МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської підготовки
Кафедра Інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка рішення для моніторингу захищеності "Розумних" пристроїв підключених до мережі»

Виконав студент 2 курсу групи МІС-18 спеціальності 122 Комп'ютерні науки
Мартинюк Іван Іванович

Керівник к.т.н., доцент
Фразе-Фразенко О. О.

Консультант

Рецензент к.т.н., доцент
Щербина Ю.В

Одеса 2020

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

3G − Third Generation

4G LTE − Fourth Generation Long-Term Evolution

NaN − Network Analyzer

CIDR − Classless Inter-Domain Routing

CLI − Command line interface

CVE − Common Vulnerabilities and Exposures

DDoS − Distributed Denial of Service

DoS − Denial of Service

DVR − Digital Video Recorder

NVR − Network Video Recorder

HTTP − HyperText Transfer Protocol

IBM − International Business Machines

IDE − Integrated development environment

IDLE − Integrated DeveLopment Environment

IoT − Internet of Things

IP − Internet Protocol

IT - Information technology

JSON − JavaScript Object Notation

NAS − Network Attached Storage

OpenVAS − Open Vulnerability Assessment

OS − Operating system

POSIX − Portable Operating System Interface

PyPI − Python Package Index

SSH − Secure Shell

TCP − Transmission Control Protocol

UML − Unified Modeling Language

WI-FI − Wireless Fidelity

ВСТУП

Зараз як ніколи раніше підняте питання про інформаційну безпеку, а особливо про безпеку пристроїв які нас оточують. Майже у кожної людини вдома є свій IoT оточення. І складається вона із звичних пристроїв, різного роду розумної електроніки та побутової техніки, різноманітних датчиків, систем відео спостереження та інших цифрових пристроїв, з'єднаних з "мережею". Як і всім комп'ютерам підключеним до мережі цих пристроїв теж потрібен захист, ця область ще досить молода і з цієї причини в ній ще досить велика кількість вразливих місць. Проект який розглядається у цій дипломної роботі створювався в дослідницьких цілях, для отримання статистичних даних про захищеність мереж. Метою даної магістерської роботи є показати поточний рівень захищеності мережевих пристроїв. Так-як наявність в домашній мережі погано налаштованого або уразливого IoT-пристрою може спричинити за собою неприємні наслідки для його користувачів.

Мета цього проекту показати рівень захищеності користувачеві наочно, на основі статистичних Даних про сервіси і пристроях які "дивляться" в мережу і їх захищеності. До недавнього часу розмови про застосування шкідливого програмного забезпечення, що впливає на IoT, розглядали здебільшого в теоретичному ключі.

Однак інциденти останніх років и поточного року (2019) підтвердили, що сьогодні поняття безпеки в кіберпросторі в контексті IoT перейшло з категорії езотерики в головну новину світу інформаційної безпеки. За останні роки (2016 − 2019) сталася маса випадків пов'язаних з зламами розумних пристроїв. І несподівано робота над безпекою і и конфіденційністю підключених до мережі пристроїв та IoT стала новим пріоритетом. Ризики кібербезпеки в майбутньому, ймовірно зростуть, адже кількість підключених до Інтернету пристроїв зростає із загрозливою швидкістю. У 2017 році в Україні налічувалося близько 11 млн «підключених» пристроїв; до

2018 року, згідно з прогнозами, їх кількість подвоїться. У середньостроковій же перспективі до 2025 р. очікується зростання до 25 млрд таких пристроїв [1][1)].

Іншими словами, мільйони цифрових пристроїв можуть стати мішенню для кіберзлочинців, які бажають отримати доступ до життєвої важливих систем і/або персональних даних, або можуть стати інструментом для проведення DDoS-атак.

Незважаючи на складності для запобігання подібних ситуацій було розроблено безліч програмних і апаратних рішень, які на даний момент, допомагають користувачам захистити себе і свої пристрої. Тепер безпеку підключених пристроїв є ще і хорошим бізнесом для антивірусних компаній які і в цьому секторі пропонують свої програмні і апаратні рішення.

Тому, у міру того як українські компанії и громадяни планують застосування IoT-технологій, вони повинні завчасно продумати стратегію забезпечення безпеки і захисту даних, а також передбачити створення належної інфраструктури. Мережеві провайдери на рівні мережі повинні забезпечити захист користувачів мережі як різних приватних і державних організацій так і простих громадян.

[1)] [1] Фальстарт або розвідка боєм: як мобільні оператори впроваджують інтернет речей. URL: https://mind.ua/ru/publications/20194010-falstart-ili-razvedka-boem-kak-mobilnye-operatory-vnedryayut-internet-veshchej. (Дата звернення 13.11.2019).

1 АНАЛІЗ КІБЕРБЕЗПЕКИ IOT СЕКТОРУ

1.1 Основні проблеми розумних мережних пристроїв

У наші дні в інтернеті неважко знайти статті про виявлення хакерами або дослідниками вразливостей в таких речах, як автомобілі, кавоварки, побутова техніка, різноманітні гаджети і системи типу «розумний будинок».

Все це об'єднано в концепцію, що отримала назву «інтернет речей».

Сьогодні це одна з найбільш популярних тем в індустрії.

Проблема з численними дослідженнями на цю тему полягає в тому, що ми як користувачі не можемо «приміряти» їх на себе: якщо ми нездатні уявити себе в якості об'єкта атаки, ми не можемо як слід осмислити результати дослідження, навіть, якщо воно виконане на високому рівні і в ньому представлені докладні дані.

Загрози оточують нас, як користувачів всіх цих благ у цифровому середовищі, де мережеві технології охоплюють всі аспекти життя.

Це особливо важливо при побудові локальної мереж будинку й у невеликих офісах.

У типовому сучасному будинку до локальної мережі підключені близько п'яти пристроїв, а іноді і більше, які не є комп'ютерами, планшетами або телефонами.

Це такі пристрої, як телевізори, маршрутизатори, принтери, ігрові приставки, мережеві накопичувачі, а також різноманітні медіа плеєри, IP-камери, системи управління електроживленням будинку [2][1)].

Нами було проведено дослідження, яке допомогло визначити актуальність проблеми.

У процесі дослідження насамперед були досліджені не комп'ютери, планшети і смартфони, а інші підключені пристрої, такі як маршрутизатори

[1)] [2] Як забезпечити IoT-безпеку в епоху "Connected Everything". URL: https://www.anti-malware.ru/analytics/Threats_Analysis/How-to-secure-IoT-devices-with-nist-recommendations. (Дата звернення 10.10.2019).

(роутери), мережеві сховища (NAS), IP-камери, системи відео спостереження (NVR / DVR).

Основною метою було показати, наскільки на практиці вразливі наші будинки, і підкріпити це демонстрацією реальних уразливих частин.

Ці мережеві пристрої були обрані як найбільш привілейовані пристрою в наших домашніх мережах, маршрутизатори − об'єднують пристрої в наших мережах, системи відео спостереження − бачать нас і наше оточення, мережеві сховища − зберігають всі наші дані.

В цілому, ми цілком ефективно захищаємо свої обчислювальні пристрої, використовуючи для цього захисне та антивірусне ПЗ.

З різних джерел новин і книг з комп'ютерної грамотності ми знаємо, як підвищити рівень безпеки наших пристроїв. У наші дні більшість людей знають, що таке комп'ютерний вірус, розуміють, що потрібно використовувати надійні паролі і, що необхідно встановлювати оновлені версії програмного забезпечення.

У дослідженнях безпеки часто використовуються аналогії начебто замкнені двері в будинку, зроблений зі скла.

У цьому дослідженні варто приділити увагу, що навіть будучі в цілому орієнтованими на забезпечення інформаційної безпеки, ми концентруємося на захист обчислювальних пристроїв і, як правило, забуваємо про інші пристрої, які також підключені до наших мереж [3][1)].

Ми намагаємося захистити свої комп'ютери від злому і зараження, тому, що не хочемо допустити крадіжку даних, однак зберігаємо повні резервні копії своїх даних на пристрої такі як мережеві сховища, яке навіть більш уразлива, ніж сам комп'ютер.

Такі пристрої як мережеві сховища, маршрутизатори, IP-камери не мають захисного ПЗ, від мережі їх відділяє лише форма входу, але із-за

---

1) [3] П'ять актуальних практик кіберзахисту для корпорацій.
URL: https://delo.ua/special/pjat-aktualnyh-praktik-kiberzaschity-dlja-korpor-351219/.
(Дата звернення 10.10.2019).

якості ПЗ під управлінням якого перебувають ці пристрої про захист за допомогою пароля як про ефективний захист яка може зупинити зловмисника або шкідливе ПЗ від проникнення на пристрій можна забути.

Потрібно розуміти, що буквально будь-який пристрій, що підключається до мережі, може стати опорним пунктом» для атакуючої сторони або навіть невидимим «опорним пунктом» для кіберзлочинця.

Будучі зламаним, такий пристрій дозволить зловмиснику при необхідності мати доступ до внутрішньої мережі.

Передбачається, що аудиторією цього нашого невеликого дослідження будуть не тільки звичайні користувачі мережі, але Інтернет провайдери та інші приватні та державні організації.

За даними різних досліджень, проведених компаніями і інститутами у світі зараз налічується понад 10 мільярдів «розумних» пристроїв.

Така кількість потенційно вразливих гаджетів не залишилася непоміченою зловмисниками: за даними на квітень 2018 року в колекції «Лабораторії Касперського» знаходилося декілька тисяч різних зразків шкідливого ПЗ для «розумних» пристроїв, причому близько половини з них були додані в 2018 році.

Наявність в домашній мережі погано налаштованого або уразливого IoT-пристрою може спричинити за собою неприємні наслідки для його користувачів.

Один з найпоширеніших сценаріїв − включення пристрою в ботнет [4][1)].

Це, мабуть, самий нешкідливий варіант для його власника; інші варіанти використання більш небезпечні.

Так, пристрої з домашньої мережі можуть використовуватися в якості проміжної ланки для здійснення протизаконних дій.

[1)] [4] Kaspersky: IoT-зловредів проводять 20 тисяч атак за 15 хвилин. URL: https://threatpost.ru/kapersky-on-iot-state-of-security-in-h1-2019/34556/. (Дата звернення 10.10.2019).

Або просто (це далеко не найгірший сценарій), заражений пристрій може бути просто зламано.

Крім того, зловмисник отримавши доступ до IoT пристрою може шпигувати за його власником з метою подальшого шантажу.

Основні проблеми «розумних» пристроїв:

− неякісні мікропрограми виробника (вендора), ПЗ розумного пристрою;

− інтернет провайдери та їх модифікації ПЗ для розумного пристрою;

− користувачі які не достатньо грамотні у сфері комп'ютерних технологій та безпеки розумних пристроїв;

− недобросовісні представники світу IT користуються уразливостями в IoT пристроях;

− несвоєчасне випуск оновлень;

– виробники відмовляються від випуску оновлень;

− шкідливе ПЗ випущене в мережу.

1.2 Інциденти за участю вразливих підключаються до мережі пристроїв і їх виробників

Злам мережевого обладнання домашніх і корпоративних користувачів може стати причиною збитків десятки і сотні мільйонів доларів США. Так, ботнет Mirai, який спочатку поширювався шляхом зараження роутерів, завдав шкоди глобальній економіці в сотні мільйонів доларів США.

Цілком ймовірно, що виробники мережевих пристроїв, усвідомивши проблему, закрили «діри» і зробили свої системи невразливими для зловредів? Насправді, не зовсім так. Якісь окремі діри були виправлені, але масштабні зломи все ще трапляються.

