МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____С.В. Казмірчук

«_____»_____ 20__р

На правах рукопису УДК

004.725, 004.418 (043.3)

ДИПЛОМНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: Програмний модуль моніторингу вразливостей Linux систем

Виконавець: А.В.Давиденко

Керівник: к.т.н. О.О.Висоцька

Нормоконтролер: к.т.н. О.О.Висоцька

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Бакалавр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.В. Казмірчук

«__» _____ 20__ р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Давиденка Артема Володимировича

Тема: Програмний модуль моніторингу вразливостей Linux систем

затверджена наказом ректора від «26» квітня 2021 р. № 652/ст.

Термін виконання: з 10.05.2021р. по 20.06.2021р.

Вихідні дані: проаналізувати існуючі системи та методики аналізу і оцінки ризиків інформаційної безпеки; на основі аналізу виділити вхідні і вихідні параметри, завдяки яким можливо провести порівняння існуючих систем, виявлення їх переваг і недоліків; розробити методику, алгоритм та програмне забезпечення системи аналізу і оцінки ризиків.

Зміст пояснювальної записки: аналіз існуючих систем та методик аналізу і оцінки ризиків інформаційної безпеки; розробка методики системи аналізу та оцінки ризиків на основі нечіткої логіки; розробка програмного забезпечення запропонованої системи, верифікація отриманих результатів

КАЛЕНДАРНИЙ ПЛАН
виконання дипломної роботи

Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
Уточнення постановки задачі	21.04.2021	*Виконано*
Аналіз літературних джерел	30.04.2021	*Виконано*
Обґрунтування рішення	15.05.2021	*Виконано*
Збір інформації	16.04.2021	*Виконано*
Аналіз та дослідження моніторингу вразливостей Linux систем	17.05.2021	*Виконано*
Розробка та дослідження програмного модулю моніторингу	20.05.2021	*Виконано*
Тестування програмного модулю моніторингу	25.05.2021	*Виконано*
Оформлення і друк пояснювальної записки	28.05.2021	*Виконано*
Перевірка на антиплагіат	30.05.2021	*Виконано*
Оформлення презентації	01.06.2021	*Виконано*
Отримання рецензій від рецензентів	14.06.2021	*Виконано*

Здобувач вищої освіти Давиденко А.В.
(підпис, дата)

Керівник дипломної роботи Висоцька О.О.
(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, загальним обсягом робота складає 60 60 сторінок, 20 рисунків, 11 джерел.

Метою цієї дипломної роботи є створення програмного модулю моніторингу вразливостей Linux систем.

Для посягнення поставленої мети необхідно розв'язати наступні задачі:

- Проаналізувати архітектуру Linux, вразливості системи та моніторинг.
- Розробити програмний модуль моніторингу вразливостей Linux систем.
- Провести тестування розробленого програмного модуля моніторингу вразливостей Linux систем.

Об’єктом дослідження є процес пошуку вразливостей в Linux системах.

Методи дослідження проведені дослідження базуються на використанні мови програмування “Lua” на платформі Nmap Scripting Engine та сучасних методів і засобів виявлення вразливостей Linux систем.

Предметом дослідження є методи та програмне забезпечення для виявлення вразливостей у linux системах.

Практична цінність полягає в тому, що в роботі розроблено програмний модуль моніторингу вразливостей Linux систем, який дозволяє сканувати порти на вразливості, використовуючи в якості допоміжної програми сканер Nmap.

Зміст

ВСТУП

Актуальність. багато людей використовують ОС Linux, оскільки вона безкоштовна та на відміну від Windows має відкритий програмний код.Характеризується меншим ступенем вразливості, крім того підтримує практично всі мови програмування.Саме тому – написання скрипту в цій системі є більш поширеною дією.Для забезпечення процесів виявлення вразливостей в цій ОС необхідна наявність програмного або програмно-апаратного забезпечення,що називається сканер.

Метою цієї дипломної роботи є створення програмного модулю моніторингу вразливостей Linux систем.

Для посягнення поставленої мети необхідно розв'язати наступні задачі:

- Проаналізувати архітектуру Linux,вразливості системи та моніторинг.
- Розробити програмний модуль моніторингу вразливостей Linux систем.
- Провести тестування розробленого програмного модулю моніторингу вразливостей Linux систем.

Об’єктом дослідження є процес пошуку вразливостей в Linux системах.

Предметом дослідження методи (або технології) та програмне забезпечення для виявлення вразливостей у linux системах.

Методи дослідження проведені дослідження базуються на використанні мови програмування “Lua” на платформі Nmap Scripting Engine та сучасних методів і засобів виявлення вразливостей Linux систем.

Галузь застосування даний програмний комплекс широко використовується в сфері захисту цифрової інформації. Особливо в кібербезпеці для захисту від атак та виявлення вразливостей системи.

Практична цінність полягає в тому, що в роботі розроблено програмний модуль моніторингу вразливостей Linux систем, який дозволяє сканувати порти на вразливості, використовуючи в якості допоміжної програми сканер Nmap.

РОЗДІЛ 1. ЩО ТАКЕ LINUX СИСТЕМИ

1.1 Архітектура Linux

Linux має довгу історію використання ретельно пророблений багатокористувацької архітектури. За своєю архітектурою Linux є в основному модульної системою. Функціонування Linux не залежить від RPC-механізму, а сервіси зазвичай за умовчанням налаштовані не використовувати RPC-механізм. Сервери Linux ідеально підходять для віддаленого адміністрування.

При подальшому читанні слід пам'ятати про варіації в використовуваних за замовчуванням конфігураціях різних дистрибутивів ОС Linux, тому те, що вірно для Red Hat Linux, може виявитися неправильним для Debian, і ще більше відмінностей може бути в SuSE. Здебільшого в тому, що стосується змін за замовчуванням, всі основні дистрибутиви Linux слідують одним і тим же розумним правилам.

Linux ніколи не була операційної системою. Тому в ній з самого початку закладений принцип ізолювання користувачів від додатків, файлів і каталогів, що впливають на операційну систему в цілому. Кожному користувачеві надається користувальницький каталог, в якому зберігаються всі його файли даних і файли конфігурації, що належать цьому користувачу. Коли користувач запускає який-небудь додаток (наприклад, текстовий процесор), воно запускається з обмеженими повноваженнями даного користувача. Запущений користувачем додаток має право на запис тільки в власний каталог цього користувача. Воно не може нічого записати в системний файл або навіть в каталог іншого користувача, якщо тільки адміністратор явно не надасть даному користувачеві таке право.

Ще важливіше, що Linux надає майже всі функціональні можливості (наприклад, візуалізацію зображень в форматі JPEG) у вигляді модульних бібліотек. Тому, коли текстовий процесор відображає JPEG-зображення, відповідні функції запускаються з тими ж обмеженими повноваженнями, що і сам текстовий процесор. Якщо в програмах візуалізації JPEG-зображень є пролом, зловмисник зможе використовувати її лише для отримання таких же

повноважень, що значно обмежує масштаби можливої шкоди. У цьому перевага модульних систем, і ці системи ближче до ідеалу операційної системи, описаного вище як конструкція з трьох сфер.

Обмеження за замовчуванням - властивість модульної архітектури Linux; чи можливо відправити електронною поштою користувачу Linux таке повідомлення, яке заразить вірусом весь комп'ютер. Не має значення, наскільки невдало сконструйований поштовий клієнт або як саме він прореагує на вірус - повноваження, встановлені для клієнта, дозволять йому заразити або пошкодити тільки файли свого користувача. Web-навігатори, що працюють в ОС Linux, не підтримують такі небезпечні за своєю природою об'єкти, як елементи управління ActiveX, але навіть якщо б вони підтримувалися, шкідливий елемент ActiveX зміг би запуститися тільки з повноваженнями того користувача, який запустив web-навігатор. І в цьому випадку найбільший шкоду, яку він зміг би принести - це заразити або видалити власні файли користувача.

Навіть сервіси, наприклад, web-сервери, зазвичай запускаються як користувачі з обмеженими повноваженнями. Так, Debian GNU / Linux запускає web-сервер Apache як користувача "www-data", що належить до групі з тим же ім'ям "www-data". Якщо зловмисник на комп'ютері з Debian отримає повний контроль над web-сервером Apache, він зможе впливати тільки на файли, що належать користувачеві "www-data", тобто на web-сторінки. У свою чергу, MySQL, сервер бази даних SQL-типу, часто використовуваний разом з Apache, запускається з повноваженнями користувача "mysql". Навіть якщо Apache і MySQL разом обслуговують web-сторінки, зловмисник, який отримав контроль над Apache, не матиме повноважень, що дозволяють використовувати уразливість в Apache для отримання контролю над сервером бази даних, тому що сервер бази даних "належить" іншому користувачеві.

Крім того, користувачі, асоційовані з такими сервісами, як Apache, MySQL і т.д., часто встановлюються з обліковими записами, які не мають доступу до командного рядка. Тому, якщо зловмисник зможе отримати права облікового запису користувача MySQL, він не зможе скористатися цією вразливістю для того, щоб дати довільні команди на сервер Linux, оскільки дана обліковий запис не може викликати команди.

За своєю архітектурою Linux є модульною, а не монолітною системою

Linux - це операційна система, сконструйована, в основному, за модульним принципом, від ядра (центрального "мозку" Linux) до додатків. У Linux практично немає нерозривних зв'язків між будь-якими компонентами. Не існує єдиного процесора web-навігатора, використовуваного довідковими системами або програмами електронної пошти. Справді, неважко настроїти більшість програм електронної пошти так, щоб використовувати вбудований web-навігатор для відображення HTML-повідомлень або запускати будь-який потрібний web-навігатор для перегляду HTML-документів або переходу за посиланнями в тексті повідомлення. Отже, пролом в одному процесорі web-навігатора необов'язково становить небезпеку для будь-яких інших додатків на даному комп'ютері, так як майже ніякі інші додатки, крім самого web-навігатора, що не залежать від єдиного процесора web-навігатора.

Не всі в Linux є модульним. Дві найбільш популярні графічні середовища, KDE та GNOME, в якомусь сенсі монолітні за своєю архітектурою. По крайней мере, монолітні настільки, що в принципі оновлення однієї частини GNOME або KDE може порушити роботу інших частин GNOME або KDE. Але і GNOME, і KDE не до такої міри монолітні, щоб було потрібно використання додатків, розроблених спеціально для GNOME або KDE. Додатки GNOME або будь-які інші додатки можна запускати під KDE, а KDE або будь-які інші програми - під GNOME.

Ядро Linux підтримує модульні драйвери, але в значній мірі є монолітним ядром, тому що сервіси в цьому ядрі взаємозалежні. Всі негативні наслідки монолітності мінімізуються тим, що ядро Linux, наскільки це можливо, розроблено як найменша частина системи. Linux майже фанатично дотримується наступного принципу: "Якщо завдання може бути виконана поза ядра, вона повинна бути виконана поза ядра". Це означає, що в Linux майже кожна корисна функція ("корисна" означає "сприймається кінцевим користувачем") не має доступу до уразливих частин системи Linux.

1.2 Переваги та недліки

Чому я вибрав саме Linux? Нижче я опишу його переваги та недоліки в зрівнянні з операційною системою Windows.

Безкоштовне використання

Одна з найголовніших переваг Linux це те,що він доступний для використання абсолютно безкоштовно. Проте, не у випадку з Windows…

Вам не доведеться платити 100-250 доларів, щоб отримати на руки справжню копію дистрибутива Linux (наприклад, Ubuntu, Fedora). Все абсолютно безкоштовно.

Якщо систему ви використовуєте абсолютно безкоштовно, це ж добре, в нашому випадку ми збережемо певну кількість коштів які ми можемо направити в більш корисному напрямку, наприклад, можна розглянути питання оновлення обладнання вашого комп'ютера.

Відкритий вихідний код

На відміну від Windows, Linux має повністю відкритий вихідний код. Ви можете подивитися вихідний код Linux, що є величезним плюсом.

Я знаю, що більшість користувачів зовсім не звертають увагу на те, чи має система відкритий код або це пропрієтарних ПО із закритим кодом, для мене відкритий код в Linux, це найважливіша особливість при використанні GNU / Linux.

Безпека

Операційна система Windows уразлива для різних типів атак. Поки що Linux не так вразливий, як Windows так як Linux поки не знайшов такого рівня популярності, щоб у кожного другого в якості основної платформи використовувався Linux. Звичайно, не можна сказати з упевненістю, що Linux повністю той ідеал який не схильний до вірусів, але в порівнянні з Windows, набагато безпечніше.

Це саме так, Linux так влаштований, що робить його безпечної операційною системою. В цілому, процес управління пакетами, робота з репозиторіями, і

багато інших функцій дозволяють використовувати більш широкі можливості, що надає Linux більше безпеки, ніж Windows.

Після установки ОС Windows, вам необхідно завантажити / купити програму захисту від вірусів, щоб зберегти ваш комп'ютер в безпеці від хакерів і шкідливих програм. Однак, Linux не вимагає використання антивірусних програм. Звичайно, існують програмні засоби, наприклад фаєрволи, які допоможуть вашій системі бути захищеною від загроз, але більшою мірою в цьому немає необхідності якщо ви використовуєте комп'ютер лише для роботи, ігор, перегляду фільмів і серфінгу по мережі.

Можливість працювати на комп’ютерах зі старим обладнанням

У міру того як розвиваються операційні системи, їх вимоги до обладнання зростає експоненціально. Наприклад, якщо ви придбаєте ліцензійну копію Windows 10, вам доведеться оновити ваше обладнання хоча-б під мінімальні вимоги які вимагає ця ОС, щоб запустити систему успішно, і щоб всі програми запускалися як годиться і не гальмували, оперативної пам'яті не менше 4 ГБ інакше буде жах, а не робота.

