:
    with open(CONFIG_FILE, 'r') as config_file:
        config = yaml.safe_load(config_file)
except FileNotFoundError:
    print(f"There was no valid {CONFIG_FILE} file in the directory of this script.")
```

```
    print("The file will be created for you, but you still need to enter your valid VirusTotal API key.")
    default_yaml = f"""
virustotal:
  api_key: enter your API key
  alerting_level: 0.1
file_path: {os.getcwd()}
recursive: True
"""
    with open(CONFIG_FILE, 'w') as config_file:
        yaml.dump(yaml.safe_load(default_yaml), config_file, default_flow_style=False)

    sys.exit(f"\nNo valid API key in {CONFIG_FILE} file found.")

VT_KEY = config['virustotal']['api_key']
ALERTING_LEVEL = config['virustotal']['alerting_level']
IS_RECURSIVE = config['recursive']
FILE_PATH = config['file_path']


parser = argparse.ArgumentParser()

parser.add_argument("-p", "--path", default=FILE_PATH, help="Directory of files to be checked, e.g. C:\\SuspiciousFiles\\")
parser.add_argument("-a", "--alertlv", default=ALERTING_LEVEL, type=float, help="Percentage of reporting scanners to define a file as malicious, e.g. 0.1")
parser.add_argument("-r", "--recursive", metavar="recursive", default=IS_RECURSIVE, choices=['True', 'False'], help="Include subfolders into search or not, e.g. True")

args = parser.parse_args()
FILE_PATH = args.path
ALERTING_LEVEL = args.alertlv
IS_RECURSIVE = args.recursive

print(f"""
Working with the following parameters:
Search path: {FILE_PATH}
Include subfolders: {IS_RECURSIVE}
Alerting level: {ALERTING_LEVEL}
""")

vt = VirusTotalPublicApi(VT_KEY)

entity_handler = entityHandler()
```

```
for file in glob.iglob(FILE_PATH+'/**/*', recursive=IS_RECURSIVE):
    if os.path.isfile(file):

        entity_handler.add_file(file, ALERTING_LEVEL)


entity_handler.retrieve_virustotal_results()

findings_counter = 0
for hash, observed_entity in entity_handler.get_entities():
    if observed_entity.is_malicious():
        findings_counter+=1
        print(f'====== {hash} ======')
        print('Potentially malicious hash for the following (identical) files:')

        i = 0
        for f in observed_entity.get_file_names():
            i+=1
            print(f'{i}: {f}')

        print(f'\n{observed_entity.count_alerting_scanners()} out of
{observed_entity.count_total_scanners()} scanners identified this file as malicious.')
        print('-------------------------------------------------------\n\n\n')

print(f'Finished processing {entity_handler.count_entities()} files. {findings_counter}
findings were reported.')
```