Ось деякі з відомих які відбувалися нещодавно:

− У серпні 2019 група дослідників з компанії Red Ballon заявили про виявлення в маршрутизаторах Cisco серії 1001-X декількох вразливостей (CVE-2019-12647, CVE-2019-12654, CVE-2019-12658, ...);

− Майже одночасно з інформацією про проблеми з маршрутизаторами Cisco в мережі з'явилися новини про уразливість тисяч розумних роутерів Linksys. Діра в захисті цих пристроїв дозволяла (і до сих пір дозволяє) отримати віддалений неавторизований доступ до них. (CVE-2018-17208, CVE-2018-3955, CVE-2018-3953, ...);

− В кінці минулого року стало відомо про те, що невідомі зловмисники скомпрометували тисячі роутерів MikroTik для створення ботнету. Незважаючи на те, що вразливість виявили в квітні 2018 року, вона залишалася актуальною ще довгий час, оскільки далеко не всі власники роутерів стали встановлювати оновлення мікропрограми. (CVE-2018-7445, CVE-2018-1156);

− У той же період часу, кінець минулого року, фахівці з кібербезпеки виявили активну атаку зловмисників на роутери Dlink. Як виявилося, в прошивці містилася вразливість, яка давала можливість без особливих проблем зламати роутери з тим, щоб перенаправляти підключених користувачів на сайти або сервіси, прописані кіберзлочинцями. (CVE-2018-17990, CVE-2018-20114);

− У самому кінці 2017 року було зафіксовано масивна атака на роутери китайської компанії Huawei. Зловмисники використовували вразливість CVE-2017-17215 для отримання доступу до пристроїв Huawei У середині 2018 року всього за один день було заражено близько 18 000 мережевих пристроїв Huawei, з яких зловмисник сформував ботнет. Як стало відомо, кіберзлочинець скористався все тієї ж вразливістю;

− Проблема торкнулася і компанії ZyXEL. Але вони постаралися максимально швидко все виправити. Уразливість, яка була актуальною, отримала порядковий номер CVE-2019-9955. Схильні до неї були не роутери, а апаратні файрволли. Для всіх моделей вже випущений хотфіксов, так що

якщо обладнання не налаштоване на автоматичне оновлення то воно встановлене не буде. Випускаються зараз пристрої вже йдуть з виправленням, так що виявлені раніше уразливості не актуальні;

− Дослідники з Cisco Talos виявили вразливості в бездротових роутерах NETGEAR. Через некоректного налаштування "потиску" між клієнтом і точкою доступу зловмисник може перехопити закриті дані мережевих пристроїв. (CVE-2019-5016, CVE-2019-5017);

− Домашні маршрутизатори виробництва компанії TP-Link містять небезпечну уразливість, яка надає можливість атакуючому, що знаходиться в тій же мережі, виконати довільні команди з правами суперкористувача. За словами розробника з компанії Google (Matthew Garrett), проблема зачіпає багато пристроїв моделей TP-Link. (CVE-2018-19537).

Але це лише мала частина відбуваються постійно випадків безпеки за участю розумних пристроїв підключених до мережі Інтернет.

Вразливі набагато більше число моделей роутерів самих різних виробників. Раніше в мережі з'являлися новини про проблеми і з роутерами Realtek, ASUS, Dasan GPON і інших. Цілком ймовірно, що поки ви це читаєте, ваш власний роутер працює в інтересах зловмисників − то чи працює в якості елемента ботнету, то чи передає персональну інформацію.

Новий варіант Mirai оперує безліччю експлойтів для проникнення на підключення до Інтернету пристрою. Одинадцять з них, за словами експертів, Mirai-подібні зловредів раніше не використовували. Так що гонка озброєнь триває далі.

1.3 Відомі зразки шкідливого пз

Багато з шкідливих програм які призначені для зараження розумних IoT пристроїв знайшли популярність завдяки колосальний шкоди який вони завдали. А деякі стали відомі завдяки тому що замість захоплення пристроїв

вони їх захищали, але були і більш радикальні методи «захисту» користувачів від вразливих пристроїв:

− Mirai (в перекладі з японської означає «майбутнє») − черв'як і ботнет, утворений зламаними (скомпрометовані) пристроями типу «інтернет речей» (відеопрогравачі, «розумні» веб-камери, інше). На базі Mirai створено безліч різних шкідливі програм (Wicked, Satori, Okiru, Masuta, etc ...).

− VPNFilter − шкідницьке програмне забезпечення, розроблене для зараження маршрутизаторів. Станом на 24 травня 2018 року, за оцінками, заражено приблизно 0,5−1 мільйонів маршрутизаторів по всьому світу.

− Linux.Wifatch − це шкідлива програма з відкритим вихідним кодом, яка, як відомо, не використовувалася для зловмисних дій, а намагалася захистити пристрою від інших шкідливих програм.

− Hajime (в перекладі з японської означає «початок») − це шкідливе ПЗ, яке схоже на шкідливе ПЗ Wifatch в тому сенсі, що воно намагається захистити пристрій.

− BrickerBot був шкідливою програмою, яка намагалася назавжди знищити («зробити з них цеглини») небезпечні пристрої Інтернету речей. BrickerBot проникав в погано захищені пристрої і виконував шкідливі команди для їх відключення. Вперше він був виявлений Radware після того, як він атакував їх приманку в квітні 2017 року. 10 грудня 2017 року BrickerBot був видалений операторами ботнету.

2 РОЗГЛЯД ПРОГРАМНИХ РІШЕНЬ АНАЛІЗУ І МОНІТОРИНГУ ЗАХИЩЕНОСТІ ПРИСТРОЇВ

2.1 Пошукова система Shodan

Shodan − це пошукова система [5] [1] (рис. 2.1), яка дозволяє користувачеві знаходити конкретні типи комп'ютерів (веб-камери, маршрутизатори, сервери тощо), підключених до Інтернету за допомогою різноманітних фільтрів. Деякі також описали це як пошукову систему службових банерів, які є метаданими, які сервер відправляє назад клієнту. Це може бути інформація про серверне програмне забезпечення, параметри, які підтримує служба, вітальне пові-домлення або що-небудь інше, що може дізнатися клієнт перед взаємодією з сервером.

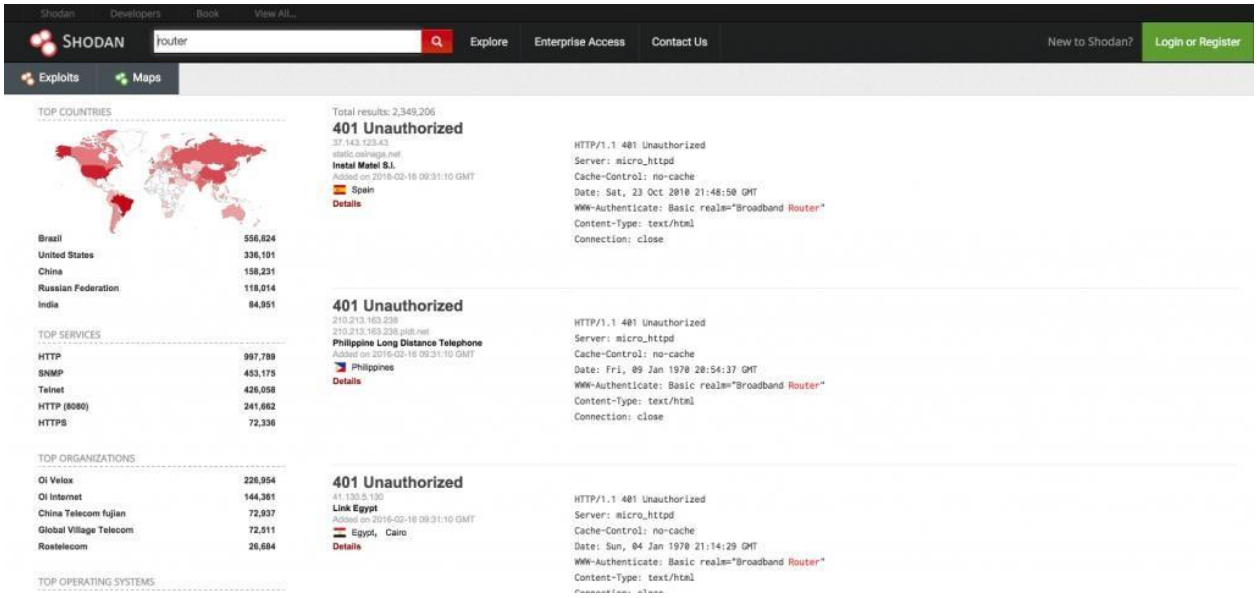

Рисунок 2.1 − Виконання пошукового запиту в Shodan

Shodan збирає дані здебільшого на веб-серверах (HTTP / HTTPS - порт 80, 8080, 443, 8443), а також на FTP (порт 21), SSH (порт 22), Telnet (порт 23),

[1] [5] Shodan is the world's first search engine for Internet-connected devices. URL: https://www.shodan.io/. (Дата звернення 01.11.2019).

SNMP (порт 161), IMAP (порти 143 або (зашифровано) 993), SMTP (порт 25), SIP (порт 5060) та протокол потокового потоку в реальному часі (RTSP, порт 554). Останні можна використовувати для доступу до веб-камер та їх відеопотоку. Його започаткував у 2009 році комп'ютерний програміст Джон Матерлі, який у 2003 році задумав ідею пошуку пристроїв, підключених до Інтернету.

Переваги:

− наочний і зручний інтерфейс;

− не вимагає високої кваліфікації;

− ведення повної історії перевірок і генерація звітів з різним рівнем деталізації;

− повна ідентифікація сервісів на випадкових портах;

− евристичний метод визначення типів і імен сервісів;

− інтеграція з браузерами завдяки розширенням;

− сервіс має API для роботи на різних мовах програмування;

− створення звітів про скануванні з графічними елементами.

Недоліки:

− заснований на Nmap, MassScan і інших утиліти для сканування мережі;

− не є сканером сам по собі а тільки зберігає дані про проведені раніше скануваннях;

− в безкоштовному режимі дуже обмежений;

− платне використання сервісу;

− ліцензійні обмеження;

− дуже дороге.

2.2 Проект Metasploit Framework

Metasploit Project − проект, присвячений інформаційній безпеці. Створено для надання інформації про уразливість, допомоги в створенні сигна-

тур для IDS, створення і тестування експлойтів [6][1)]. Найбільш відомий проект Metasploit Framework (рис. 2.2) − зручна платформа для створення і налагодження експлойтів. Крім того, проект включає в себе базу опкодов, архів шеллкодов і інформацію з досліджень інформаційної безпеки.