Проте, з Linux, ви можете використовувати навіть ваш найстаріший комп'ютер з старим обладнанням і система буде виконувати всі необхідні завдання. Це не означає, що кожен Linux дистрибутив буде працювати з 256 МБ оперативної пам'яті в поєднанні з застарілим процесором. У вас є вибір і ви можете вибрати з безлічі варіантів, ви можете встановити на такому обладнанні систему з мінімальними вимогами для роботи, наприклад Puppy Linux.

В результаті якщо порівнювати ефективність Windows і Linux, в будь-якому випадку виграє Linux так як тут ви знайдете для себе дистрибутив під ваше обладнання будь це старий комп'ютер або новий, ви зможете підібрати такий дистрибутив в якому ви зможете працювати з мінімальними навантаженнями. На відміну від операційної системи Windows, яка має більш високі вимоги до обладнання котрі дають вам вибір і не зважаючи на те, старе у вас обладнання або нове.

В цілому, навіть якщо порівнювати Linux і Windows, в Linux у вас є вибір на відміну від Windows де вам нав'язують лише те, що ви повинні встановити систему і використовувати її на постійній основі. Ті хто хоч раз спробував Linux, вже мають досвід роботи як по одну сторону барикад так і по другу і можуть порівняти, де краще і що краще і чому там потрібно купувати ліцензійне ПЗ, а там немає і ще й вибір величезний, бери то бери інше , що тобі

подобається то і встановлюй. Ось чому більшість серверів по всьому світу вважають за краще працювати на Linux.

Linux - Ідеал для програмування

Linux підтримує практично всі основні мови програмування (Python, C / C ++, Perl, Java , Ruby і т.д.). Крім того, він пропонує широкий спектр програм, що використовуються для програмування.

Термінал Linux перевершує в можливостях командний рядок. Ви можете знайти багато бібліотек, розроблених спочатку для Linux. Крім того, багато програмістів відзначають, що менеджер пакетів в Linux, допомагає їм встановити більшість додатків легко і просто.

Цікаво, що можливості сценаріїв BASH також є однією з найбільш переконливих причин, чому програмісти воліють використовувати ОС Linux. Linux має вбудовану підтримку SSH, який допоможе вам легко управляти вашими серверами.

Оновлення ПЗ

Microsoft пропонує оновлення програмного забезпечення тільки тоді, коли у них збереться велика кількість багів або інших проблем і тільки тоді випускаються великі поновлення, іншими словами кажучи, в Windows не поспішають щодня випускати оновлення якщо потрібно виправити лише один баг в безпеки або ще щось на відміну від Linux. З іншого боку, іноді можна було спостерігати оновлення програмного забезпечення для вирішення дрібних проблем.

У Linux, ви помітите більше оновлень, які в більшій частині пропонують виправити виявлені баги або встановити оновлення безпеки або ж просто оновити ваші додатки до більш нової версії, з усім цим ви можете зіткнутися. Ви не тільки побачите велику кількість оновлень програмного забезпечення, ви також будете спостерігати набагато швидке оновлення і без необхідності перезавантаження комп'ютера на відміну від Windiws.

Налаштування та оформлення

Одна з головних переваг використання Linux в порівнянні з Windows, є тонке налаштування системи під свої потреби. Якщо вам не подобається зовнішній вигляд вашої системи, в Linux ви маєте можливості налаштувати все ідеально як вам потрібно.

Крім установки тем, для Вас доступна маса чудових тем іконок. На додаток до цього, ви можете використовувати Conky для відображення системної інформації на робочому столі в зручному для вас місці розмістивши для цього відповідний віджет. Це лише мала частина того, що ви зможете зробити з вашим робочим столом.

Підтримка

Вам не потрібно наймати експерта, щоб вирішити ваші проблеми, якщо ви зіткнулися з проблемами у використанні Linux. Вам просто потрібно пошукати рішення в мережі Інтернет, якщо в мережі рішення не знайдеться, хоча це вкрай рідко трапляється так як до вас, питання на який ви шукайте відповідь, вже задавали тисячі користувачів. Якщо все ж відповідь ви не знайдете, ви можете запитати у спільноти на форумах де вам допоможуть користувачі які вже стикалися з цією проблемою і ви обов'язково знайдете рішення і за це вам не доведеться платити.

Конфіденційність

Ви напевно вже чули, що Microsoft збирає дані, отримані від кожного користувача, часто це відбувається саме при оновленні системи, ви навіть і не підозрюєте, але Microsoft вже буде знати все що ви робили в мережі і які сайти відвідували з ким ви спілкувалися в соціальних -Мережі або про що говорили з дівчиною в Skype. Багато хто чув про Windows 10 і яка кількість бруду на неї було вилито, як вона збирає дані, що збирає і т.д. і т.п..

Якщо ви коли-небудь використовували Windows 10, ви могли побачити, що в налаштуваннях конфіденційності, все включено за замовчуванням. Навіть якщо ви відмовитеся відправити інформацію про своїх даних, Microsoft як і раніше буде збирати ваші дані і відправляти на сервера Microsoft. Використовуючи Windows, ви не можете бути спокійні при використанні цієї операційної системи так як на кожному етапі тут використовуються шпигунські модулі які не можна відключити.

На противагу цьому, Linux буде ідеальним рішенням і конфіденційна інформація нікуди не буде злита, за це вам турбуватися не потрібно. По-перше, дистрибутиви не збирають ваші дані. Крім того, ви не будете потребувати додаткових інструментах для захисту вашої особистої інформації, а точніше в установці антивіруса, на крайній випадок можна встановити фаєрвол Gufw, який дозволить вам захиститися від можливих мережевих загроз, хоча щоб постраждати від чого або в Linux, доведеться дуже попітніти або ж самому щось зламати як в більшій частині випадків і відбувається.

Стабільність та надійність Linux

Система Windows стає все більш непридатною для використання день за днем, якщо за системою не стежити і не чистити регулярно сміття, через кілька місяців вам обов'язково потрібно буде перевстановлювати заново систему.

Якщо ви використовуєте Linux, вам не доведеться турбуватися про переустановлення, щоб система працювала швидше вам просто так само необхідний хоча-б невеликий догляд за чищенням кеша після установки додатків, очищення сміття після установки додатків, для всього цього є готове безкоштовне ПО хоча можна все виконати використовуючи термінал виконавши певний набір команд, все дуже просто і ваша система буде працювати стабільно на постійній основі. Так, навіть якщо ви працюєте в Linux, потрібно стежити за оновленнями, що ви встановлюєте, не поспішати постійно оновлювати ядро, якщо система працює стабільно, в оновленні до більш нової версії ядра немає необхідності, інша справа якщо це оновлення безпеки, це звичайно ж потрібно встановлювати. На особистому досвіді пройдений етап, сильно часті оновлення ядер до доброго не приведуть так як обов'язково десь ви зустрінете недопрацювання чи ще якийсь баг.

Той факт,що якщо ви використовуєте Windows,будьте готові до постійних перезавантажень системи як з приводом,так і без нього…

Ви тільки що встановили програмне забезпечення - перезавантаження!

Нещодавно видалили програмне забезпечення - перезавантаження!

Ви щойно встановили оновлення Windows - знову перезавантаження!

Якщо система повільно працює - знову перезавантаження…

Проте, у випадку з Linux, вам не доведеться перезавантажити комп'ютер для ситуацій, які згадані вище. Ви можете з комфортом продовжити вашу роботу, і Linux не буде турбувати вас так настирливо як Windows.

Ще один факт, що доводить надійність Linux, це веб-сервери. Ви могли помітити, що більшість інтернет-гігантів, таких як Google і Facebook працюють на Linux. Все суперкомп'ютери працюють на Linux.

Отже, чому не Windows, Linux краще? Linux відбувається тому, що є набагато більш надійним, ніж ОС Windows.

1.3 Вразливості та їх види

Вразливість це певна помилка в програмі, яка допущена її розробником. Цю помилку може використовувати будь-який користувач для того, щоб нашкодити функціонуванню ПО, в тому числі для злочинних цілей. Найбільш небезпечні уразливості це ті, які дають можливість виконувати довільний код.

Всі уразливості можна умовно поділити на локальні і віддалені. Локальні можуть бути, якщо у шахрая є доступ до локального комп'ютера, а віддалені, коли потрібен тільки доступ через Інтернет.

Одним з найбільш поширених методів проникнення в систему - використання вразливостей. Відсутність процедур управління і відстеження вразливостей, не кажучи вже про відсутність збудованих процесів установки виправлень, може привести до того, що системи виявляться беззахисними після виявлення чергової уразливості і публікації експлойта для неї. Часто експлойт публікують вже через кілька годин після її виявлення. Для Linux ця проблема більш критична, оскільки відкритий вихідний код дозволяє швидко знайти проблемну функцію і написати код для експлуатації помилки.

Важливо відзначити, що значна кількість вразливостей в сервісі або платформі не обов'язково означає, що ці уразливості обов'язково несуть істотний ризик.

Кожен виробник дистрибутива Linux виконує свою процедуру обробки вразливостей. У той час як виправлення від вендорів приходять в різний час, заплатки upstream, будь то оригінальний пакет або вихідний код утиліти, з'являються першими. Вендори Linux відповідають за виправлення вразливостей в таких компонентах, як ядро, утиліти і пакети. У 2019 Red Hat виправив понад 1000 CVE в своєму дистрибутиві Red Hat Enterprise Linux (RHEL), згідно з їх звітом Product Security Risk Report. Це більше 70% від загального числа вразливостей, виправлених у всіх продуктах.

Уразливості додатків, що працюють під управлінням Linux, були причиною декількох серйозних інцидентів. Наприклад, гучна витік даних в Equifax сталася в результаті експлуатації уразливості CVE-2017-5638 в Apache Struts. Тоді хакери проникли в корпоративну мережу бюро кредитних історій Equifax 13 травня 2017 року, але підозрілу активність служба безпеки помітила тільки в кінці липня. Кіберзлочинці провели всередині мережі 76 днів, встигнувши за цей час завантажити з 51 бази даних особисту інформацію 148 млн американців - це 56% дорослого населення США. Крім американських громадян в витік потрапили відомості 15 млн клієнтів Equifax в Великобританії і близько 20 тис.

Громадян Канади. Загальні витрати Equifax в результаті цього інциденту за два наступні роки склали понад 1,35 млрд доларів США і включають витрати на зміцнення систем безпеки, підтримку клієнтів, оплату юридичних послуг, а також виплати за судовими позовами.

Що таке вразливості та їх види

Перед тим як перейти до самого списку вразливостей важливо зрозуміти що це таке і якими вони бувають. Як я вже сказав, вразливість - це помилка в програмі, за допомогою якої користувач може використовувати програму так, як це не було заплановано її розробником.

Це може бути відсутність перевірки на правильність отриманих даних, перевірки джерела даних і найцікавіше - розміру даних. Найнебезпечніші уразливості - це ті, які дозволяють виконувати довільний код. В оперативній пам'яті всі дані мають певний розмір і програма розрахована на запис в пам'ять даних від користувача певного розміру. Якщо користувач передасть більше даних, то вона повинна видати помилку.

Але якщо програміст припустився помилки, то дані перезапишуть код програми і процесор буде намагатися їх виконати, таким чином і виникають уразливості переповнення буфера.

Також всі уразливості можна поділити на локальні, які працюють тільки якщо у хакера є доступ до локального комп'ютера і віддалені, коли досить доступу через інтернет. А тепер перейдемо до списку вразливостей.

1. DIRTY COW

Першою в нашому списку буде свіжа уразливість, яка була виявлена цієї осені. Назва Dirty COW розшифровується як Copy on Write. Помилка виникає в файлової системі під час копіювання при записі. Це локальна уразливість, яка дозволяє отримати повний доступ до системи будь-якого звичайним користувачем.

Якщо коротко, то для використання уразливості потрібно два файли, один доступний на запис тільки від імені суперкористувача, другий для нас. Починаємо дуже багато раз записувати дані в наш файл і читати з файлу суперкористувача, через певний час настане момент, коли буфери обох файлів

змішаються і користувач зможе записати дані в файл, запис якого йому недоступна, таким чином можна видати себе права root в системі.

Уразливість була в ядрі близько 10 років, але після виявлення була швидко усунена, хоча залишилися ще мільйони пристроїв Andoid в яких ядро не оновлювалося і не думає і де цю уразливість можна експлуатувати. Уразливість отримала код CVE-2016-5195.

2. Вразливість GLIBC

Уразливість отримала код CVE-2015-7547. Це була одна з найбільш обговорюваних вразливостей серед проектів з відкритим вихідним кодом. У лютому 2016 з'ясувалося, що бібліотека Glibc має дуже серйозну уразливість, яка дозволяє зловмисникові виконати свій код на віддаленій системі.

Важливо зауважити що Glibc - це реалізація стандартної бібліотеки Ci і C ++, яка використовується в більшості програм Linux, в тому числі сервісів і мов програмування таких як PHP, Python, Perl.

Помилка була допущена в коді розбору відповіді DNS сервера. Таким чином, уразливість могли використовувати хакери, до DNS яких зверталися вразливі машини, а також виконують MITM атаку. Але уразливість давала повний контроль над системою

Уразливість була в бібліотеці ще з 2008 року, але після виявлення досить швидко були випущені патчі.