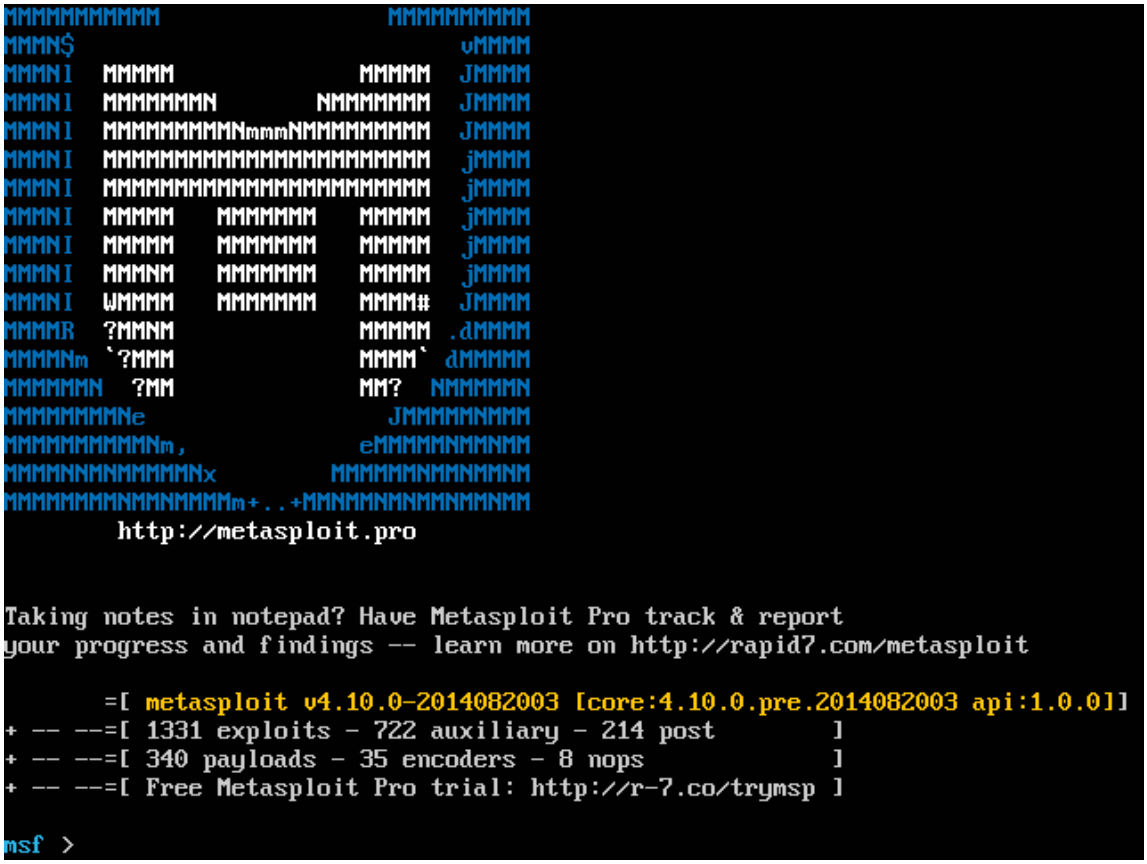

Рисунок 2.2 − Привітання командного рядка фреймворка

Metasploit був створений в липні 2003 року. Версія 1.0 була написана на мові Perl і містила псевдографічний інтерфейс на базі curses. Автором виступав HD Moore. При роботі над другою (2.x) версією до HD Moore приєднався Метт Міллер і кілька добровольців. Третя версія була повністю переписана на Ruby, її розробляла компанія Metasploit LLC (заснована розробниками в 2006 році). У 2008 році ліцензія Metasploit Framework була

[1)] [6] Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit. URL: https://www.metasploit.com/. (Дата звернення 01.11.2019).

змінена з пропрієтарної на BSD. У 2009 фірма Rapid7, що займається управлінням уразливими, оголосила про придбання Metasploit, популярного відкритого програмного пакета подвійного призначення для проведення тестів на проникнення. Некомерційна версія утиліти і раніше буде доступна для всіх бажаючих. Інструмент для створення, тестування і використання експлойтів. Дозволяє конструювати експлойти з необхідною в конкретному випадку «корисним навантаженням» (payloads), яка виконується в разі вдалої атаки, наприклад, установка shell або VNC сервера. Також фреймворк дозволяє шифрувати шеллкод, що може приховати факт атаки від IDS або IPS. Для проведення атаки необхідна інформація про встановлені на віддаленому сервері сервісах і їх версії, тобто потрібно додаткове дослідження за допомогою таких інструментів, як nmap або nessus.

Переваги:

− велика база для перевірки наявності вразливостей;

− можливість написання довільних сценаріїв (скриптів) на мові програмування Ruby;

− відкритий вихідний код;

− інтеграція з багатьма інструментами для проведення тесту на проникнення.

Недоліки:

− дуже складно в управлінні, необхідно ретельне ознайомлення з керівництвом користувача;

− не дружній до користувача інтерфейс;

− вимагає навичок роботи з терміналом.

2.3 Мережевий сканер Nmap

Nmap − вільна утиліта, призначена для різноманітних типів сканування IP-мереж з будь-якою кількістю об'єктів, визначення стану об'єктів сканов-

аній мережі (портів і відповідних їм служб) [7] [1]. Це досить складна у використанні програма та для виконання вузькоспеціалізованих завдань вимагає написання спеціальних сценаріїв мовою Lua. Так само складності у використанні для звичайних користувачів додає CLI інтерфейс (рис. 2.3). Використовується у фреймворку Metasploit для пошуку мережевих пристроїв і сервісів.

```
16982  zatevaxin rach  ~
$  nmap 10.10.10.1/24
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-02 21:51 EET
Nmap scan report for OpenWrt.lan (10.10.10.1)
Host is up (0.00040s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
53/tcp open  domain
80/tcp open  http

Nmap scan report for zux-p.lan (10.10.10.100)
Host is up (0.00037s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap scan report for server.local (10.10.10.101)
Host is up (0.00024s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
3000/tcp open  ppp
8000/tcp open  http-alt
8086/tcp open  d-s-n
9091/tcp open  xmltec-xmlmail

Nmap done: 256 IP addresses (3 hosts up) scanned in 40.75 seconds
```

Рисунок 2.3 − Взаємодія з програмою Nmap через CLI інтерфейс

Nmap також підтримує великий набір додаткових можливостей, а саме:

− визначення операційної системи віддаленого хоста з використанням відбитків стека TCP/IP;

− «невидиме» сканування;

− динамічне обчислення часу затримки та повтор передачі пакетів;

[1] [7] Nmap: the Network Mapper - Free Security Scanner. URL: https://nmap.org/. (Дата звернення 01.11.2019).

− паралельне сканування;

− визначення неактивних хостів методом паралельного ping-опитування;

− сканування з використанням помилкових хостів;

− визначення наявності пакетних фільтрів;

− сканування з використанням IP-фрагментації;

− довільна вказівка IP-адрес і номерів портів сканованих мереж;

− можливість написання довільних сценаріїв (скриптів) на мові програмування Lua.

Переваги:

− відкритий вихідний код;

− підтримується Open Source товариством;

− є безкоштовним;

− просто встановлюється і не вимагає налаштування;

− написання розширень під свої потреби на вбудованій скриптовій мові;

− безліч режимів сканування;

− гнучке управління параметрами сканування;

− одночасне сканування великої кількості комп'ютерів;

− підтримка плагінів;

− безліч надбудов (Графічні інтерфейси, Фреймворки, Плагіни).

Недоліки:

− дуже складно в управлінні, необхідно ретельне ознайомлення з керівництвом користувача;

− досить складний інтерфейс;

− не дружній до користувача інтерфейс;

− вимагає навичок програмування на мові Lua;

− вимагає навичок роботи з терміналом.

2.4 Zenmap інтерфейс взаємодії з Nmap

Zenmap − Офіційний, призначений для користувача інтерфейс (рис. 2.4) для програми Nmap. Програма написана на мові Python, і працює на платформах, на яких працюють Python і nmap [8][1)]. Часто використовувані настройки сканування можуть бути збережені як профілі для повторного використання. Майстер створення команд підтримує інтерактивне створення командних рядків Nmap. Скановані результати зберігаються і можуть переглядатися пізніше, порівнюватися з іншими результатами на факт змін. Результати зберігаються в пошуковій базі даних.

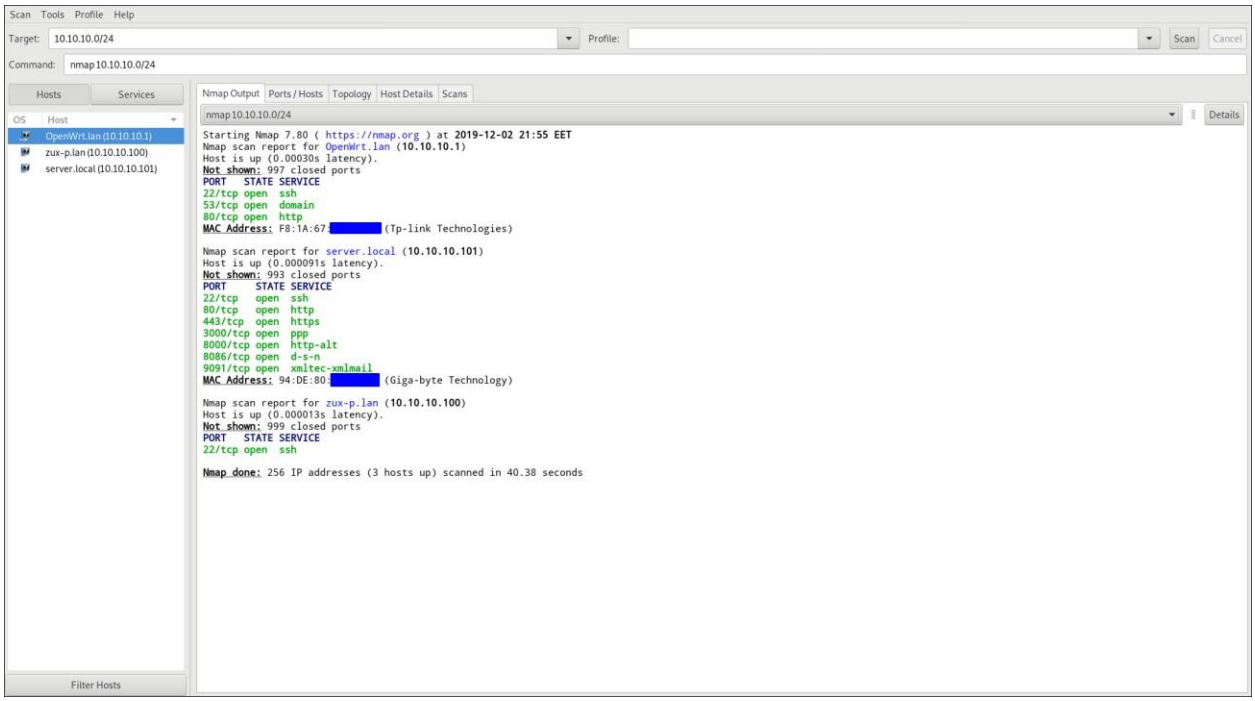

Рисунок 2.4 − Взаємодія з програмою Nmap через графічний інтерфейс Zenmap

Переваги:

− відкритий вихідний код;

1) [8] Zenmap - Official cross-platform Nmap Security Scanner GUI. URL: https://nmap.org/zenmap/. (Дата звернення 01.11.2019).

− підтримується Open Source товариством;

− є безкоштовним;

− простий користувацький інтерфейс;

− просто встановлюється і не вимагає налаштування;

− збереження інформації про попередні сесії сканування;

− одночасне сканування великої кількості комп'ютерів;

− можливість збереження параметрів сканування як профілів.

Недоліки:

− складне в управлінні, необхідно ретельне ознайомлення з керівництвом користувача;

− є лише графічним інтерфейсом полегшує роботу з Nmap;

− надбудова над Nmap, не є самостійною програмою.

2.5 NmapSI4 надбудова над Nmap

NmapSI4 − надбудова над Nmap яка являє собою графічний інтерфейс (рис. 2.5) для полегшення використання сканера Nmap, так само надає додаткові можливості, такі як: пошук знайденого сервісу по базах даних вразливостей, виявлення мережевих пристроїв, створення профілю для наступних операцій сканування мережі, повна підтримка всіх функцій Nmap [9][1)].

Переваги:

− відкритий вихідний код;

− підтримується Open Source товариством;

− є безкоштовним;

− зручний і дружній графічний інтерфейс;

− просто встановлюється і не вимагає налаштування;

− пошук опису вразливостей сервісів і пристроїв через вбудований браузер по базах даних вразливостей;

[1)] [9] NmapSI4 Security Interface. URL: http://web.archive.org/web/20180324022230/http://nmapsi4.org/. (Дата звернення 01.11.2019).

− збереження інформації про попередні сесії сканування;

− одночасне сканування великої кількості комп'ютерів;

− можливість створення профілів для різних типів сканування.

Недоліки:

− досить складний інтерфейс;

− надбудова над Nmap, не є самостійною програмою.