3. HEARTBLEED

У 2014 році була виявлена одна з найсерйозніших за масштабом і наслідками вразливість. Вона була викликана помилкою в модулі heartdead програми OpenSSL, звідси і назва Heartbleed. Уразливість дозволяла зловмисникам отримати прямий доступ до 64 кілобайтам оперативної пам'яті сервера, атаку можна було повторювати, поки вся пам'ять не буде прочитана.

Незважаючи на те, що виправлення було випущено дуже швидко, постраждало дуже багато сайтів і додатків. Фактично уразливими були всі сайти, які використовують HTTPS для захисту трафіку. Зловмисники могли отримати паролі користувачів, їхні особисті дані і все що знаходилося в пам'яті в момент атаки. Уразливість отримала код CVE-2014-0160.

4. STAGEFRIGHT

Якщо вразливість отримала кодове ім'я, це однозначно означає, що вона заслуговує на увагу. Уразливість Stagerfight не виняток. Правда, це не зовсім проблема Linux. Stagefright - це бібліотека для обробки мультимедійних форматів в Android.

Вона реалізована на C ++, а значить обходить всі захисні механізми Java. У 2015 році було виявлено цілу групу вразливостей, які дозволяли виконати віддалено довільний код у системі. Ось вони CVE-2015-1538, CVE-2015-1539, CVE-2015-3824, CVE-2015-3826, CVE-2015-3827, CVE-2015-3828 і CVE-2015-3829.

Зловмиснику було досить відправити MMS на уразливий смартфон із спеціально модифікованим медіафайлів, і він отримував повний контроль над пристроєм з можливістю записувати і читати дані з карти пам'яті. Уразливість була виправлена розробниками Android але до сих пір мільйони пристроїв залишаються уразливими.

5. zero-day ЯДРА

Це локальна уразливість, яка дозволяє підвищити права поточного користувача до root через помилки в системі роботи з криптографічними даними ядра, які зберігаються в пам'яті. Вона була виявлена в лютому 2016 року і охоплювала всі ядра починаючи від 3.8, а це значить що уразливість існувала 4 роки.

Помилка могла використовуватися хакерами або шкідливим програмним забезпеченням для підвищення своїх повноважень в системі, але дуже швидко була виправлена.

6. Вразливість B MYSQL

Ця вразливість отримала код CVE-2016-6662 і зачепила всі доступні версії сервера баз даних MySQL (5.7.15, 5.6.33 і 5.5.52), бази даних Oracle і клони MariaDB і PerconaDB.

Зловмисники могли отримати повний доступ до системи через SQL запит передавався код, який дозволяв замінити my.conf на свою версію і перезавантажити сервер. Також була можливість виконати шкідливий код з правами суперкористувача.

Рішення MariaDB і PerconaDB випустили патчі досить оперативно, Oracle відреагував, але набагато пізніше.

7. SHELLSHOCK

Ця вразливість була виявлена в 2014 році перед тим як проіснувала 22 роки. Їй було присвоєно код CVE-2014-6271 і кодове ім'я Shellshock. Ця вразливість порівнянна за небезпекою з уже відомою нам Heartbleed. Вона викликана помилкою в інтерпретаторі команд Bash, який використовується за умовчанням в більшості дистрибутивів Linux.

Bash дозволяє оголошувати змінні оточення без аутентифікації користувача, а разом в ними можна виконати будь-яку команду. Особливої небезпеки це набирає в CGI скриптах, які підтримуються більшістю сайтів. Уразливі не тільки сервери, але і персональні комп'ютери користувачів, маршрутизатори та інші пристрої. По суті, зловмисник може виконати віддалено будь-яку команду, це повноцінне вилучене керування без аутентифікації.

Уразливості були піддані всі версії Bash, включаючи і 4.3, правда після виявлення проблеми розробники дуже швидко випустили виправлення.

8. QUADROOTER

Це ціла серія вразливостей в Android, яка була виявлена в серпні 2016. Вони отримали коди CVE-2016-5340, CVE-2016-2059, CVE-2016-2504, CVE-2016-2503. Помилку схильні до більш 900 мільйонів Android пристроїв. Всі уразливості були виявлені в драйвері ARM процесора Qualcomm і всі вони можуть використовуватися для отримання root доступу до пристрою.

Як і DirtyCOW тут не потрібно ніяких повноважень, досить встановити шкідливий додаток і воно зможе отримати всі ваші дані і передати їх зловмиснику.

9. Вразливість В OPENJDK

Це дуже серйозна уразливість linux 2016 Java машині OpenJDK з кодом CVE-2016-0636, вона зачіпає всіх користувачів, що працюють з Oracle Java SE 7 Update 97 і 8 Update 73 і 74 для Windows, Solaris, Linux і Mac OS X. Ця вразливість дозволяє зловмисникові виконати довільний код за межами Java

машини, якщо ви відкриєте спеціальну сторінку в браузері з вразливою версією Java.

Це дозволяло зловмисникові отримати доступ до ваших паролів, особистих даних, а також запускати програми на вашому комп'ютері. У всіх версіях Java помилка була дуже оперативно виправлена, вона проіснувала з 2013 року.

10. Вразливість Протоколу HTTP / 2

Це ціла серія вразливостей, яка була виявлена в 2016 році в протоколі HTTP / 2. Вони отримали коди CVE-2015-8659, CVE-2016-0150, CVE-2016-1546, CVE-2016-2525, CVE-2016-1544. Вразливостям були піддані всі реалізації цього протоколу в Apache, Nginx Microsoft, Jetty і nghttp2.

Всі вони дозволяють зловмисникові дуже сильно уповільнити роботу веб-сервера і виконати атаку відмова від обслуговування. Наприклад, одна з помилок приводила до можливості відправки невеликого повідомлення, яке на сервері розпаковувати в гігабайти. Помилка була дуже швидко виправлена і тому не викликала багато шуму в співтоваристві.

11. MELTDOWN I SPECTRE

Це одні з найгучніших вразливостей кінця 2017 початку 2018 року. Уразливості отримали такі назви: CVE-2017-5754, CVE-2017-5753 і CVE-2017-5715. Їм схильні не тільки Linux, але і Windows системи, тому що їх причина - помилка в алгоритмах роботи процесора. Процесор намагається вгадати яку інструкцію треба буде виконати далі, виконує її і поміщає значення в кеш. І до цього кешу виявляється можна отримати доступ в певних обставинах. А це надає зловмисникові необмежений доступ до будь-яких даних у вашій системі. Спільнота розробників операційних систем оперативно організувалося і дуже швидко був випущений патч, які вирішував якщо не все, то більшість проблем.

12. ВРАЗЛИВОСТІ В EXIM

У 2019 багато шуму виникло навколо вразливостей поштового сервера Exim. Ці уразливості отримали назву CVE-2019-15846 і CVE-2019-10149. Обидві уразливості дозволяли зловмисникам при певних умовах виконувати довільний код на сервері від імені root. На момент виникнення вразливостей поштовий сервер Exim використовувався на 50% всіх вузлів з поштовими серверами.

Уразливість була швидко усунена, але через її масштабність було дуже багато розмов.

Розглянувши найнебезпечніші уразливості Linux останнього десятиліття. Більшість з них могли завдати серйозної шкоди системам якби не були вчасно виправлені. Завдяки відкритому вихідному коду такі уразливості Linux ефективно виявляється і швидко виправляються.

1.4 Найгірші вразливості за останні роки

За рівнем їх вразливості:

1 CVE-2017-18017

CVSS v2: 10 високий

Ця підозріла вразливість очолила наш список CVE ядра Linux за 2018 рік, незважаючи на те, що в її ідентифікатор зазначений 2017 рік. Це пов'язано з тим, що про нього вперше було повідомлено, і його ідентифікатор був зарезервований у 2017 році до того, як він був опублікований в Національній базі даних вразливостей в січні 2018 року.

Згідно з його опису, функція tcpmss_mangle_packet в net / netfilter / xt_TCPMSS.c може дозволити віддаленим хакерам виконати атаку відмови в обслуговуванні (використання після звільнення і пошкодження пам'яті. Звіти показують, що зловмисники можуть використовувати присутність xt_TCPMSS в iptables для виконання невизначеного діапазону інших впливів на ваше програмне забезпечення.

Ця конкретна уразливість ядра Linux являє собою справжній удар по зубах, з огляду на важливу роль, яку вона відіграє в фільтрації мережевого обміну даними, визначаючи максимальний розмір сегмента, який дозволений для прийому заголовків TCP. Без цих важливих елементів управління користувачі можуть зіткнутися з проблемами переповнення.

Загальна тема, яку ми бачимо в усіх вразливості ядра Linux в цьому списку, полягає в тому, що атаки можуть виконуватися віддалено, без будь-яких дій з боку мети. Ці віддалені атаки становлять велику загрозу, ніж, скажімо, та, яка вимагає від хакера роботи локально.

Нарешті, як зазвичай, добрі люди зі спільноти Linux допомогли нам зібрати виправлення, яке допоможе нам зберегти безпеку нашого продукту.

2 CVE-2015-8812

CVSS v2: 10 високий

Серйозний недолік був виявлений в драйверах / infiniband / hw / cxgb3 / iwch_cm.c ядра Linux, коли було виявлено, що він неправильно визначає умови помилки. Результатом цієї уразливості було те, що вона дозволяла віддаленим зловмисникам виконувати довільний код або викликати відмову в обслуговуванні (використання після звільнення) за допомогою створених пакетів.

3 CVE-2016-10229

CVSS v2: 10 високий

Коротко і по суті з цієї вразливістю Linux, де udp.c дозволяє віддаленим зловмисникам виконувати довільний код через трафік UDP, який запускає небезпечну другу контрольну суму під час виконання системного виклику recv з прапором MSG_PEEK.

Уразливість була виявлена внутрішньої командою Google, яка використовувала компонент для мобільної ОС Android. Це проявилося, коли користувачі надали буфер менше, ніж корисне навантаження skb.

4 CVE-2014-2523

CVSS v2: 10 високий

Інша серйозна уразливість виникла в зв'язку з netfilter в ядрі Linux, на цей раз через неправильне використання покажчика заголовка DCCP. Цей недолік дозволяє віддаленим зловмисникам викликати відмову в обслуговуванні (збій системи) або, можливо, виконати довільний код через пакет DCCP, який запускає виклик функції dccp_new, dccp_packet або dccp_error.

5 CVE-2016-10150

CVSS v2: 10 високий

Ця вразливість ядра Linux, пов'язана з використанням після звільнення, була виявлена в функції kvm_ioctl_create_device virt / kvm / kvm_main.c. Це дозволяє користувачам ОС викликати атаку відмови в обслуговуванні.
Можливо, навіть гірше, якщо хакери можуть використовувати цю вразливість для отримання привілеїв за допомогою створених викликів ioctl на пристрої teh / devkvm.
6 CVE-2010-2521
CVSS v2: 10 високий
Множинні переповнення буфера в fs / nfsd / nfs4xdr.c в реалізації XDR на сервері NFS в ядрі Linux дозволяють віддаленим зловмисникам викликати відмову в обслуговуванні. Також існує ймовірність того, що зловмисники можуть виконати довільний код через створений складовою запит WRITE NFSv4, пов'язаний з функціями read_buf і nfsd4_decode_compo

1.5 Фактори ризику

Існують три дуже важливі чинники ризику, які істотно залежать один від одного. Комбінація цих трьох факторів вирішальним чином впливає на загальну серйозність тієї чи іншої проломи в системі захисту. Такими елементами показника загальної серйозності явля-ються: можливий збиток, можливість використання і можлива доступність.

Елементи показника загальної серйозності

Можливі збитки будь-якої виявленої уразливості показує, якої шкоди може бути завдано в результаті її використання злоумишлен-ником. Уразливість, що дозволяє розкрити всі паролі адміністраторів, має високий показник можливого збитку. Цей показник для проломи, в результаті використання якої починає мерехтіти екран, зазвичай набагато нижче, він зростає тільки в тому випадку, якщо це конкретне пошкодження важко виправити.

Можливість використання показує, наскільки просто або складно використовувати дану уразливість, чи буде потрібно висока кваліфікація в програмуванні або цю уразливість зможе використовувати будь-яка людина з самими елементарними знаннями в зазначеній галузі.

Можлива доступність показує, який рівень доступу необхідний для використання даної уразливості. Якщо будь-який початківець хакер (той, кого зазвичай називають "script kiddies") з Інтернету може використовувати пролом на сервері, захищеному фаєрволом, то цей пролом має дуже високу можливу доступність. Якщо ж використовувати пролом зможе тільки співробітник компанії, що має діючі реєстраційне ім'я та пароль, причому виключно з комп'ютера, що знаходиться в будівлі компанії, то можлива доступність цього пролому значно менше.

Показник загальної серйозності і взаємозв'язок між трьома факторами ризику

Один або декілька з перерахованих вище факторів ризику можуть мати вирішальний вплив на загальну серйозність помилки. Розглянемо наступну ситуацію. Керівник інформаційної служби компанії, що займається електронною комерцією через web-сайт, дізнається від аналітика з безпеки, що виявлена пролом в операційній системі, під управлінням якої функціонують

сервери компанії. Зловмисник може використовувати цей пролом, щоб видалити всю інформацію з дисків на всіх серверах, які використовуються компанією.

Можливі збитки - наслідки від цього пролому катастрофічні.

Гірше того, аналітик додає, що з технічної точки зору цей пролом використовувати тривіально просто. Можливість використання має критичний рівень.

Час натискати аварійну кнопку, чи не так?

Але припустимо, що аналітик додає ще кілька слів. Використовувати цей пролом може тільки той, у кого є ключ від серверної кімнати, тому що ця дана уразливість вимагає фізичного доступу до комп'ютерів. Цей єдиний ключовий показник, вибачте за каламбур, радикально змінює загальну серйозність загрози, породженої даної конкретної проломом. Вкрай низька можлива доступність переводить стрілку на шкалі серйозності з "тривога!" на "під контролем".