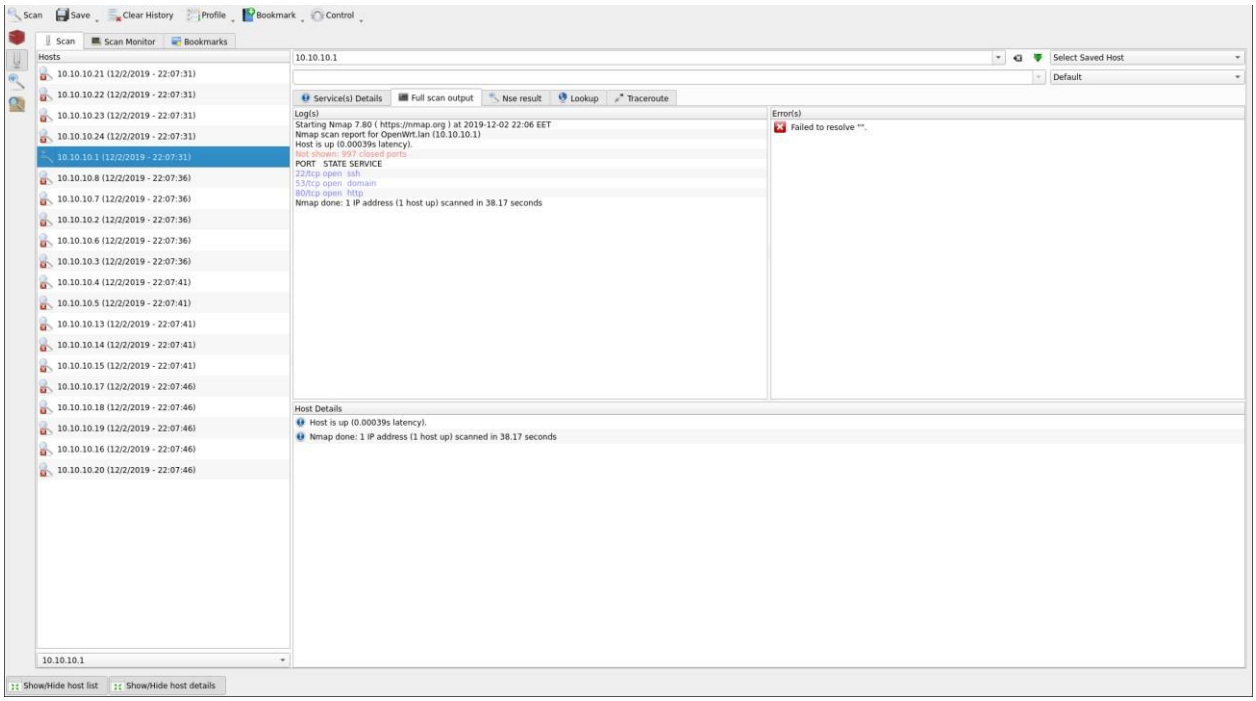

Рисунок 2.5 − Сканування локальної мережі за допомогою NmapSI4

2.6 Мережевий сканер XSpider

XSpider − платна пропрієтарна програма-сканер вразливостей (рис. 2.6). Випускається російською фірмою Positive Technologies [10][1)]. Спочатку, в 1998−2001 роках програма розроблялася Дмитром Максимовим, фахівцем з інформаційної безпеки для власних потреб. Пізніше, Дмитро в партнерстві з Євгеном Кірєєвим заснував фірму Positive Technologies, яка, крім усього інш-

1) [10] XSpider. URL: https://www.ptsecurity.com/ru-ru/products/xspider/. (Дата звернення 01.11.2019).

ого, стала розвивати продукт і пропонувати його користувачам на безоплатній основі. Влітку 2003 року була випущена версія 7.0, що стала платною.

Це є відмінним сканером вразливостей, але є платним і його вихідний код закритий. Інтелектуальний сканер XSpider здатний виявити максимальну кількість вразливостей в інформаційній системі до того, як вони будуть виявлені і використані зловмисниками.

Регулярне автоматичне сканування за допомогою XSpider вимагає мінімального втручання фахівця. Сканер працює віддалено, ніякого додаткового ПЗ на хости, що перевіряються встановлювати не потрібно. Після сканування XSpider видає чіткі рекомендації щодо усунення виявлених вразливостей.

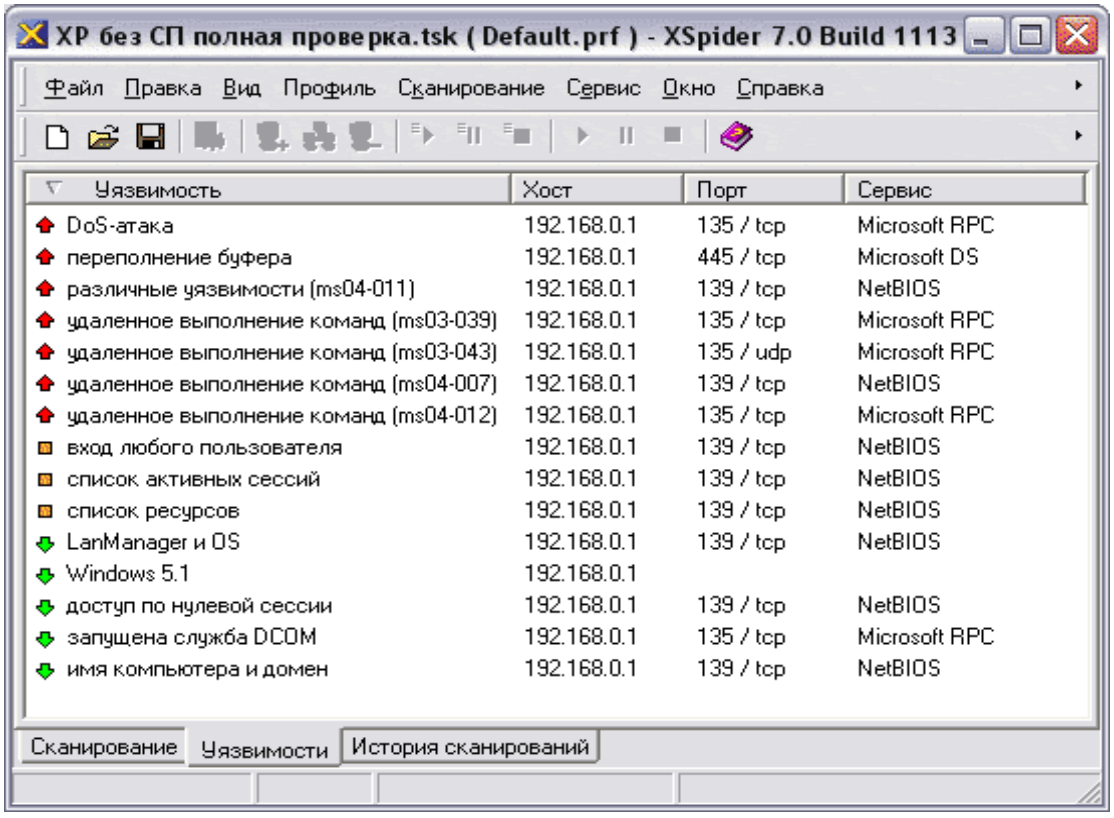

Рисунок 2.6 − Сканування локальної мережі за допомогою Xspider

Сканер XSpider також підтримує набір додаткових можливостей:

− контроль змін на сканованих вузлах дозволяє отримати повну картину захищеності в динаміці;

− повна ідентифікація сервісів на випадкових портах для виявлення вразливостей серверів з нестандартною конфігурацією;

− евристичний метод визначення типів і імен сервісів для визначення справжнього імені сервера і коректної роботи перевірок;

− аналіз структури HTTP-серверів для пошук слабких місць в конфігурації;

− проведення перевірок на нестандартні атаки.

Переваги:

− наочний і зручний інтерфейс;

− швидке встановлення і налаштування, що не вимагає високої кваліфікації;

− гнучкий планувальник завдань для автоматизації роботи;

− одночасне сканування великої кількості комп'ютерів;

− ведення повної історії перевірок і генерація звітів з різним рівнем деталізації;

− вбудована документація;

− вбудована контекстна довідка і підручник;

− контроль змін на сканованих вузлах;

− повна ідентифікація сервісів на випадкових портах;

− перевірка слабкості парольного захисту;

− евристичний метод визначення типів сервісів;

− евристичний метод визначення імен сервісів;

− створення звітів про скануванні з графічними елементами.

Недоліки:

− підтримується тільки операційною системою Windows;

− платне програмне забезпечення;

− ліцензійні обмеження;

− дуже дороге.

2.7 Сканер вразливостей OpenVAS

OpenVAS (Open Vulnerability Assessment) − фреймворк складається з декількох сервісів і утиліт, який дозволяє проводити сканування вузлів мережі на наявність вразливостей і управління вразливостями (рис. 2.7). Розробник Greenbone Networks GmbH [11][1)].

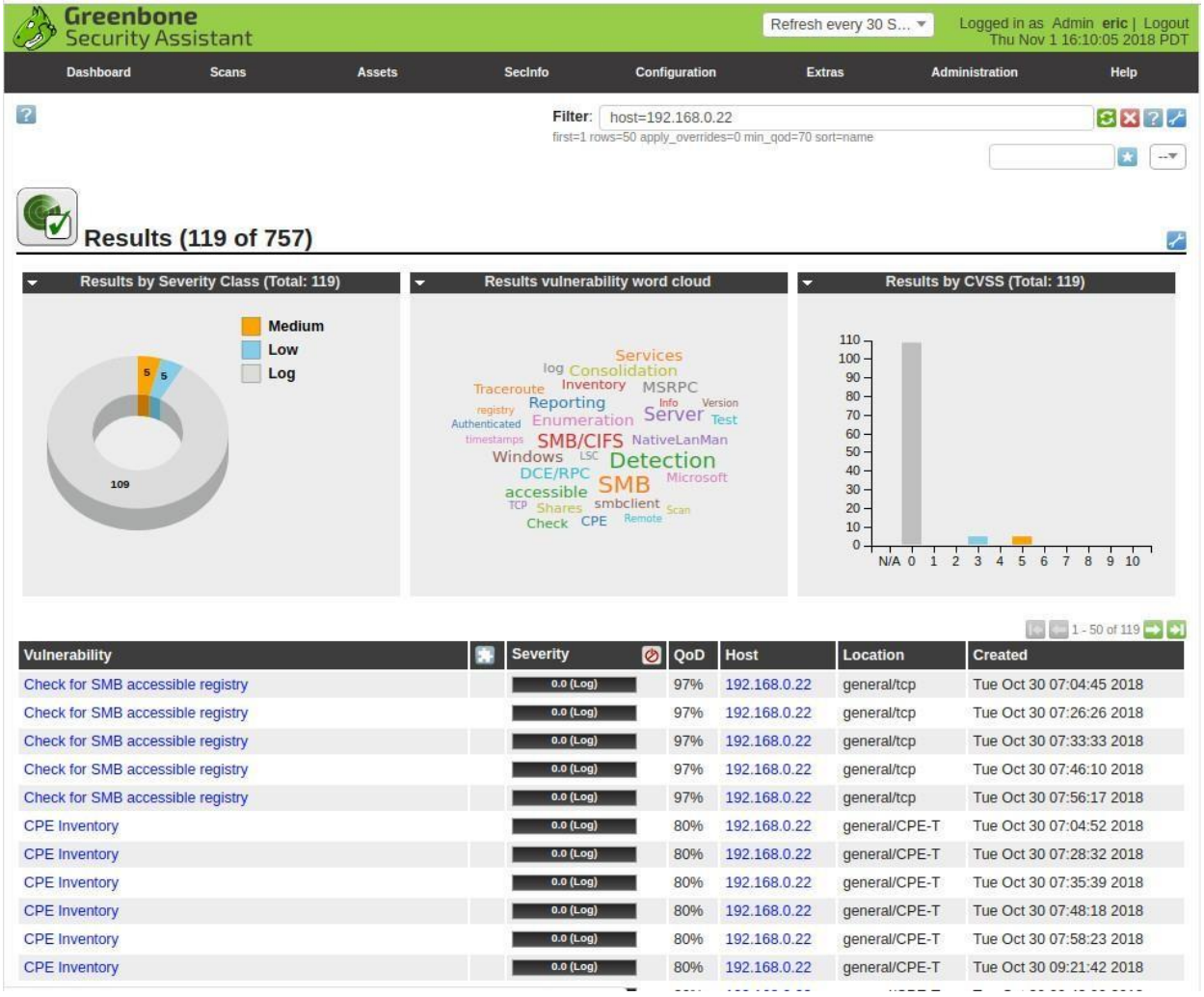

Рисунок 2.7 − Сканування хоста мережі за допомогою OpenVAS

[1)] [11] OpenVAS - Open Vulnerability Assessment Scanner. URL: http://www.openvas.org/. (Дата звернення 01.11.2019).