Навпаки, інша уразливість може бути доступна будь-якому починаючому хакеру в Інтернеті, але як і раніше буде мати незначну серйозність, якщо можливі збитки від цього пролому є несуттєвим.

Можливо, тепер більш зрозуміло, чому оманлива, якщо не зовсім безвідповідальна, методика оцінки безпеки по одному-єдиному показнику, наприклад, за кількістю попереджень про уразливість. У самому крайньому випадку слід розглянути також зазначені три чинники ризику. Яка операційна система заслуговує більше довіри - з сотнею проломів нехтує малою серйозності або ж з десятком проломів надзвичайно високою серйозності (використання проломи веде до катастрофи)? Якщо при оцінці не враховувати загальну серйозність проломів, то підрахунок їх кількості в кращому випадку не має ніякого значення, в гіршому - вводить в оману.

Виняток із правила

Показник загальної серйозності має три згаданих "головних" компонента. Вище ж було показано, як низький можливий збиток або низька можлива доступність можуть практично звести нанівець інші фактори ризику, якими б високими вони не були. Можливість використання - виключення з цього правила.

Пролом, для використання якої потрібно висококваліфікований фахівець, значно менше компенсує високий показник можливого збитку або можливої доступності.

Пояснюється це просто. Якщо для використання проломи необхідно потрапити в кімнату з комп'ютерами, то справа не тільки в тому, що це важко, але і в тому, що будь-яка спроба проникнути в цю кімнату збільшує для зловмисника ризик бути спійманим. І саме тому пролом, яку може використовувати тільки співробітник компанії, менш серйозна, ніж пролом, яку може використовувати будь-який початківець хакер з Інтернету. У першому випадку ризик бути спійманим набагато вище, ніж у другому.

З іншого боку, анонімні зловмисники-програмісти дуже звичайною кваліфікації можуть протягом декількох тижнів або місяців розробляти програму, що дозволяє використовувати будь-яку пролом у захисті, практично не ризикуючи при цьому бути спійманими. Єдина важлива задача, що стоїть перед таким зловмисником, - активувати шкідливу програму так, щоб неможливо було відстежити її автора.

Вже поверхневе знайомство з сучасним станом шкідливого програмного забезпечення показує самоочевидність цього винятку. Мало хто скористається базуки, щоб прокласти собі шлях в машинний зал і "зламати" знаходяться там сервери. Але існує безліч троянських програм, мережевих черв'яків і вірусів, які як і раніше заражають безліч комп'ютерів, і одна з причин в тому, що програмісти, талановиті і не дуже, вважають ознакою професіоналізму вміння подолати технічні труднощі написання шкідливого коду або переробки шкідливого коду, написаного іншими. Очевидно, що технічні труднощі необов'язково компенсують високу небезпеку проломи, викликану іншими причинами.

Застосування показника загальної серйозності

Тільки оцінивши загальну серйозність конкретної проломи, можна перейти до осмислення таких показників, як "скільки попереджень про уразливість існує для Windows у порівнянні з Linux" або "скільки часу проходить між виявленням і виправленням помилки в разі Windows і в разі Linux".

Припустимо, для однієї операційної системи зареєстровано набагато більше попереджень про уразливість, ніж для іншої. Цей показник має сенс тільки в

одному випадку - якщо для даної системи є також більше попереджень про уразливість з високим рівнем загальної серйозності. Одна справа, якщо досить часто трапляються різні дрібні неприємності, які не становлять ніякої небезпеки, і зовсім інша - коли регулярно виявляються нехай і нечисленні, але такі проломи, які ставлять під удар всю компанію.

Припустимо, що для деякої операційної системи зафіксовано більш короткий час між виявленням проломи і виходом відповідної програмної корекції. І в цьому випадку показник має сенс тільки тоді, коли цей час відноситься до Брешії з високою загальною серйозністю. Одна справа - кілька місяців чекати корекції для проломи, використання якої може завдати незначної шкоди невеликій кількості комп'ютерів або навіть зовсім не заподіяти ніякої шкоди. І зовсім інша справа - чекати місяцями корекції для проломи, використання якої може поставити під удар всю компанію.

Можливість використання

Цей показник враховує технічні труднощі, які необхідно подолати, щоб використовувати пролом у захисті. Зазвичай ці труднощі потрапляють в одну з перерахованих нижче категорій. Категорії перераховані в порядку зростання їх серйозності (на практиці реальний порядок цих категорій може змінюватися, але наведений список може виявитися зручним орієнтиром).

Пролом існує, але ще не виявлений. Для використання цього пролому необхідні або виняткова компетентність, або щасливий випадок. Для використання проломи необхідні висока кваліфікація в програмуванні і глибоке знання операційної системи, але про існування цього пролому відомо недостатньо широко, через що малоймовірно, що її використовують багато порушники. Про існування проломи відомо, і для її використання необхідні висока кваліфікація в програмуванні і глибоке розуміння того, як функціонують яких атакували програмне забезпечення та операційна система. Для використання проломи необхідна висока кваліфікація в програмуванні, але вже створений вірус, троянська програма або мережевий черв'як, які можуть служити основою для атаки. Програмісту потрібно тільки модифікувати цей код, щоб використовувати нову пролом або зробити даний вірус більш небезпечним. Для створення коду, що використовує пролом, необхідна висока кваліфікація в програмуванні, але відповідний код вже існує, і досить середньої кваліфікації в програмуванні, щоб удосконалити або модифікувати цей код так, що він буде

використовувати існуючу прогалину або "майбутні" проломи. Для використання проломи досить середньої або початкової кваліфікації в програмуванні або ж елементарного знання комп'ютера. Не має значення, наскільки важко використовувати пролом, оскільки вся робота по створенню засобів для використання цього пролому вже пророблена, а шкідливий код зроблений загальнодоступним для його застосування новачками. Хто завгодно може використовувати пролом, ввівши простий текст в командному рядку або вказавши URL в web-навігаторі.

Можливі збитки

Оцінити цей показник найважче. Потрібно визначити, принаймні, два різних набору категорій. По-перше, треба врахувати, якої шкоди завдасть використання проломи додатків або комп'ютерів. По-друге, необхідно оцінити можливі збитки з точки зору наслідків для всієї компанії. Наприклад, в ситуації, коли пролом дозволяє порушнику прочитати неопубліковані web-сторінки. Шкода від цього незначний, якщо на комп'ютері не зберігається конфіденційна інформація. Але якщо неопублікована web-сторінка містить будь-яку конфіденційну інформацію, наприклад, номери кредитних карт, то загальний можливий збиток може бути дуже великим, при тому що можливий технічний збиток мінімальний. Нижче наводяться (у порядку зростання серйозності) найбільш важливі фактори, які слід враховувати при оцінці можливого технічного збитку в результаті використання будь-якої конкретної проломи.

Пролом впливає тільки на продуктивність іншого комп'ютера, але не настільки, щоб він перестав відповідати на запити. Пролом впливає тільки на власні програми або файли порушника, але не зачіпає файли або програми інших користувачів. Пролом робить незахищеною інформацію в файлах інших користувачів, але не обліковий запис адміністратора або системні файли. Пролом дозволяє порушнику переглядати, змінювати або видаляти призначені для користувача файли. Вона не дає можливості переглядати, змінювати або видаляти файли адміністратора або системні файли. Пролом дозволяє порушнику переглядати критичну інформацію або шляхом дослідження мережевого трафіку, або шляхом доступу з правами "тільки читання" до файлів адміністратора або до системних файлів. Пролом дозволяє порушнику отримати деякі, але не всі, повноваження адміністративного рівня, можливо, в обмеженому оточенні. Пролом дозволяє порушнику викликати відмову системи

або якимось іншим шляхом змусити її не відповідати на звичайні запити. Це - типова атака "відмова в обслуговуванні".

РОЗДІЛ 2. МЕТОДИ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ LINUX СИСТЕМ

2.1 Моніторинг Linux систем

У діяльності будь-якої сучасної компанії вкрай важливу роль грає IT-інфраструктура, яка може включати Wi-Fi роутери, принтери, ноутбуки, файлові сервери, мережеве обладнання, сервери і багато іншого, встановлені або в офісі самої компанії, або в центрі обробки даних. Таким чином, безперервність роботи тієї чи іншої організації в значній мірі залежить від стабільності власної інформаційної мережі та інфраструктури. Саме тут на допомогу приходить системне та мережеве адміністрування, в тому числі, моніторинг системних ресурсів.

Навіщо потрібен моніторинг системи?

Існує багато причин, чому необхідно виконувати моніторинг. До них відносяться:

-Доступ до стану всієї інфраструктури в режимі онлайн.

-Аналіз довгострокових тенденцій.

-Порівняння або проведення експериментів (наприклад, яка підсистема зберігання краще для моєї бази даних - MyISAM або InnoDB?)?

-Оповіщення (повідомлення про перевищення встановлених лімітів).

-Побудова інформаційних панелей (графів, карт стану мережі і т.д.): візуалізація даних (пов'язано з пунктом 2). Допомагає побачити тенденції, неочевидні в їх вихідному необробленому вигляді.

ретроспективний аналіз: взаємозв'язок між отриманими оповіщенням і іншими даними, зібраними за той же проміжок часу.

-Скорочення витрат. Наприклад, один з серверів, розгорнутий в хмарі, використовується не в повну силу. Можна перемістити додаток, встановлений на ньому, на сервер меншого розміру або продовжувати встановлювати на нього нові додатки, не створюючи для них новий сервер

-Перетворення зібраних даних в бізнес-аналітику.

-Відстеження проблем безпеки.

Створення SBOM(software bill of materials)

Інструмент аналізу складає програмне забезпечення (SCA-Software composition Analysis), що створює специфікацію програмного забезпечення (SBOM), що представляє собою список програмних компонентів, встановлених на вашому продукті, а потім створює звіт про уявлення, порівняння компонентів із списком відомих уявлень. Існує кілька різних способів створення SBOM, як описано нижче:

Система побудови

Кращим варіантом для створення SBOM є отримання інформації про компонентах з системи збирання. Системи збирання містять всю інформацію, необхідну для створення SBOM, а також додаткові метадані, які можуть допомогти підвищити точність звітів про уразливість. Метадані складаються з:

Конфігурації, включені в програмному забезпеченні (наприклад, драйвери, включені в ядрі Linux)

Список вже усунених вразливостей на основі застосованих патчів

Інформація про апаратній платформі

Ці додаткові метадані можуть використовуватися інструментом моніторингу вразливостей, щоб зменшити кількість помилкових спрацьовувань, повідомляючи про «незафіксованих» вразливості, які можна застосувати до «вашій апаратній платформі», на основі «включених конфігурацій». У таких компонентах, як ядро Linux, це може знизити кількість вразливостей до 75%, що заощадить вам час на дослідження і виправлення.

Дві популярні системи збирання, підтримувані постачальниками напівпровідників, - це Yocto і Buildroot, тому ідеальний інструмент повинен інтегруватися з цими системами збірки. За допомогою Vigiles ми створили програмне забезпечення з відкритим вихідним кодом для створення SBOM з даними про уразливість, які безпосередньо інтегруються з Yocto (meta-timesys) і Buildroot (vigiles-buildroot) для задоволення потреб вбудованих систем.

Двійкові сканери: генеруйте SBOM, аналізуючи завантажені цільові двовимірні зображення на основі сигналу файлів. Це працює досить добре, коли SBOM і / або джерела збірників недоступні. Однак нижче перераховані недоліки такого підходу:

Ложні срабатывания: метаданні (включені конфігурації, прикладні виправлення та т. Д.), Доступні в системі зборів, які не є частиною двоїчного обробки. Тому, якщо вищестояща версія Yocto або Buildroot поправила уявлення, бінарний сканер показує її як «незафіксовану». Точно так же, якщо уявляти затвора лише для драйвера або платформи, не підходить для вашого продукту, про них все рівно буде повідомлено.

Внутрішні уявні уявлення: іногда SBOM не може бути точно генерованим, оскільки компонент і / або версія не може бути визначена на основі подвійної підписи (наприклад, більш нова або стара частина програмного забезпечення, яка не каталогізується в базі даних сканера).

Приклад інструмента: Зміст Blackduck

Сканери вихідного коду: генеруйте SBOM, аналізуйте вихідний код та цей сканер зазвичай інтегрується з програмним забезпеченням для контролю версії або менеджерів пакетів програм (наприклад, npm, nuget, python pip, godep і т. Д.). Це досить добре працює для підготовки програмного забезпечення, яке включає в себе кілька компонентів з відкритим вихідним кодом, але розвалюється, коли справа приходить до роботи із системами збірників, такими як Yocto та Buildroot, за наступною причиною:

Целеве пристрій SBOM не може бути легко сгенерировано. Щоб створити власний дистрибутив Linux (ОС), системи, що збираються, використовують тисячі пакетів з відкритим вихідним кодом. Некоторые пакети необхідних хост-машин для створення цепочок інструментів, в той час, як кілька інших залежать від часу збірки, і лише частина пакетів у конечному ітоге встановлюється на

цільовому пристрої. Сканери вихідного коду не можуть ідентифікувати встановлені пакети та в конечному ітозі повідомляють усіх програмним забезпеченням як частини SBOM. Замовити вручну, які джерела слід відслідковувати, утомительно та непрактично.

Якщо вам вдалося сгенерувати SBOM, підхід сканера вихідного коду по-прежнему страждає від ложних срабатываний, оскільки він не може збирати артефакти збірки, такі як виправлення, застосовувані систематичні збірки, конфігурація ядра та т. Д.

Інші методи:

Мережеві сканери, такі як Tenable Nessus, Rapid7 InsightVM, nmap і т. Д., Можуть визначити лише запущене програмне забезпечення на основі відкритих портів на пристрої та не може генерувати повний звіт про SBOM та уявлення.

Запрос живого цільового пристрою: якщо пристрій використовує диспетчер пакетів, його можна використовувати для генерації SBOM. Зазвичай це відбувається при використанні стандартних дистрибутивів Linux, таких як RedHat, Debian, Ubuntu та їх відповідні засоби, що відповідають за безпеку, можуть використовуватися для відстеження уявних користувачів. Це виходить за рамки повідомлення обговорення.

Моніторинг

Наступним кроком після створення SBOM є створення списку вразливостей, що впливають на компоненти. Більшість інструментів використовують Національну базу даних вразливостей (NVD), підтримувану Національним інститутом стандартів і технологій (NIST), для повідомлення про уразливість, які в базі даних називаються Common Vulnerabilities and Exposures (CVE). У цьому розділі ми зануримося в потребу в більш точних даних, ніж ті, які повідомляються в NVD.

Дані про уразливість

Як правило, звіти про вразливості працюють таким чином: дослідник безпеки знаходить вразливість, резервує CVE-ID і виконує відповідальне розкриття інформації, дозволяючи фахівцеві із супроводу вразливої продукту досліджувати і виправляти вразливість, якщо це можливо. Потім вразливість з'являється в NVD, і фахівці з обслуговування NVD аналізують уразливість і

додають метадані, такі як Common Platform Enumeration (CPE), що дозволяє автоматичним інструментам аналізувати, яка CVE впливає на які компоненти і версії програмного забезпечення стандартизованим способом. Хоча це звучить чудово, фактична реалізація має кілька недоліків, як показано нижче.

Пропущені CVE і помилкові спрацьовування: згідно з нашим аналізом, по крайней мере, 40% записів CVE NVD мають невірні дані CPE, що призводить до пропущених CVE або помилкових спрацьовувань. Пропущена CVE ставить під загрозу безпеку пристрою, тому що про схильності пристрої CVE практично не повідомляється. Хибнопозитивний результат призводить до втрати часу на аналіз, оскільки CVE, мабуть, впливає на пристрій, що призводить до того, що фахівці з обслуговування пристрою витрачають час на дослідження тільки для того, щоб виявити, що це не так. Причини неточностей наступні:

Неузгоджене іменування (наприклад, один і той же пакет названий по-різному для різних CVE)

Помилки (наприклад, помилки при введенні імені пакета або вразливою версії)

Неправильний / неповний аналіз (наприклад, перераховані не всі порушені версії програмного забезпечення і т. Д.)

Застаріла інформація (наприклад, коли патч переноситься назад, щоб сказати LTS-версії ядра, порушене діапазон пакету для цього CVE не оновлюється)

Немає інформації про версії (наприклад, вводиться тільки ім'я пакета без інформації про версії)

Затримки в звітності: це тимчасова затримка з моменту, коли уразливість стає загальнодоступною (новини, список розсилки і т. Д.), До того моменту, коли NVD проаналізує CVE для вашого інструменту, щоб фактично повідомити про уразливість в вашому продукті.

Легкість моніторингу

Сповіщення: можливість запускати оповіщення про нові вразливості, які впливають на вашу SBOM.

Порівняння: виділення того, що змінилося між скануваннями або маніфестами SBOM. Таким чином, ви можете зосередитися на змінах між скануваннями або збірками.

Розширений пошук: можливість пошуку CVE по імені і версії пакету або за ідентифікатором для отримання докладної інформації про уразливість, створеної Timesys.

Моніторинг вразливостей для вбудованих систем має унікальні потреби, які не вирішуються традиційними інструментами SCA IT або додатків. Використовуючи найкращий інструмент, придатний для роботи, можна заощадити до 75% часу розробника на розслідування помилкових спрацьовувань і виконання завдань по забезпеченню безпеки, одночасно підвищуючи безпеку продукту. Навіть якщо у вашій організації вже є універсальний інструмент для моніторингу вразливостей, спеціальний додатковий інструмент для розробників вбудованого Linux забезпечує значну економію витрат на обслуговування через унікальних вимог до управління уразливими вбудованого Linux.

1.Osquery. Osquery - це інструментальне середовище операційної системи, яка представляє операційну систему як високопродуктивну реляційну базу даних. Він відстежує події в режимі реального часу, щоб ви знали все, що відбувається на сервері, що корисно як частина процесу реагування на інциденти. Він дозволяє вам читати дані з ОС, як якщо б вони були звичайними таблицями, що містять такі речі, як запущені процеси, користувачі, що увійшли в систему, що встановлюються мережеві з'єднання і багато, багато іншого.

2.OSSEC і Wazuh. Протягом багатьох років служби безпеки використовували OSSEC для відстеження систем Linux на предмет потенційних вторгнень; Wazuh - це сучасна альтернатива OSSEC. Хоча можливості розподілених запитів і запитів в реальному часі роблять osquery ідеальним для пошуку загроз, обидва ці інструменту гарні для аналізу файлів журналів, виконуючи моніторинг цілісності файлів (FIM).

3.Аудітд. Auditd, призначений для користувача компонент системи аудиту Linux, є корисною функцією Linux, яку ви можете використовувати для відстеження подій, пов'язаних з безпекою, від процесів до аутентифікації. Він

забезпечує набагато більш детальний спосіб реєстрації активності в системах, ніж той, який зазвичай доступний в системах Linux без нього.

2.2 Інструменти для моніторингу вразливостей Linux систем

Nessus - найвідоміший і ефективний багатоплатформовий сканер вразливостей. Він має графічний інтерфейс користувача і сумісний практично з усіма операційними системами, включаючи Windows, MAC і Unix-подібні операційні системи. Спочатку це був безкоштовний продукт з відкритим вихідним кодом, але потім, в 2005 році, він був закритий і видалений з проектів з відкритим вихідним кодом. Тепер його професійна версія коштує близько 2190 доларів в рік, згідно з їх веб-сайту, що як і раніше набагато дешевше в порівнянні з продуктами конкурентів. Також доступна обмежена безкоштовна версія Nessus Home, але ця версія не має всі її функціями і може використовуватися тільки в домашніх мережах.

1.Запуск Nessus див.рис. 2.1

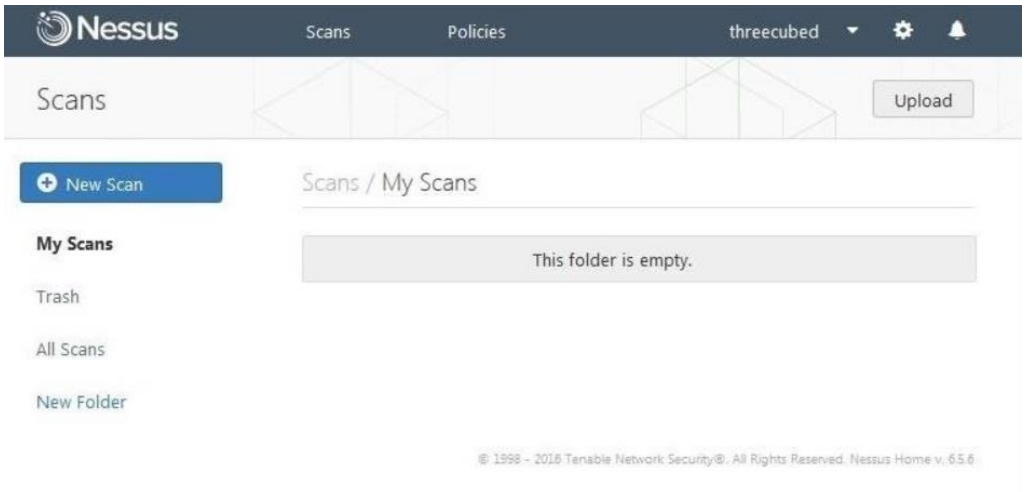


Рис. 2.1 Головна сторінка

2.Це відкриє нову сторінку, на якій ви зможете вибрати тип сканування

Див. Рис. 2.2. доречі, там завжди міститися найактуальніші на сьогоднішній день моделі загроз.

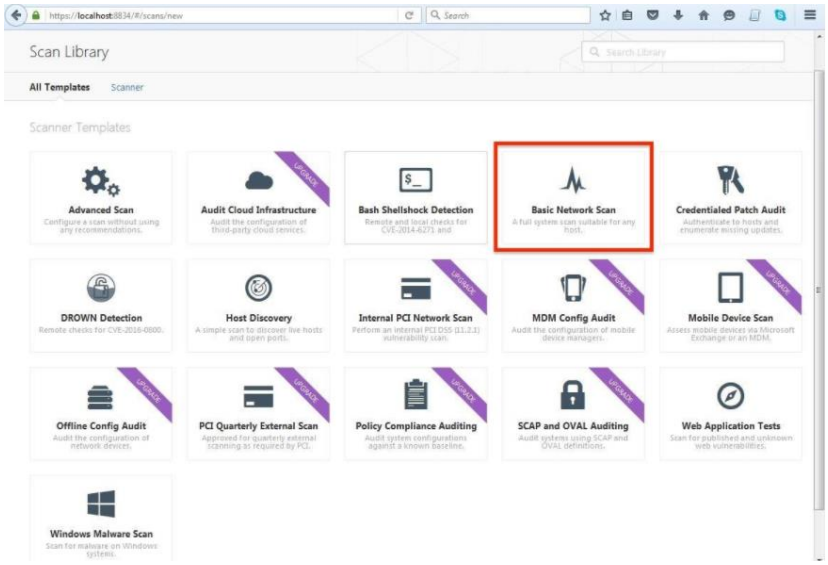


Рис. 2.2 Вибір типу сканування

3.Вибираємо,що ми будемо сканувати Див.Рис. 2.3

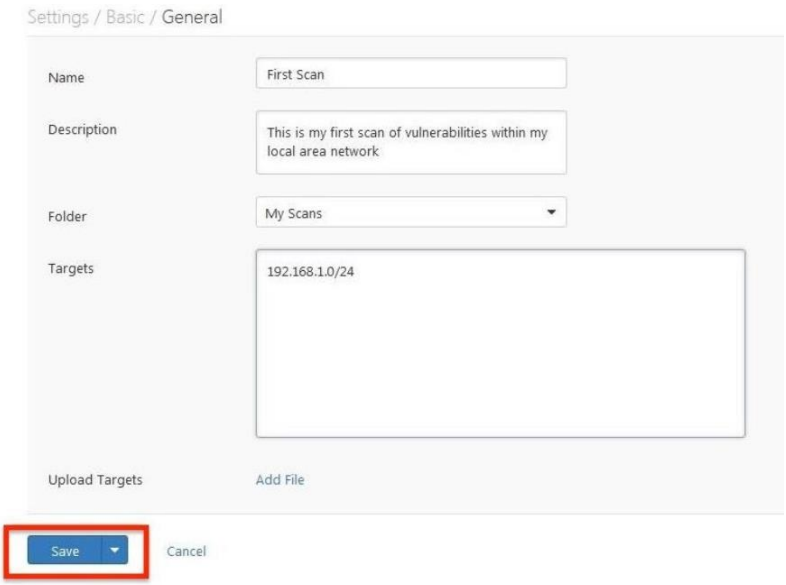


Рис. 2.3 початок сканування

4.За результатами сканування ми отримуємо список з IP адресами та пов'язані з ними ризики Див.Рис. 2.4

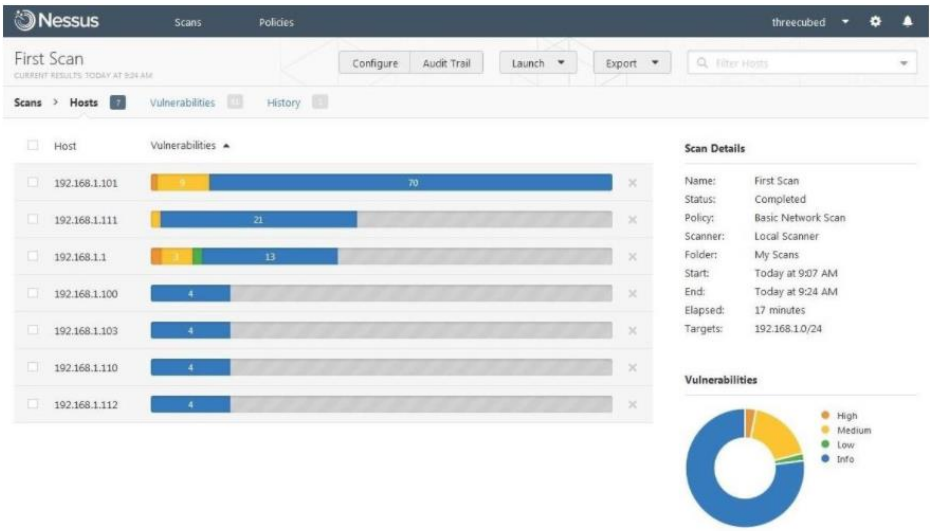


Рис. 2.4 результат сканування

5.Натискаємо "vulnerabilities" в верхньому меню, щоб відобразити всі уразливості, виявлені в мережі Див.Рис.2.5

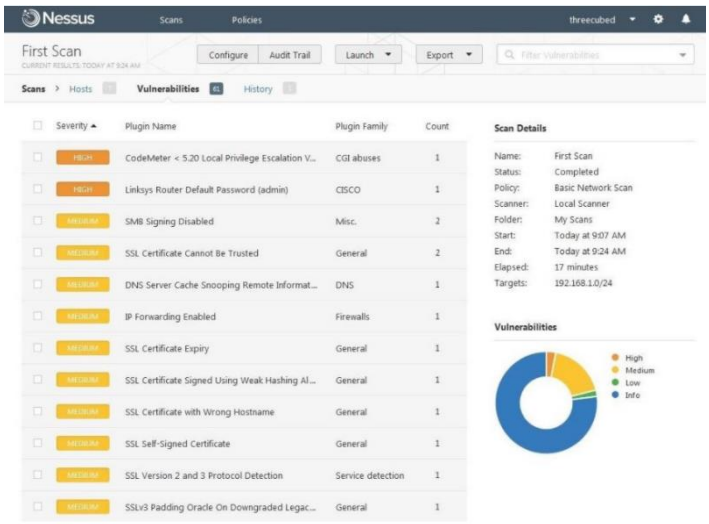


Рис. 2.5 самі вразливості

Nessus **V**ulnerability **S**canner навіть у безкоштовній версії є досить простий у використанні, але в той же час потужний сканер вразливостей. Головною його перевагою є те, що в ньому завжди можна знайти актуальні моделі загроз, на можливість експлуатації яких він швидко і якісно перевірить вашу мережу.