Переваги:

− відкритий вихідний код;

− підтримується Open Source товариством та розробником;

− генерація звітів з різним рівнем деталізації;

− створення звітів про скануванні з графічними елементами;

− одночасне сканування великої кількості комп'ютерів;

− є найрозвиненішим вільним сканером вразливостей;

− має клієнт-серверну архітектуру;

− вбудована документація;

− підтримка плагінів;

− є безкоштовним.

Недоліки:

− встановлення і налаштування, вимагає високої кваліфікації;

− досить складний інтерфейс;

− не всі плагіни є безкоштовними.

2.8 Сканер вразливостей Nessus

Nessus − програма для автоматичного пошуку відомих вад в захисті інформаційних систем. Вона здатна виявити найбільш часто зустрічаються види вразливостей. Розробник Tenable Network Security [12][1)]. Перш за все використовується для сканування портів і визначає сервіси, що використовують їх. Також проводиться перевірка сервісів по базі вразлив-остей. Для тестування вразливостей використовуються спеціальні плагіни, написані на мові NASL (Nessus Attack Scripting Language).

База вразливостей оновлюється щотижня, проте для комерційних передплатників є можливість завантажувати нові плагіни без семиденної затримки.

[1)] [12] Nessus Vulnerability Assessment | Tenable®.
URL: https://www.tenable.com/products/nessus. (Дата звернення 01.11.2019).

При відключеній опції «safe checks» деякі тести на уразливості, що використовуються Nessus, можуть привести до порушень в роботі сканованих систем (рис. 2.8).

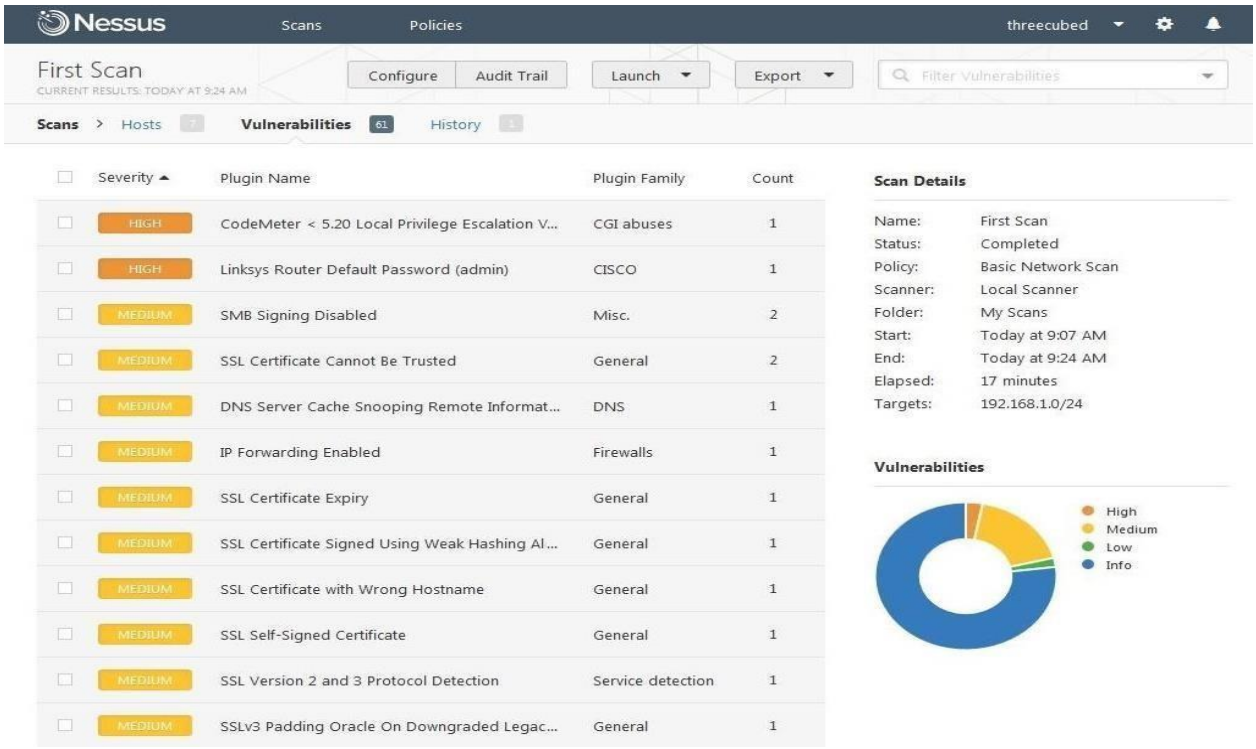

Рисунок 2.8 − Сканування локальної мережі за допомогою Nessus

Переваги:

− відкритий вихідний код;

− підтримується Open Source товариством та розробником;

− одночасне сканування великої кількості комп'ютерів;

− генерація звітів з різним рівнем деталізації;

− створення звітів про скануванні з графічними елементами;

− має клієнт-серверну архітектуру;

− підтримка плагінів;

− є частково безкоштовним.

Недоліки:

− встановлення і налаштування, вимагає високої кваліфікації;

− досить складний інтерфейс;

− платна підтримка;

− платні компоненти;

− ліцензійні обмеження.

3 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ, ТЕХНОЛОГІЙ І СЕРВІСІВ

3.1 Інструменти розробки ПЗ

Python − інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією [13][1)]. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах.

В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна.

Серед основних її переваг можна назвати такі:

− чистий синтаксис (для виділення блоків слід використовувати відступи);

− переносність програм (що властиве більшості інтерпретованих мов);

− стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);

− можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

− стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і, яке написане на мові Python;

− зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної

[1)] [13] The Python Wiki. URL: https://wiki.python.org/moin/. (Дата звернення 12.11.2019).

величини, у діалоговому режимі може використовуватися як потужний калькулятор);

− відкритий код.

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування (рис. 3.1). Елегантний си-нтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Рисунок 3.1− Логотип мови програмування Python

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++. Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Visual Studio Code − редактор початкового коду, розроблений Microsoft для Windows, Linux і macOS. Позиціюється як «легкий» редактор коду для платформонезалежної розробки веб та хмарних додатків [14][1)]. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне

[1)] [14] Documentation for Visual Studio Code. URL: https://code.visualstudio.com/docs. (Дата звернення 12.11.2019).

забезпечення з відкритим початковим кодом, але готові збірки розповсюджуються під пропрієтарною ліцензією.

Visual Studio Code заснований на Electron − фреймворк, що дозволяє з використанням Node.js розробляти настільні додатки, які працюють на движку Blink (рис. 3.2). Незважаючи на те, що редактор заснований на Electron, він не використовує редактор Atom. Замість нього реалізується веб-редактор Monaco, розроблений для Visual Studio Online.

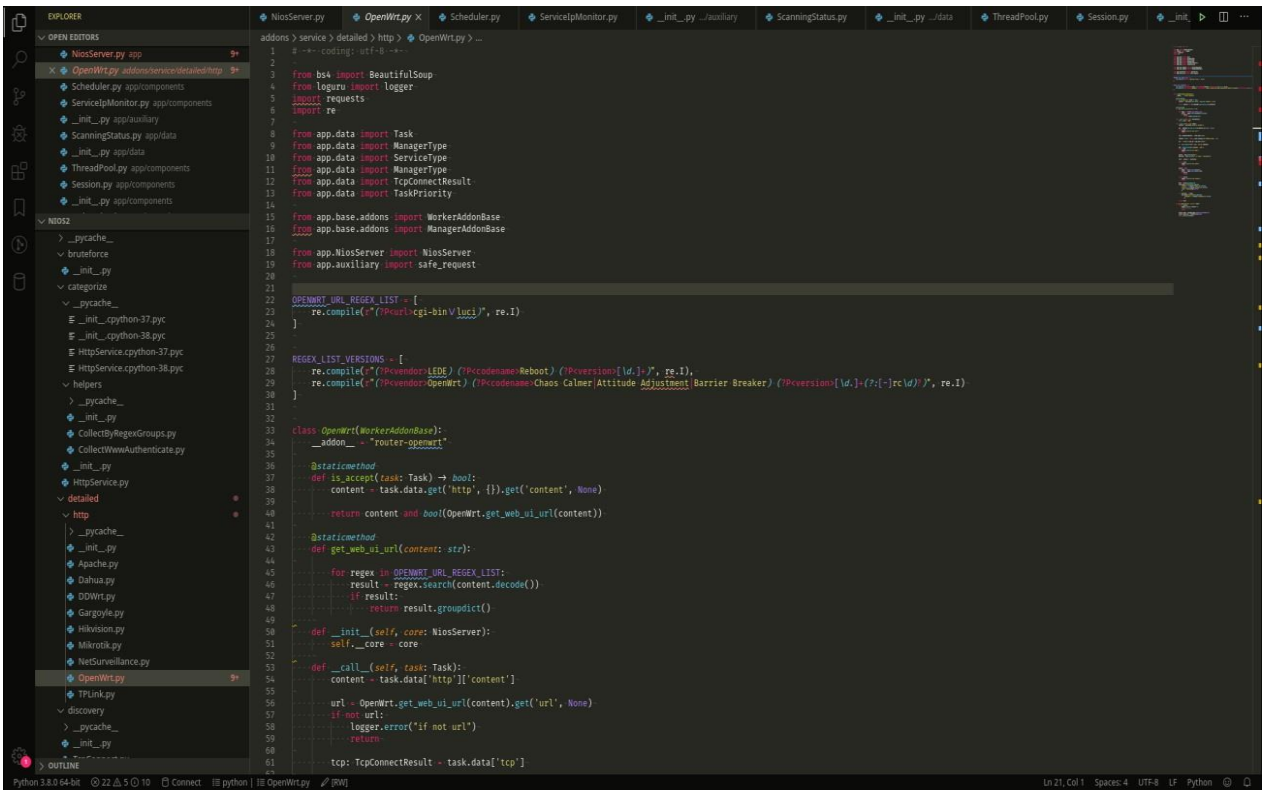

Рисунок 3.2 − Редактор Visual Studio Code з відкритим в ньому проектом NaN

Visual Studio Code − це редактор вихідного коду. Він підтримує ряд мов програмування, підсвічування синтаксису, IntelliSense, рефакторинг, налагодження, навігацію по коду, підтримку Git та інші можливості. Багато можливості Visual Studio Code не доступні через графічний інтерфейс, найчастіше вони використовуються через палітру команд або JSON файли (наприклад,

призначені для користувача настройки). Палітра команд представляє собою подобу командного рядка, яка викликається поєднанням клавіш.

Visual Studio також дозволяє замінювати кодову сторінку при збереженні документа, символи перекладу рядка і мову програмування поточного документа.

З 2018 року з'явилися розширення Python для Visual Studio Code з відкритим вихідним кодом. Воно надає розробникам широкі можливості для редагування, налагодження і тестування коду. На березень 2019 року за допомогою вбудованого в продукт призначеного для користувача інтерфейсу можна завантажити і встановити кілька тисяч розширень.