Nmap - це найбільш гнучкий і всеосяжний інструмент з відкритим вихідним кодом, який використовується для виявлення мережі та сканування безпеки. Він може робити все, від сканування портів до зняття відбитків пальців. Операційні системи і сканування вразливостей. Nmap має інтерфейси CLI і GUI, графічний інтерфейс користувача називається Zenmap. Він має власний механізм сценаріїв і поставляється з попередньо написаними сценаріями .nse, використовуваними для сканування вразливостей. У нього є безліч різних опцій для швидкого і ефективного сканування.

Nmap - вільна утиліта, призначена для різноманітного налаштвуання сканування IP-мереж з будь-якою кількістю об'єктів, визначення стану об'єктів сканируемой мережі (портів і відповідних їм служб). Спочатку програма була реалізована для систем UNIX, але зараз доступні версії для безлічі операційних систем.

OpenVAS - це розгалужена версія останнього безкоштовного Nessus на github після його закриття в 2005 році. Для своїх плагінів він як і раніше використовує ту саму мову NASL, що і Nessus. Це безкоштовний потужний сканер мережевих вразливостей з відкритим вихідним кодом.

В якості первісних аутентифікаційних даних використовується логін - admin і пароль, який видається в результаті роботи команди openvas-setup Див.Рис. 2.6

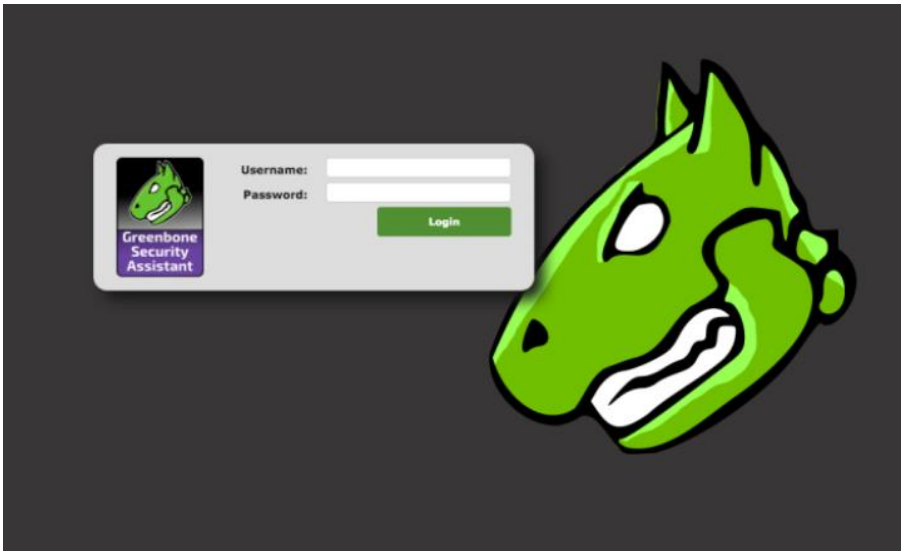


Рис. 2.6 вхід в OpenVAS

В якості первісних аутентифікаційних даних використовується логін - admin і пароль, який видається в результаті роботи команди openvas-setup.

Після того, як аутентифікація пройшла успішна, створимо зразкову завдання по скануванню нашої мережі. Для цього переходимо в головному верхньому горизонтальному меню в Scans -> Tasks. Далі натиснувши в лівому верхньому кутку на іконку з чарівною паличкою, вибираємо у випадаючому меню Advanced Task Wizard. Приклад заповнення віконця даного візарду представлений нижче Див.Рис. 2.7

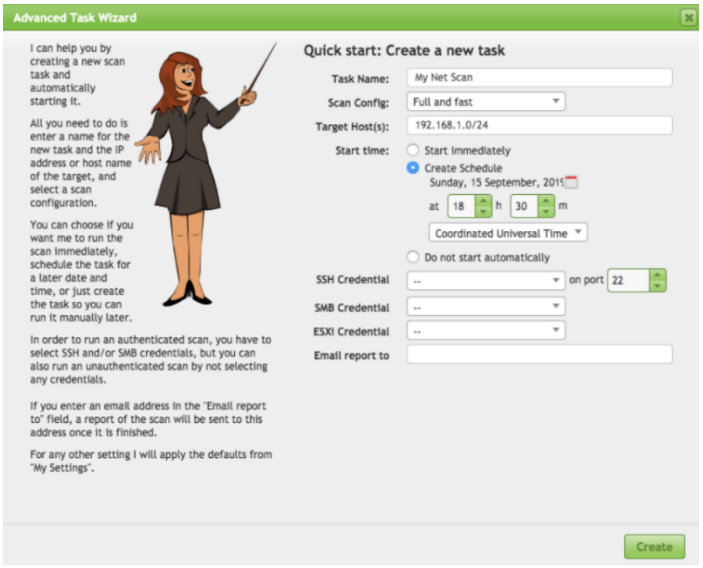


Рис. 2.7 Налаштування

Тут ми вже можемо налаштувати його на перевірку нашої мережі, а також відправку на електронну пошту повідомлень про результати сканування.

OpenVAS - це сканер вразливостей з відкритим вихідним кодом. OpenVAS призначений для активного моніторингу вузлів обчислювальної мережі на предмет наявності проблем, пов'язаних з безпекою, оцінки серйозності цих проблем і для контролю їх усунення. Активний моніторинг означає, що OpenVAS виконує якісь дії з вузлом мережі: сканує відкриті порти, посилає спеціальним чином сформовані пакети для імітації атаки або навіть авторизується на вузлі, отримує доступ до консолі управління, і виконує на ньому команди. Потім OpenVAS аналізує зібрані дані і робить висновки про наявність будь-яких проблем з безпекою. Ці проблеми, в більшості випадків стосуються встановленого на вузлі неоновлення ПО, в якому є відомі і описані уразливості, або ж небезпечно налаштованого ПО.

З цього маємо висновок:

Сканування вразливостей потрібно як домашнім, так і корпоративним мережам для боротьби з погрозами вразливостей. На ринку доступний широкий спектр сканерів. Як ви його виберете, залежить від вашого використання. Якщо ви хочете сканувати свою домашню мережу, OpenVAS може бути кращим варіантом, але якщо ви хочете сканувати і управляти великим корпоративним сектором, то слід пошукати комерційні сканери вразливостей.

В зрівнянні Nmap швидше за Nessus,але Nessus - більш повний

Nmap не є сканером вразливостей, це сканер мережевих сервісів, він тільки виявляє доступні мережеві сервіси, але не сканує їх на наявність вразливостей.

Порівняння роботи сканерів

OpenVAS використовується як альетрнатива Nessus,тож будемо порівнювати тільки ці два сканери:

OpenVAS та Nmap сильно відрізняються. Ви можете використовувати OpenVas для пошуку вразливостей, не знаючи, як їх шукати, оскільки OpenVAS відчуває численні атаки, зібрані з різних джерел, тоді як вам дійсно потрібно знати, що ви робите, де шукати, з Nmap.

Nessus згідно веб-сайту «Tenable», «Nessus HomeFeed» дає вам можливість сканувати вашу особисту домашню мережу (до 16 IP-адрес) з такою ж високошвидкісною, всебічною оцінкою і зручністю сканування без агентів, які подобаються передплатникам ProfessionalFeed. . Але при використанні сканера Nessus з домашньої подачею його не можна використовувати в професійній або комерційному середовищі.

OpenVAS: використовувалися стандартні сигнатури і програмне забезпечення OpenVAS 5 з відкритим вихідним кодом. Це безкоштовно для використання під Стандартної громадської ліцензією GNU (GNU GPL).

NeXpose: версія спільноти NeXposebил протестований. Згідно веб-сайту Rapid7, «Nexpose Community Edition працює на тому ж движку сканування, що і відзначений нагородами Nexpose Enterprise Edition, і пропонує багато хто з тих же функцій». У цій версії ви можете сканувати до 32 IP-адрес.

Nmap використовує записи CVE для поліпшення визначення версій Nmap. Він ідентифікує інформацію про версії відсканованого сервісу. Скрипти NSE використовуватимуть цю інформацію і створювати відомі CVE, які можна використовувати для експлуатації служби, що значно спрощує пошук вразливостей.

Нижче наведено приклад визначення версії Nmap без використання скриптів NSE. Nmap виявив одну службу SSH на 22 порту, що використовує версію 4.3

Nmap розпізнає наступні стану портів: open, filtered, closed, або unfiltered.

Open означає, що додаток на цільовій машині готове для прийняття пакетів на цей порт. Filtered означає, що брандмауер, фільтр, або щось інше в мережі блокує порт, так що Nmap не може визначити, чи є порт відкритим чи закритим. Closed - не пов'язані в даний момент ні з яким додатком, але можуть бути відкриті в будь-який момент. Unfiltered порти відповідають на запити Nmap, але не можна визначити, чи є вони відкритими або закритими.

«Filtered» Див.Рис 2.8

Рис. 2.8

«Open» Див.Мал. 2.9

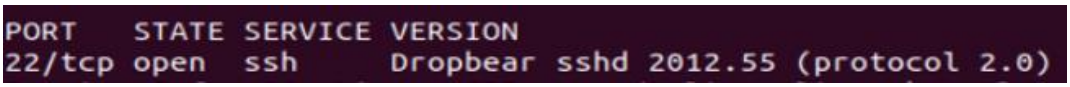


Рис. 2.9

2.3 Сканер Nmap для роботи написаного скрипта

Nmap - один з кращих мережевих сканерів. Сканування нашої мережі за допомогою Nmap було настільки ефективним, що Nmap зміг ідентифікувати порти, служби та інші комп'ютери в цій мережі. Він також може дізнатися, яка операційна система використовується комп'ютерами в нашій мережі. дійсно

корисна опція збереження результатів сканування Nmap в текстовий файл. Nmap доступний для Windows і Linux.

Nmap може бути вирішенням проблеми ідентифікації активності в мережі, оскільки він сканує всю систему і становить карту кожної її частини.

Загальною проблемою інтернет-систем є те, що вони занадто складні для розуміння звичайною людиною. Навіть невелика домашня система надзвичайно складна. Коли справа доходить до більш великих компаній і агентств, які мають справу з сотнями або навіть тисячами комп'ютерів в мережі, ця складність зростає в геометричній прогресії.

Щоб дізнатися, які порти відкриті і які ці правила, можна використовувати програму під назвою Nmap. Ця програма сканує мережу, до якої підключений комп'ютер, і виводить список портів, імена пристроїв, операційні системи і кілька інших ідентифікаторів, які допомагають користувачеві зрозуміти деталі, які стоять за станом їх підключення.

Nmap може використовуватися хакерами для отримання доступу до неконтрольованих портам в системі. Все, що потрібно зробити хакеру для успішного проникнення в цільову систему, - це запустити Nmap в цій системі, знайти уразливості і з'ясувати, як їх використовувати. Однак хакери - не єдині люди, які використовують програмну платформу. Компанії, що займаються IT-безпекою, часто використовують його як спосіб реплікації атак, з якими потенційно може зіткнутися система.

Nmap перевіряє мережу на наявність хостів і сервісів. Після виявлення програмна платформа відправляє інформацію тим хостам і службам, які потім відповідають. Nmap читає і інтерпретує відповідь, який повертається, і

використовує інформацію для створення карти мережі. Створена карта включає докладну інформацію про те, що робить кожен порт і хто (або що) його використовує, як підключаються вузли, що проходить і що не проходить через брандмауер, а також перераховує всі виникаючі проблеми безпеки.

Як все це досягається? Nmap використовує складну систему скриптів, які взаємодіють з усіма частинами мережі. Скрипти діють як засоби зв'язку між компонентами мережі та їх користувачами-людьми. Скрипти, які використовує Nmap, здатні виявляти уразливості, виявляти бекдори, використовувати уразливості і виявляти мережі. Nmap - надзвичайно потужне програмне забезпечення, але, як правило, для його правильного використання потрібно багато базових знань.