JavaScript − мультіпарадігменна мова програмування (рис. 3.3). Підтримує об'єктно-орієнтований, імперативний і функціональний стилі [15][1)].

Рисунок 3.3 − Логотип мови програмування JavaScript

JavaScript − зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок. Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти

[1)] [15] JavaScript. URL: https://en.wikipedia.org/wiki/JavaScript. (Дата звернення 15.11.2019).

першого класу. На JavaScript вплинули багато мов, при розробці була мета зробити мову схожу на Java, але при цьому легкою для використання непрограмістів. Мовою JavaScript не володіє ніяка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в веб-розробці. Назва «JavaScript» є зареєстрованим товарним знаком компанії Oracle Corporation в США.

Highcharts − бібліотека для створення чартів написана на JavaScript, дозволяє легко додавати інтерактивні, анімовані графіки на сайт або в веб-додаток [16][1)]. На даний момент чарти підтримують велику кількість діаграм лінійних, кругових, стовпчикових, розсіюючих і багатьох інших типів (рис. 3.4).

Рисунок 3.4 − Логотип бібліотеки Highchart

Підтримуються наступні типи виведення графіків: line, spline, area, areaspline, column, bar, pie і scatter. І вони можуть бути комбіновані між один одним при виведенні кінцевому користувачеві, і мають можливість відключення будь-якого з них в режимі реального часу безпосередньо користувачем для зручності розбирання інформації.

Дані для побудови графіків можуть бути взяті як з самого JavaScript, так з локального файлу, бази даних, або з віддаленого сервера.

Переваги:

[1)] [16] Interactive JavaScript charts. URL: https://www.highcharts.com/. (Дата звернення 15.11.2019).

– Чарти працюють на чистому JavaScript і не потребують будь-яких плагінів;

– Висновок чартів досить простий, і вони можуть бути використані навіть новачками в веб-розробці;

– Є збільшення окремих областей;

– Підтримка скінів / тем оформлень;

– Підтримка tooltip з виведенням інформації;

– У більшості типів чартів є підтримка date-time X-осі;

– Розмір бібліотеки близько 18кб.

Bootstrap – вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення (рис. 3.5) [17][1)].

Рисунок 3.5 – Логотип библиотеки Bootstrap

Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів. Основні інструменти Bootstrap:

---

1) [17] Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/. (Дата звернення 15.11.2019).

− Сітки − заздалегідь задані розміри колонок, які можна відразу ж використовувати, наприклад, ширина колонки 140 px відноситься до класу .span2 (.col-md-2 в третій версії фреймворка), який можна використ-овувати в CSS-описі документа.

− Шаблони − фіксований або гумовий шаблон документа;

− Типографіка − опис шрифтів, визначення деяких класів для шрифтів, таких як код, цитати і т.п;

− Медіа − надає деяке упорядкування зображень та відео;

− Таблиці − засоби оформлення таблиць, аж до додавання функціональності сортування;

− Форми − класи для оформлення форм і деяких подій, що відбуваються з ними;

− Алерти − оформлення діалогових вікон, підказок і спливаючих вікон.

Normalize.css − CSS-фреймворк, створений для спрощення роботи верстальника, швидкості розробки і виключення максимально можливого числа помилок верстки (проблеми сумісності різних версій браузерів і т.д.) (рис. 3.6) [18][1)].

Normalize.css

Рисунок 3.6 − Логотип бібліотеки normalize.css

[1)] [18] Normalize.css: Make browsers render all elements more consistently. URL: https://necolas.github.io/normalize.css/. (Дата звернення 15.11.2019).

Як і бібліотеки скриптових мов програмування, CSS-бібліотеки, зазвичай мають вигляд зовнішнього css-файлу, «підключаються» до проекту (додаю-ться до заголовку веб-сторінки) .

Дозволяє недосвідченому в тонкощах верстки програмісту або дизайнерові правильно створити HTML-макет:

− Верстка на базі шарів, а не таблиць;

− Більш швидка розробка;

− Кросбраузерність;

− Можливість використання генераторів коду і візуальних редакторів.

− Одноманітність коду при роботі в команді дозволяє знизити число розбіжностей при розробці.

jQuery − набір функцій JavaScript, що фокусується на взаємодії JavaScript і HTML (рис. 3.7). Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX. Розробка jQuery ведеться командою добровольців на пожертвування [19][1)].

Рисунок 3.7 − Логотип бібліотеки JQuery

Regular expressions − формальна мова пошуку і здійснення маніпуляцій з підрядками в тексті, заснована на використанні метасимволов (символів-джокерів, англ. Wildcard characters) [20] [2)]. Для пошуку використовується рядок-зразок (англ. Pattern, українською її часто називають «шаблоном»,

1) [19] jQuery. URL: https://jquery.com/. (Дата звернення 15.11.2019).
2) [20] Regular-Expressions.info - Regex Tutorial, Examples and Reference - Regexp Patterns. URL: https://www.regular-expressions.info/. (Дата звернення 15.11.2019).

«маск-ою»), що складається з символів і метасимволов і задає правило пошуку. Для маніпуляцій з текстом додатково задається рядок заміни, який також може містити в собі спеціальні символи (рис. 3.8).

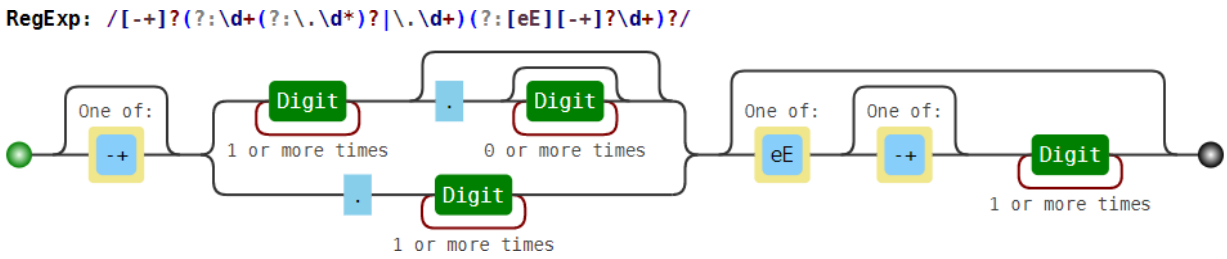

Рисунок 3.8 − Графічний приклад роботи регулярних виразів

JSON − текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстові форматів, JSON легко читається (рис. 3.9) [21][1)].

Рисунок 3.9 − Логотип мови JSON

Формат JSON був розроблений Дугласом Крокфордом. Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA-262 1999 роки), формат вважається незалежним від мови і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує гото-вий код для створення і обробки даних в форматі JSON. За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для сериализації складних структур. Застосовується у

[1)] [21] JavaScript Object Notation. URL: https://www.json.org/. (Дата звернення 22.11.2019).

веб-додатках як для обміну даними між браузером і сервером (AJAX), так і між серверами (програмні HTTP-сполучення).

Git − розподілена система керування версіями файлів та спільної роботи. Проект створив Лінус Торвальдс для управління розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано. Git є однією з, найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок [22][1)].

Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів. Система спроектована як набір програм, спеціально розроблених з урахуванням їх використання в сценаріях. Це дозволяє зручно створювати спеціалізовані системи контролю версій на базі Git або призначені для користувача інтерфейси (рис. 3.10).

Рисунок 3.10 − Логотип системи контролю версій Git

Git підтримує швидке поділ і злиття версій, включає інструменти для візуалізації та навігації по нелінійної історії розробки. Ядро Git є набором утиліт командного рядка з параметрами. Всі налаштування зберігаються в текстових файлах конфігурації. Така реалізація робить Git легко переносним

[1)] [22] Система контролю версій Git. URL: https://git-scm.com/. (Дата звернення 22.11.2019).

на будь-яку платформу і дає можливість легко інтегрувати Git в інші системи.

GNU/Linux − загальна назва UNIX-подібних операційних систем на основі однойменного ядра. Це один із найвидатніших прикладів розробки вільного та відкритого програмного забезпечення. На відміну від власницьких операційних систем (на кшталт Microsoft Windows та MacOS X), їх вихідні коди доступні усім для використання, зміни та розповсюдження абсолютно вільно та безкоштовно (рис. 3.11) [23][1)].

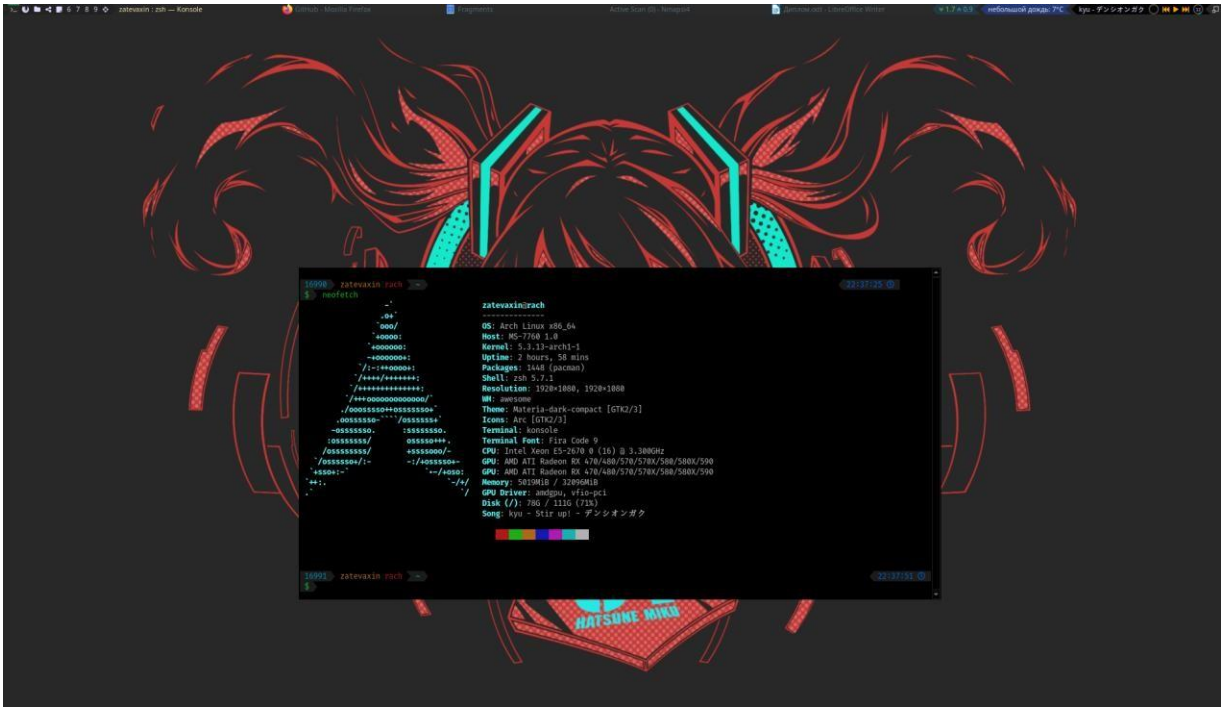

Рисунок 3.11 − Один з можливих користувацьких інтерфейсів системи GNU/Linux

Linux, спершу розроблений для використання окремими ентузіастами на своїх персональних комп'ютерах, пізніше, завдяки підтримці таких компаній, як IBM, Sun Microsystems, HP, Novell та інших, набув неабиякої

[1)] [23] Linux. URL: https://en.wikipedia.org/wiki/Linux. (Дата звернення 20.11.2019).

популярності як серверна операційна система. Лінукс портовано на велику кількість апаратних платформ.

Тепер ця ОС досить успішно використовується як на мейнфреймах та суперкомп'ютерах, так і вбудована в багато інших пристроїв (смартфони, планшетні ПК, маршрутизатори комп'ютерних мереж (роутери), пристрої автоматики, системи управління телевізорами та ігровими консолями тощо).

Від середини 1990-х років Linux все частіше встановлюється і на настільні комп'ютери. Так, станом на грудень 2019 року його частка складала 4.8% світового ринку операційних систем на персональних комп'ютерах без урахування використання на серверах та спеціалізованих прис-троях .

UML − це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація [24][1)].

Для проекту було використано тільки два типи UML діаграм, Діаграма класів та Діаграма варіантів використання, але в мові UML їх безліч, діаграми класів, використання і т.д.

Діаграма варіантів використання (Use case diagram) − діаграма, на якій відображені відносини, що існують між акторами і варіантами використання. Основне завдання − становити собою єдиний засіб, що дає можливість замовнику, кінцевому користувачеві і розробнику спільно обговорювати функціональність і поведінку системи.

[1)] [24] Unified Modeling Language.
URL: http://en.wikipedia.org/wiki/Unified_Modeling_Language. (Дата звернення 20.11.2019).

Діаграма класів (Class diagram) − статична структурна діаграма, що описує структуру системи, що демонструє класи системи, їх атрибути, методи і залежності між класами (рис. 3.12).

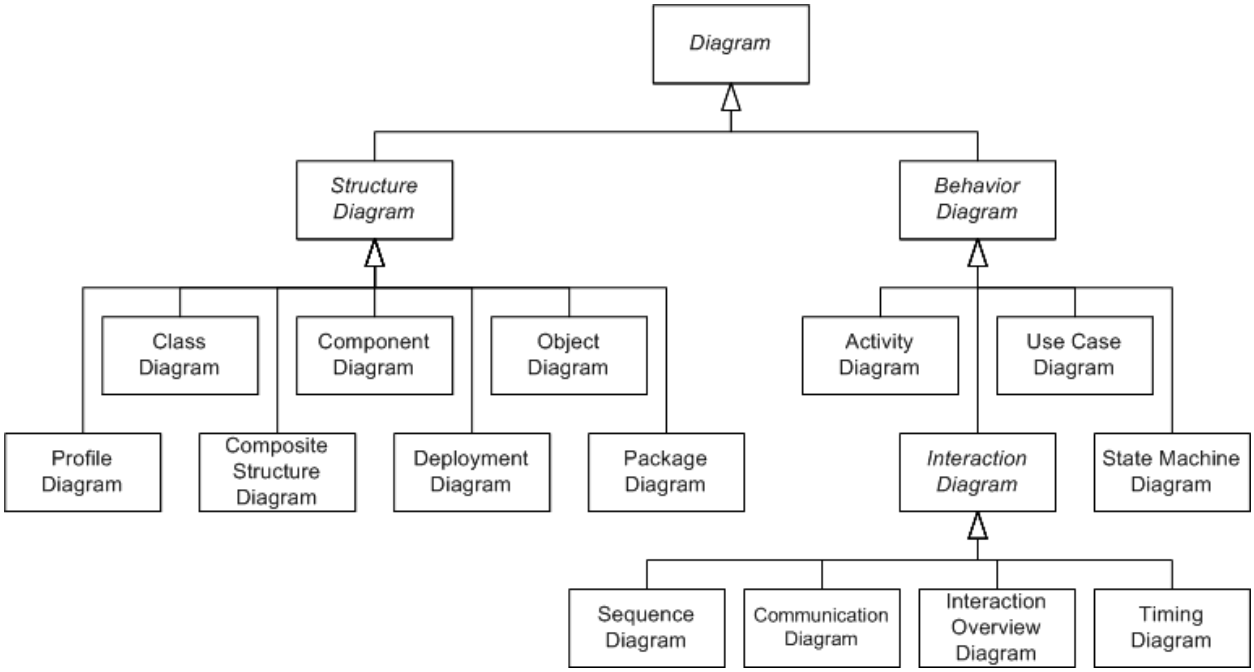

Рисунок 3.12 − Структуру діаграм UML 2.0

Існують різні точки зору на побудову діаграм класів в залежності від цілей їх застосування:

− концептуальна точка зору − діаграма класів описує модель предметної області, в ній присутні тільки класи прикладних об'єктів;

− точка зору специфікації − діаграма класів застосовується при проектуванні інформаційних систем;

− точка зору реалізації − діаграма класів містить класи, використовувані безпосередньо в програмному коді (при використанні об'єктно-орієнтованих мов програмування).

StarUML − програмний інструмент моделювання (рис. 3.13), який підтримує UML. StarUML орієнтований на UML 2.0. Середовище розробки StarUML чудово налаштовується відповідно до потреб користувача і має вис-

окий ступінь розширюваності, особливо в області своїх функціональних можливостей) (рис. 3.13) [25][1)].

Особливості:

− приємні і наочний користувацький інтерфейс;

− повна підтримка стандарту UML 2.0;

− можливість встановлення доповнень;

− генерація діаграм по вихідному коду;

− генерація вихідного коду по діаграмах;

− платформонезалежне програмне забезпечення;

− є безкоштовною.

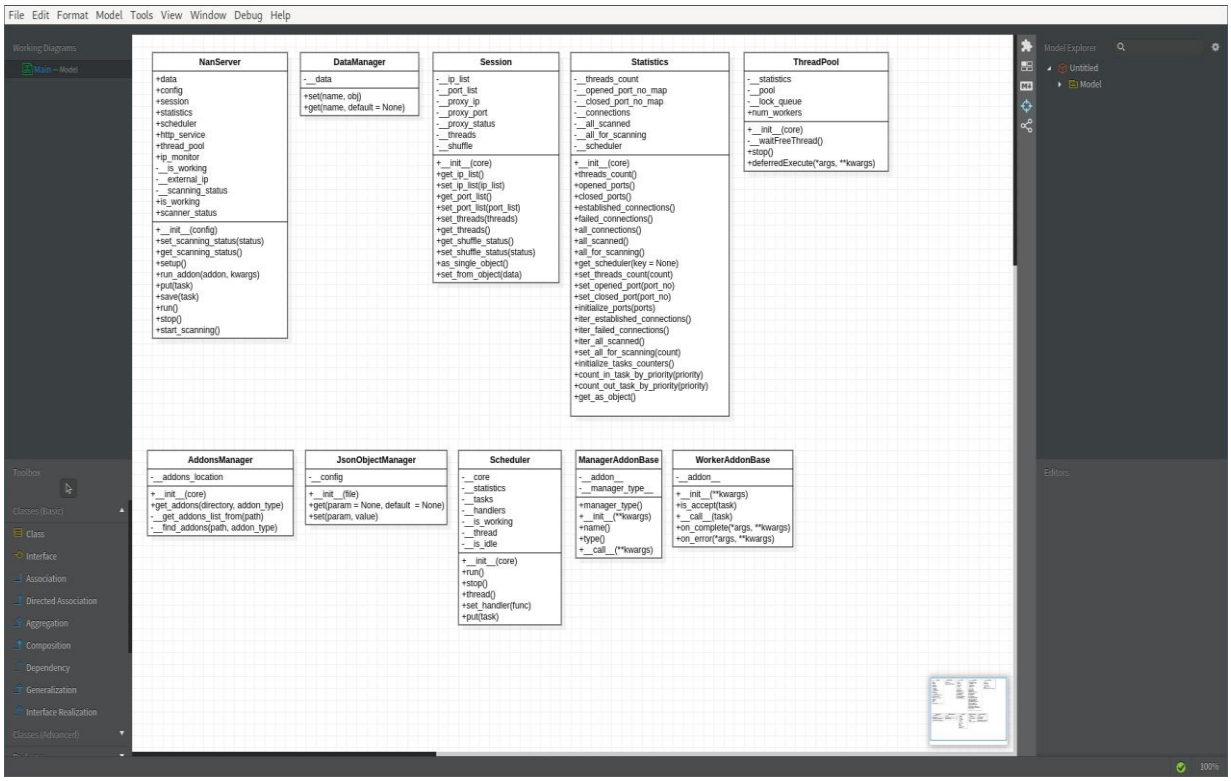

Рисунок 3.13 − Використання StarUML для моделювання структури проекту NaN

[1)] [25] Середовище розробки на мові UML, StarUML. URL: http://staruml.io/. (Дата звернення 22.11.2019).

Використання StarUML, одного з провідних програмних інструментів моделювання, гарантує досягнення максимальної продуктивності і якості розроблюваних програмних проектів.

StarUML надає максимальну ступінь адаптації середовища розробки користувача, пропонуючи налаштування параметрів, які можуть впливати на методологію розробки програмного забезпечення, проектну платформу і мову.

3.2 Технології застосовані в розробці ПЗ

WebSocket − протокол зв'язку поверх TCP-з'єднання, призначений для обміну повідомленнями між браузером і веб-сервером у режимі реального часу (рис. 3.14) [26][1)].

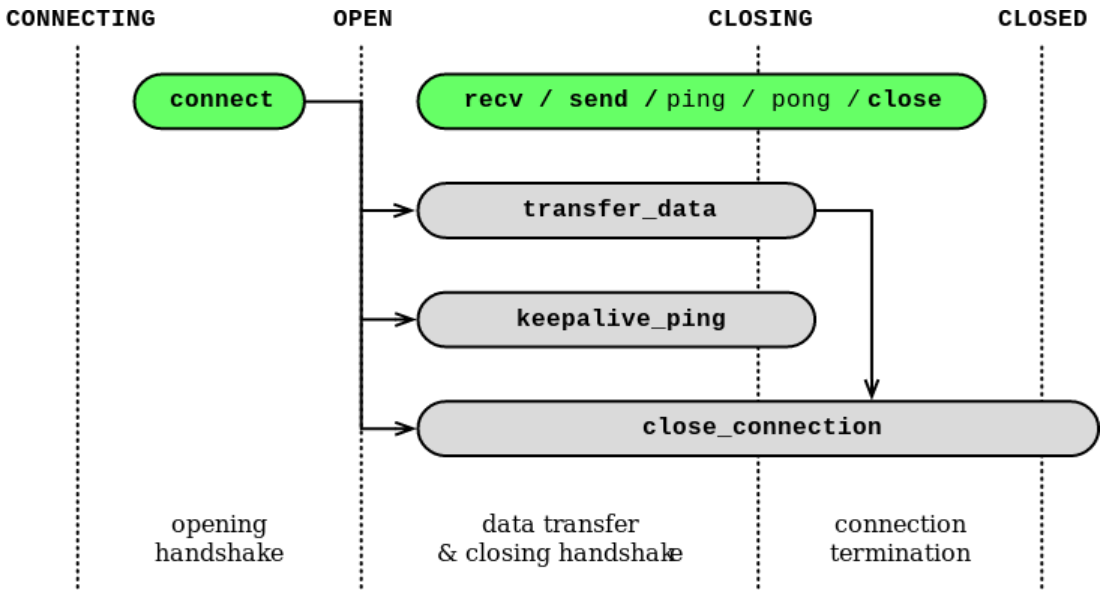

Рисунок 3.14 − Робота протоколу веб-сокетів в на діаграмі послідовності

1) [26] WebSocket, протокол зв'язку поверх TCP-з'єднання. URL: https://en.wikipedia.org/wiki/WebSocket. (Дата звернення 20.11.2019).

В даний час в W3C здійснюється стандартизація API Web Sockets. Чорновий варіант стандарту цього протоколу затверджений IETF.

WebSocket розроблений для втілення в веб-браузерах і веб-серверах, але він може бути використаний для будь-якого клієнтського або серверного додатка. Протокол WebSocket − це незалежний протокол, заснований на прот-околі TCP. Він робить можливим більш тісний контакт між браузером і веб-сайтом, сприяючи поширенню інтерактивного вмісту та створення додатків реального часу.

HTML5 − мова для структурування та подання вмісту всесвітньої павутини. Це п'ята версія HTML [27] [1]). Хоча стандарт був завершений (рекомендована версія до використання) тільки в 2014 році (попередня, четверта, версія опублікована в 1999 році), вже з 2013 року браузерами оперативно здійснювалася підтримка, а розробниками − використання робочого стандарту (англ. HTML Living Standard). Мета розробки HTML5 − поліпшення рівня підтримки мультимедіа-технологій з одночасним збереженням зворотної сумісності, зрозумілості коду для людини і простоти аналізу для парсерів (рис. 3.15).