Компанії, що займаються інтернет-безпекою, можуть використовувати Nmap для сканування системи і розуміння існуючих слабких місць, які потенційно може використовувати хакер. Оскільки програма має відкритий вихідний код і безкоштовна, це один з найбільш поширених інструментів, що використовуються для сканування мереж на предмет відкритих портів і інших слабких місць. У Holm Security ми використовуємо цю технологію дуже ефективно, оскільки ми надаємо відмінну веб-службу безпеки, яка гарантує, що порти клієнтів залишаються надійно закритими для тих, кому не надано дозвіл.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГРАМНОГО ЗАПЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ВРАЗЛИВОСТЕЙ

3.1 Моніторинг за допомогою Nmap Scripting Engine та алгоритм програмного модулю

Розроблений скрипт,написаний на мові програмування “Lua” яка використовується для автоматизації широкого кола мережевих завдань. Який перевіряє які саме вразливості «CVE» мають дані порти. Працює з базою даних для надання інформації яку шкоду можуть нанести вразливості “CVE” та показує всю інформацію про них в пересиланні на сам сайт.У якості допоміжних інструментів використовує nmap для сканування стану портів(відкритий,закритий і т.д.) та веб-сайт «www.cvedetails.com» (який надає простий у використанні веб-інтерфейс для даних вразливостей CVE) .

Алгоритм програмного модулю:

- 1)Сканування стану портів допоміжною програмою «Nmap».
- 2)Пошук вразливостей на цих портах за допомогою написаного програмного модулю.
- 3)Виявлення та сортування вразливостей за їх рівнем небезпеки.
- 4)Надання їх назви з пересиланням на сайт та інформацією про цю вразливість.

Ось приклади роботи самого nmap без використання написаного скрипту.

Коли виявляється нова вразливість, потрібно швидко просканувати свої мережі, щоб виявити вразливі системи до того, як це роблять ті,хто бажає вам зла. Незважаючи на те, що Nmap не є комплексним сканером вразливостей, NSE є досить потужним для обробки навіть вимогливих перевірок вразливості.

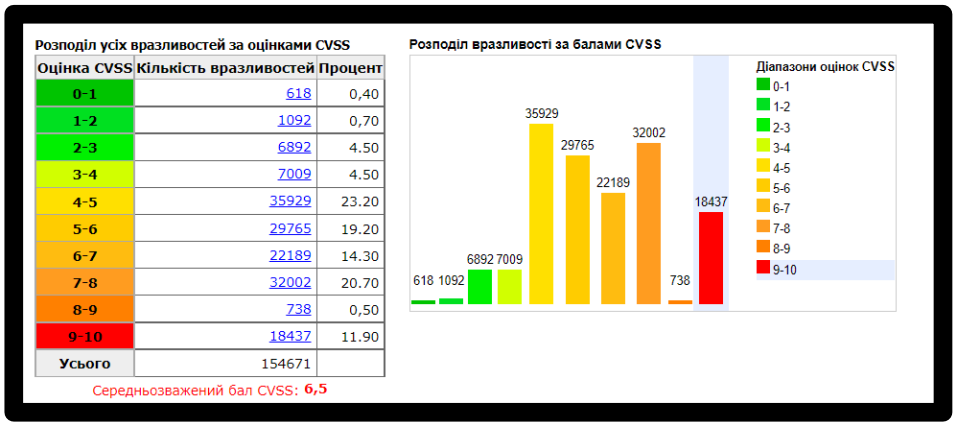
Нижче наведено приклад виявлення версії Nmap без використання скрипту. Nmap виявив одну службу SSH на порту 22, використовуючи версію "OpenSSH 4.3." Див.Рис.3.1.

```
nmap -sV -p22 1##.##.###.#21

Starting Nmap 7.60 ( https://nmap.org )
Nmap scan report for 1##.##.###.#21
Host is up (0.58s latency).

PORT    STATE   SERVICE VERSION
22/tcp  open    ssh     OpenSSH 4.3 (protocol 2.0)
```

Рис. 3.1. Сканування nmap без використання скрипту

Інформацію про вразливості я беру з бази даних cvedetails.com

Вона дозволяє переглянути які є вразливості,якого вони рівню, та як з ними боротися Див.Рис 3.2.



Розподіл усіх вразливостей за оцінками CVSS

Оцінка CVSS	Кількість вразливостей	Процент
0-1	618	0,40
1-2	1092	0,70
2-3	6892	4.50
3-4	7009	4.50
4-5	35929	23.20
5-6	29765	19.20
6-7	22189	14.30
7-8	32002	20.70
8-9	738	0,50
9-10	18437	11.90
Усього	154671	

Середньозважений бал CVSS: 6,5

Рис. 3.2. сайт cvedetails.com

Загальні уразливості і уразливості (CVE) - це уразливості системи безпеки, Служба вразливостей виявляє CVE, що впливають на ваші системи, і надає вам інформацію, необхідну для розуміння їх потенційних ризиків і способів їх усунення.

Common Vulnerability Scoring System (CVSS) - це відкритий стандарт, який використовується для розрахунку кількісних оцінок вразливості в безпеки комп'ютерної системи, зазвичай з метою зрозуміти пріоритет її виправлення.

Для проведення тестування приклад тестової мережі

Див.Рис. 3.3.

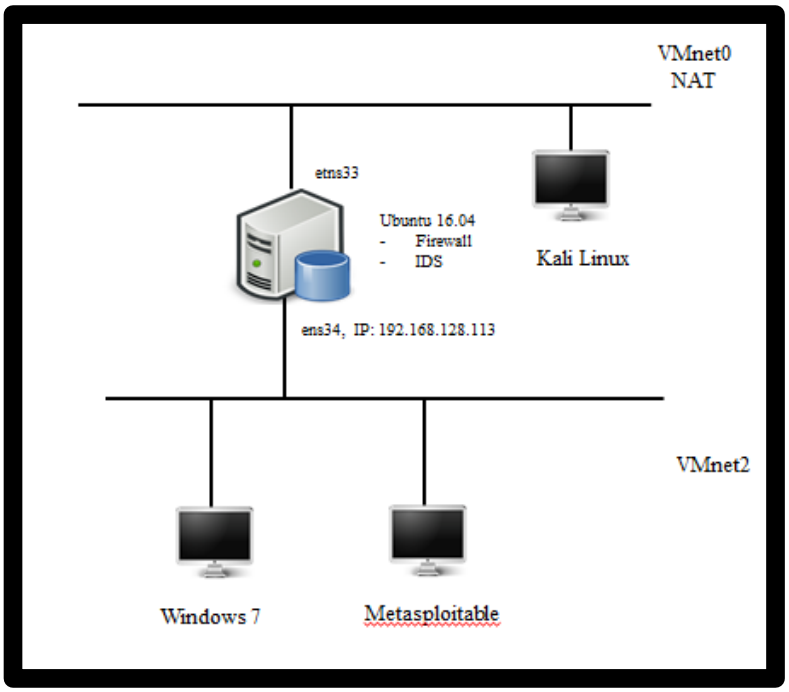


Рис. 3.3. Схема мережі

• Windows 7 з усіма встановленими оновленнями, запущеним додатком XAMPP, розгорнутими сервісами MySQL і Apache. Також розгорнута тестова система DVWA.

• Metasploitable 2 - операційна система з попередньо встановленими уразливими сервісами та додатками, яка використовується для тестування.

• Ubuntu 16.04 с встановленим IDS Suricata [10] і сконфігурованим iptables [9].

• Kali Linux - дистрибутив Linux, який використовується для тестування на проникнення.

Та ось Перелік вразливих сервісів в Metasploitable 2 наведені Див. 3.4.

Service	Port	Status
Vsftpd 2..four	21	Open
OpenSSH four.7p1 Debian 8ubuntu 1 (protocol 2.zero)	22	Open
Linux telnetd service	23	Open
Postfix smtpd	25	Open
ISC BIND 9.four.2	53	Open
Apache httpd 2.2.eight Ubuntu DAV/2	80	Open
A RPCbind service	111	Open
Samba smbd .X	139, 445	Open
r companies	512, 513, 514	Open
GNU Classpath grmiregistry	1099	Open
Metasploitable root shell	1524	Open
A NFS service	2048	Open

Рис. 3.4. Перелік стану

3.2 Реалізація скрипту

Для самого написання скрипта я використовував платформу “Lua” (скриптова

```lua
dependencies = {"http-vulners-regex"}
author = 'gmedian AT vulners DOT com'
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"vuln", "safe", "external", "default"}


local http = require "http"
local json = require "json"
local string = require "string"
local table = require "table"
local nmap = require "nmap"
local stdnse = require "stdnse"

local api_version="1.6"
local mincvss=stdnse.get_script_args("vulners.mincvss")
mincvss = tonumber(mincvss) or 0.0

portrule = function(host, port)
  local vers=port.version
  return vers ~= nil and vers.version ~= nil or
  (host.registry.vulners_cpe ~= nil)
end

local cve_meta = {
  __tostring = function(me)
      return ("\t%s\t%s\thttps://vulners.com/%s/%s%s"):format(me.id,
      me.cvss or "", me.type, me.id, me.is_exploit and '\t*EXPLOIT*'
      or '')
  end,
}
```

мова програмування) це легша версія “python” Див.Рис. 3.6.

Рис.3.6. Фрагмент написаного коду

Встановлення Nmap на якому будуть розміщені написані скрипти Див.Рис. 3.7.

```
root@artem-VirtualBox:/usr/share/nmap/vulscan#  sudo apt install nmap
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Предлагаемые пакеты:
  ndiff
Следующие НОВЫЕ пакеты будут установлены:
  nmap
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 270 пакетов не обновлено.
Необходимо скачать 5.174 kB архивов.
После данной операции объём занятого дискового пространства возрастёт на 24,0 MB.
Пол:1 http://ua.archive.ubuntu.com/ubuntu bionic/main amd64 nmap amd64 7.60-1ubuntu5 [5.174 kB]
Получено 5.174 kB за 1с (8.805 kB/s)
Выбор ранее не выбранного пакета nmap.
(Чтение базы данных … на данный момент установлено 130362 файла и каталога.)
Подготовка к распаковке …/nmap_7.60-1ubuntu5_amd64.deb …
Распаковывается nmap (7.60-1ubuntu5) …
Настраивается пакет nmap (7.60-1ubuntu5) …
Обрабатываются триггеры для man-db (2.8.3-2ubuntu0.1) …
```

Рис.3.7. Фрагмент завантаження nmap

Самі скрипти,назвав їх як “Linux-Vulners”,тобно «лінукс вразливості» створені в відповідній папці /usr/share/nmap/Linux-vulners Див.Рис.3.8.

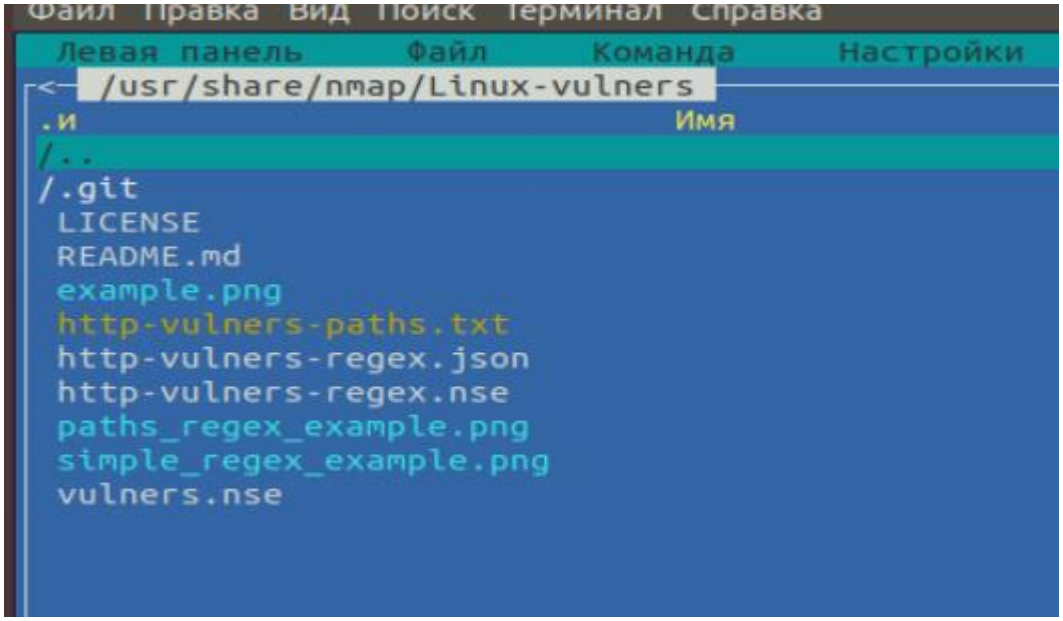


Рис.3.8. Розміщення самого скрипту

Для використання скрипта. Все, що нам потрібно зробити, це прописати --script Linux-vulners –sV 212.##.##.### до нашої команди Nmap і вказати, який сценарій використовувати.

Скрипт сканує хост на якому розміщена система Лінукс.В цьому випадку це власний айпі.

Опція nmap -sV забезпечує виявлення версії для кожної служби, яка використовується для визначення потенційних недоліків відповідно до ідентифікованим продуктом Див.Рис. 3.9.

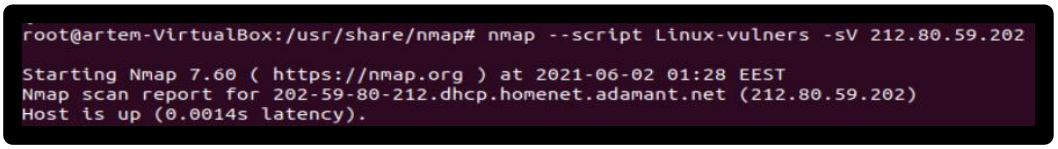


Рис. 3.9. Початок сканування

Nmap з ключем -sV сканує таргет на відкриті порти (а так само отримує інформацію про версії софта або служби на даному порту)

Після того він починає звіряти отриманий результат, і при збігу це видавати на екран.

Показує які вразливості виявлено і перенапрявляє на сайт з їх описанням: https://www.cvedetails.com/ Див.Рис. 3.10.

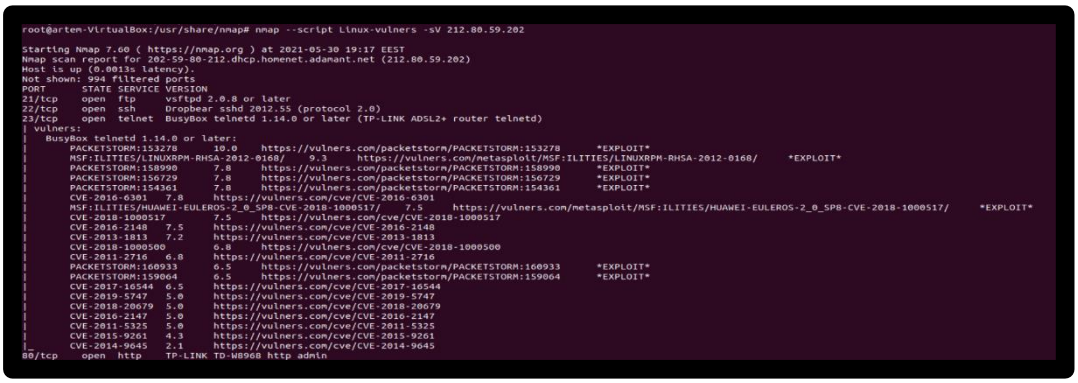


Рис. 3.10. Результат сканування

З цього скану ми бачимо статус,які вразливості виявлені на цих портах.Та їх рівні вразливості на систему зі ссиланням на сайт для ознайомлення з ними.

Для прикладу сканував систему яка знаходиться в віртуальній машині, в якій власноруч закрив відкриті порти (командою

-Sudo ufw deny “номер порту”) Див.Рис. 3.11.

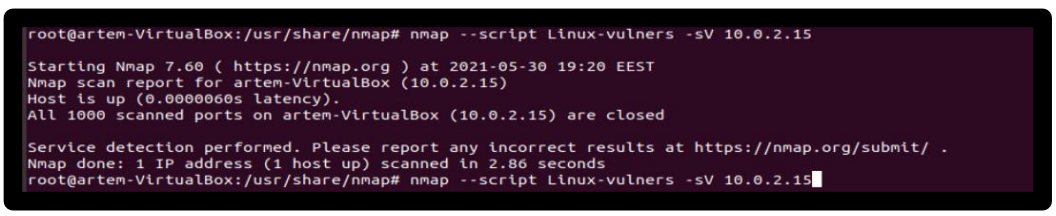


Рис. 3.11. Сканування з закритими портами

ВИСНОВКИ

В результаті виконання дипломної роботи,отримані наступні результати:

1) Проаналізовано архітектуру системи Linux, притаманні їй вразливості та їх види, ступінь захищеності системи, на основі проведеного аналізу були зроблені висновки про доцільність розробки програмного модуля моніторингу вразливостей Linux систем.
2) Розроблено програмний модуль моніторингу вразливостей Linux систем, який надає можливість виявляти вразливості в системі, завдяки пошуку по CVE; сортувати вразливості за рівнем небезпечності; вказувати CVE код вразливості, при використанні допоміжного сканеру стану портів «nmap».
3) Здійснено тестування розробленого програмного модулю моніторингу вразливостей Linux систем, що дозволило довести можливість його використання для вирішення поставленої задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.https://www.cvedetails.com/

2.https://wazuh.com/blog/detecting-vulnerable-software-on-linux-systems/

3.https://access.redhat.com/documentation/en-us/red_hat_insights/2020-10/html-single/assessing_and_monitoring_security_vulnerabilities_on_rhel_systems/index

4. https://linuxhint.com/top_vulnerability_scanning_tools/

5. https://www.uptycs.com/blog/best-linux-resources

6. https://linuxhint.com/top_vulnerability_scanning_tools/

7. http://citforum.ru/security/articles/win_lin/

8. https://losst.ru/opasnye-uyazvimosti-linux

9. https://www.opennet.ru/opennews/art.shtml?num=54542

10. https://hyperhost.ua/info/ru/uyazvimosti-linux-i-ih-vidyi

11. https://habr.com/ru/company/trendmicro/blog/545312

ДОДАТКИ

Лістинг програми

```
-- @usage
-- nmap -sV --script vulners [--script-args mincvss=<arg_val>] <target>
--
-- @args vulners.mincvss Limit CVEs shown to those with this CVSS score or
greater.
--
-- @output
--
-- 53/tcp   open     domain             ISC BIND DNS
-- | vulners:
-- |   ISC BIND DNS:
-- |     CVE-2012-1667    8.5    https://vulners.com/cve/CVE-2012-1667
-- |     CVE-2002-0651    7.5    https://vulners.com/cve/CVE-2002-0651
-- |     CVE-2002-0029    7.5    https://vulners.com/cve/CVE-2002-0029
-- |     CVE-2015-5986    7.1    https://vulners.com/cve/CVE-2015-5986
-- |     CVE-2010-3615    5.0    https://vulners.com/cve/CVE-2010-3615
-- |     CVE-2006-0987    5.0    https://vulners.com/cve/CVE-2006-0987
-- |_    CVE-2014-3214    5.0    https://vulners.com/cve/CVE-2014-3214
--
-- @xmloutput
-- <table key="cpe:/a:isc:bind:9.8.2rc1">
--   <table>
--     <elem key="is_exploit">false</elem>
--     <elem key="cvss">8.5</elem>
--     <elem key="id">CVE-2012-1667</elem>
--     <elem key="type">cve</elem>
--   </table>
--   <table>
--     <elem key="is_exploit">false</elem>
--     <elem key="cvss">7.8</elem>
--     <elem key="id">CVE-2015-4620</elem>
--     <elem key="type">cve</elem>
--   </table>
-- </table>

dependencies = {"http-vulners-regex"}
author = 'gmedian AT vulners DOT com'
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"vuln", "safe", "external", "default"}


local http = require "http"
local json = require "json"
local string = require "string"
local table = require "table"
local nmap = require "nmap"
local stdnse = require "stdnse"
```

```
local api_version="1.6"
local mincvss=stdnse.get_script_args("vulners.mincvss")
mincvss = tonumber(mincvss) or 0.0

portrule = function(host, port)
  local vers=port.version
  return vers ~= nil and vers.version ~= nil or (host.registry.vulners_cpe
~= nil)
end

local cve_meta = {
  __tostring = function(me)
      return ("\t%s\t%s\thttps://vulners.com/%s/%s%s"):format(me.id, me.cvss
or "", me.type, me.id, me.is_exploit and '\t*EXPLOIT*' or '')
  end,
}

---
-- Return a string with all the found cve's and correspondent links
--
-- @param vulns a table with the parsed json response from the vulners
server
--
function make_links(vulns)
  local output = {}

  if not vulns or not vulns.data or not vulns.data.search then
    return
  end

  for _, vuln in ipairs(vulns.data.search) do
    local v = {
      id = vuln._source.id,
      type = vuln._source.type,
      -- Mark the exploits out
      is_exploit = vuln._source.bulletinFamily:lower() == "exploit",
      -- Sometimes it might happen, so check the score availability
      cvss = tonumber(vuln._source.cvss.score),
    }

    -- NOTE[gmedian]: exploits seem to have cvss == 0, so print them anyway
    if v.is_exploit or (v.cvss and mincvss <= v.cvss) then
      setmetatable(v, cve_meta)
      output[#output+1] = v
    end
  end

  if #output > 0 then
    -- Sort the acquired vulns by the CVSS score
    table.sort(output, function(a, b)
        return a.cvss > b.cvss or (a.cvss == b.cvss and a.id > b.id)
      end)
    return output
  end
end
```

```
---
-- Issues the requests, receives json and parses it, calls
<code>make_links</code> when successfull
--
-- @param what string, future value for the software query argument
-- @param vers string, the version query argument
-- @param type string, the type query argument
--
function get_results(what, vers, type)
  local api_endpoint = "https://vulners.com/api/v3/burp/software/"
  local vulns
  local response
  local status
  local attempt_n=0
  local option={
    header={
      ['User-Agent'] = string.format('Vulners NMAP Plugin %s', api_version)
    },
    any_af = true,
  }

  stdnse.debug1("Trying to get vulns of " .. what .. " for type " .. type)

  -- Sometimes we cannot contact vulners, so have to try several more times
  while attempt_n < 3 do
    stdnse.debug1("Attempt ".. attempt_n .. " to contact vulners.")
    response =
http.get_url(('%s?software=%s&version=%s&type=%s'):format(api_endpoint,
what, vers, type), option)
    status = response.status
    if status ~= nil then
      break
    end
    attempt_n = attempt_n + 1
    stdnse.sleep(1)
  end

  if status == nil then
    -- Something went really wrong out there
    -- According to the NSE way we will die silently rather than spam user
with error messages
    stdnse.debug1("Failed to cantact vulners in several attempts.")
    return
  elseif status ~= 200 then
    -- Again just die silently
    stdnse.debug1("Response from vulners is not 200 but " .. status)
    return
  end

  status, vulns = json.parse(response.body)

  if status == true then
    stdnse.debug1("Have successfully parsed json response.")
    if vulns.result == "OK" then
      stdnse.debug1("Response from vulners is OK.")
```

```
      return make_links(vulns)
    else
      stdnse.debug1("Response from vulners is not OK with body:")
      stdnse.debug1(response.body)
    end
  else
    stdnse.debug1("Unable to parse json.")
    stdnse.debug1(response.body)
  end
end


---
-- Calls <code>get_results</code> for type="software"
--
-- It is called from <code>action</code> when nothing is found for the
available cpe's
--
-- @param software string, the software name
-- @param version string, the software version
--
function get_vulns_by_software(software, version)
  return get_results(software, version, "software")
end


---
-- Calls <code>get_results</code> for type="cpe"
--
-- Takes the version number from the given <code>cpe</code> and tries to get
the result.
-- If none found, changes the given <code>cpe</code> a bit in order to
possibly separate version number from the patch version
-- And makes another attempt.
-- Having failed returns an empty string.
--
-- @param cpe string, the given cpe
--
function get_vulns_by_cpe(cpe)
  local vers_regexp=":([%d%.%-%_]+)([^:]*)$"

  -- TODO[gmedian]: add check for cpe:/a  as we might be interested in
software rather than in OS (cpe:/o) and hardware (cpe:/h)
  -- TODO[gmedian]: work not with the LAST part but simply with the THIRD
one (according to cpe doc it must be version)

  -- NOTE[gmedian]: take only the numeric part of the version
  local _, _, vers, patch = cpe:find(vers_regexp)

  if not vers then
    return
  end

  stdnse.debug1("Got cpe " .. cpe .. " with version ".. vers .. " and patch
" .. (patch or "nil"))
```

```
  local output = get_results(cpe, vers, "cpe")

  if not output then
    if patch and patch ~= "" then
        local new_cpe

        new_cpe = cpe:gsub(vers_regexp, ":%1:%2")
        stdnse.debug1("Forming new cpe for another attempt " .. new_cpe)
        output = get_results(new_cpe, vers, "cpe")
    end
  end

  return output
end


action = function(host, port)
  local tab=stdnse.output_table()
  local changed=false
  local output

  for i, cpe in ipairs(port.version.cpe) do
    -- There are two cpe's for nginx, have to check them both
    cpe = cpe:gsub(":nginx:nginx", ":igor_sysoev:nginx")
    stdnse.debug1("Analyzing cpe " .. cpe)
    output = get_vulns_by_cpe(cpe)
    if cpe:find(":igor_sysoev:nginx") then
      cpe = cpe:gsub(":igor_sysoev:nginx", ":nginx:nginx")
      stdnse.debug1("Now going to analyze the second version " .. cpe)
      local output_nginx=get_vulns_by_cpe(cpe)
      if not output then
        output = output_nginx
      elseif output_nginx then
        -- Need to merge two arrays, sorted by cvss
        -- Presumably the former output contains by far less entries, so
iterate on it and insert into the latter
        -- pos will represent current position in output_nginx
        local pos=1
        for i, v in ipairs(output) do
          while pos <= #output_nginx and output_nginx[pos].cvss >= v.cvss do
              pos = pos + 1
          end
          table.insert(output_nginx, pos, v)
        end
        output = output_nginx
      end
    end
    if output then
      tab[cpe] = output
      changed = true
    end
  end

  -- NOTE[gmedian]: check whether we have pre-matched CPEs in registry (from
http-vulners-regex.nse in particular)
```

```
  if host.registry.vulners_cpe ~= nil and #host.registry.vulners_cpe > 0
then
    stdnse.debug1("Found some CPEs in host registry.")
    for i, cpe in ipairs(host.registry.vulners_cpe) do
      -- avoid duplicates in output, will still however make redundant
requests
      if tab[cpe] == nil then
        stdnse.debug1("Analyzing pre-matched cpe " .. cpe)
        output = get_vulns_by_cpe(cpe)
        if output then
          tab[cpe] = output
          changed = true
        end
      end
    end
  end

  -- NOTE[gmedian]: issue request for type=software, but only when nothing
is found so far
  if not changed then
    local vendor_version = port.version.product .. " " ..
port.version.version
    output = get_vulns_by_software(port.version.product,
port.version.version)
    if output then
      tab[vendor_version] = output
      changed = true
    end
  end

  if (not changed) then
    return
  end
  return tab
end
```