Рисунок 3.15 − Логотип нового стандарту HTML, HTML5

18 січня 2011 року W3C ввів логотип, щоб представити використання або додати інтерес до HTML5. На відміну від інших знаків, випущених

[1)] [27] Мова розмітки, HTML5. URL: https://en.wikipedia.org/wiki/HTML5. (Дата звернення 20.11.2019).

раніше компанією W3C, він не має на увазі відповідність певному стандарту. З 1 квітня 2011 року цей логотип вважається офіційним.

Під час першого показу його публіці, W3C оголосив логотип HTML5 як символ «універсальної візуальної ідентифікації даних для широкого набору відкритих Веб-технологій, включаючи HTML, CSS, SVG, WOFF і інші». Деякі захисники веб-стандартів, включаючи і The Web Standards Project, розкритикували це визначення HTML5 як узагальнене і розмите поняття. Трьома днями пізніше W3C відповів на відгук спільноти і змінив визначення логотипу, прибравши перерахування пов'язаних технологій. Потім W3C заявив, що логотип «представляє HTML5, наріжний камінь для сучасних Веб додатків»

CSS3 − специфікація CSS що активно розробляється. Являє собою формальну мову, реалізовану за допомогою мови розмітки. Наймасштабніша редакція в порівнянні з CSS1, CSS2 та CSS2.1. Головною особливістю CSS3 є можливість створювати анімовані елементи без використання JavaScript, підтримка лінійних і радіальних градієнтів, тіней, згладжування та інше (рис. 3.16) [28][1)].

Рисунок 3.16 − Логотип стандарту CSS3

Переважно використовується як засіб опису та оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мов розмітки HTML і

1) [28] Cascading Style Sheets. URL: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. (Дата звернення 20.11.2019).

XHTML, але може також застосовуватися до будь-яких XML-документах, наприклад, до SVG або XUL.

На відміну від попередніх версій специфікація розбита на модулі, розробка і розвиток яких йде незалежно. CSS3 заснований на CSS2.1, доповнює існуючі властивості і значення і додає нові.

Нововведення, починаючи з малих, начебто закруглених кутів блоків, закінчуючи трансформацією (анімацією) і, можливо, введенням змінних.

3.3 Сервіси застосовувані в розробці і роботі ПО

Індекс пакетів Python, скорочений як PyPI, є офіційним стороннім програмним репозиторієм для Python. Деякі менеджери пакетів, включаючи pip, використовують PyPI як джерело за замовчуванням для пакетів та їхніх залежностей [29][1)]. Понад 113000 пакетів Python можна отримати через PyPI. PyPI в основному розміщує пакети Python у вигляді архівів, які називаються (вихідні дистрибутиви) або попередньо скомпільовані "wheel". PyPI як індекс дозволяє користувачам шукати пакети за ключовими словами або фільтрами за їх метаданими, такими як ліцензія на безкоштовне програмне забезпечення або сумісність із POSIX. Єдиний запис на PyPI може зберігати пакет та його метадані, попередніх релізів пакета, попередньо скомпільованих "wheels", а також різних форм для різних операційних систем та версій Python. Каталог PyPI служить джерелом інформації для систем оновлення програмного забезпечення на Python, що неминуче ставить питання забезпечення інформаційної безпеки. Система управління бібліотеками Python дозволяє не довіряти один одному розробникам робити свої біблі-отеки доступними користувачам.

В даний час в PyPI відсутній механізм захисту виявлення оновлень (англ. Update discovery) і процесу установки, але існує The Update Framework −

---

1) [29] PyPI · The Python Package Index. URL: https://pypi.org/. (Дата звернення 08.11.2019).

прототип каркаса для безпечної роботи з PyPI за допомогою утиліти easy_install. Модуль розповсюдження Python (distutils) Модуль Python вперше був доданий до стандартної бібліотеки Python у версії 1.6.1, у вересні 2000 року та у версії 2.0 у жовтні 2000 року, через дев'ять років після першого випуску python в лютому 1991 р., з метою спрощення процесу встановлення сторонніх пакетів Python (рис. 3.17).

Рисунок 3.17 − Логотип Python Package Index

Однак distutils надавали лише інструменти для упаковки коду Python, і не більше. Він зміг збирати та поширювати метадані, але не використовував їх для інших цілей. У Python все ще бракувало централізованого каталогу для пакетів в Інтернеті. PEP 241, пропозиція щодо стандартизації метаданих для індексів, була доопрацьована в березні 2001 року. Пропозиція створити комплексний централізований каталог, розміщений у домені python.org, була згодом доопрацьована у листопаді 2002 року. 16 квітня 2018 року весь трафік PyPI почав обслуговуватися більш сучасною платформою веб-сайтів, Складом, а старий веб-сайт був вимкнений наприкінці цього місяця. Усі існуючі пакети були перенесені на нову платформу та їх історія збережена.

VirusTotal − безкоштовна служба, що здійснює аналіз підозрілих файлів і посилань (URL) на предмет виявлення вірусів, хробаків, троянів і всіляких шкідливих програм. VirusTotal нагороджений американським виданням PC World Magazine як один з 100 кращих товарів 2007 року. Має локалізацію на

багато мов світу, включаючи українську. Сервіс є повністю безкоштовним. Результати перевірок файлів сервісом не залежать від якогось одного виробника антивірусів.

У VirusTotal використовується кілька десятків антивірусних систем, що дозволять робити більш надійні висновки про небезпеку файлу (якщо всі або більшість антивірусів вважають файл небезпечним), в порівнянні з якимось одним продуктом, може допомогти передбачити можливі помилкові спрацьовування якогось одного антивіруса (або антивірусів) або, навпаки, не спрацьовування на свіжу загрозу, можливо, вже внесену іншими виробниками в свої бази (рис. 3.18) [30][1)].

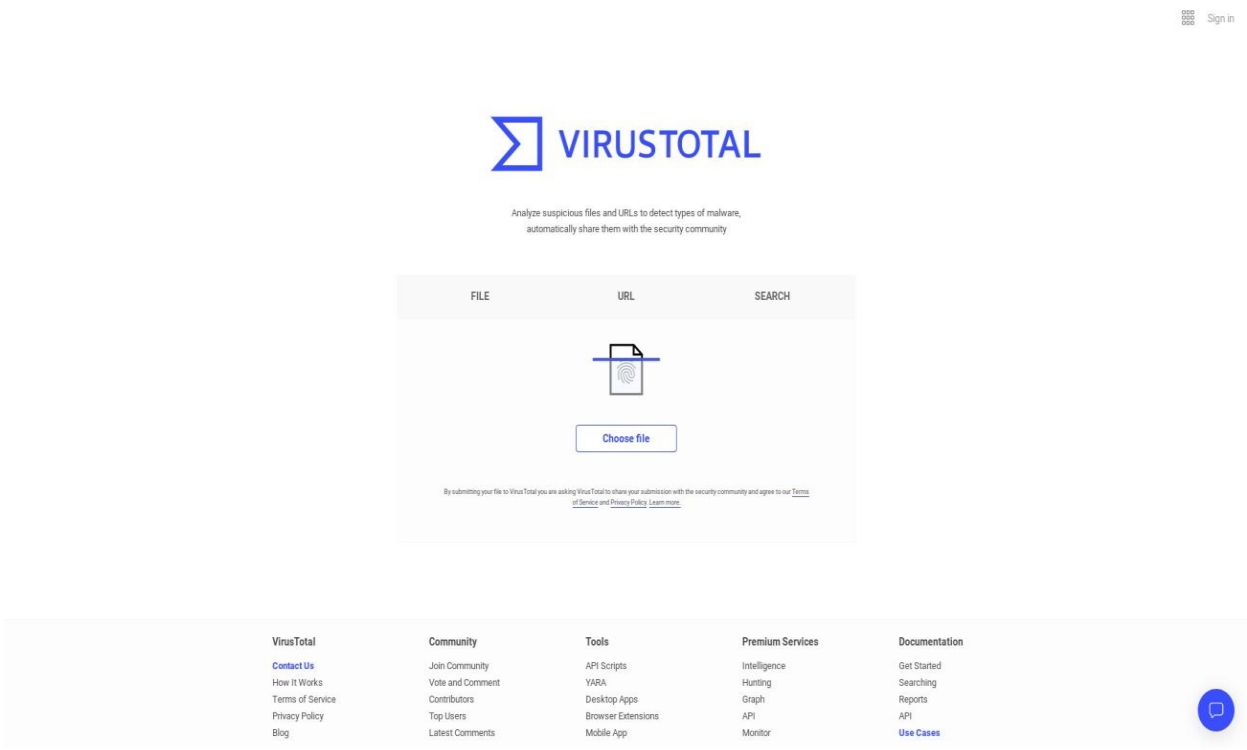

Рисунок 3.18 − Головна сторінка сервісу VirusTotal

У компаній-розробників антивірусного програмного забезпечення існують власні класифікації та номенклатури шкідливих програм, тому при

1) [30] Сервіс перевірки підозрілих файли файлів, VirusTotal. URL: https://www.virustotal.com/. (Дата звернення 08.11.2019).

перевірці файлу антивіруси на Virustotal можуть видавати різні результати, наприклад, одні антивіруси визначать файл небезпечним, а інші − безпечним.

Антивіруси на Virustotal не гарантують 100% відсутності шкідливого коду у файлі. Всі використовувані сервісом антивірусні бази постійно оновлюються. У результатах перевірки вказуються дати останніх оновлень всіх баз.

Після завантаження файлу система обчислює його хеш і при наявності результатів перевірки файлу з таким же хешем пропонує або переглянути останній аналіз (вказавши дату першої і останньої перевірки), або повторити аналіз. Сервіс постійно розвивається, постійно підключаються нові сканери (антивіруси і антітрояни). VirusTotal відсилає підозрілі файли виробникам антивірусів на аналіз.

Сервіс не замінює антивірус на локальному комп'ютері, оскільки перевіряються тільки окремі файли і окремі URL-адреси на вимогу. Сервіс не забезпечує постійного захисту на комп'ютері користувача і є доповненням до встановленого антивірусу. Хоча сервіс і використовує кілька антивірусних движків, результат роботи антивірусів не гарантує нешкідливість файлу або URL-посилання. Максимальний розмір файлу обмежений 550 мегабайтами.

VirusTotal для браузерів − Доступні кілька розширень браузера, такі як VTzilla для Mozilla Firefox, VTchromizer для Google Chrome та VTexplorer для Internet Explorer. Вони дозволяють користувачеві завантажувати файли безпосередньо з веб-програми VirusTotal перед їх зберіганням у комп’ютері, а також сканувати URL-адреси.

VirusTotal для мобільних пристроїв − У сервісі також пропонується додаток для Android, в якому використовується публічний API для пошуку будь-якої встановленої програми для раніше відсканованих програм VirusTotal та показує її статус. Можна подати будь-яку програму, яку раніше не сканували, але необхідно вказати ключ API та застосувати інші обмеження щодо використання публічного API.

Public API − VirusTotal надає як безкоштовний сервіс публічний API, який дозволяє автоматизувати деякі його онлайн-функції, такі як "завантажувати та сканувати файли, подавати та сканувати URL-адреси, отримувати доступ до готових звітів про сканування та робити автоматичні коментарі щодо URL-адрес та зразків". Деякі обмеження застосовуються до запитів, зроблених через загальнодоступний API, наприклад, вимагає індивідуального ключа API, вільно отриманого при реєстрації в режимі он-лайн, черги сканування з низьким пріоритетом, обмеженої кількості запитів у часові рамки тощо.

Virustotal не призначений для порівняння антивірусів з наступних причин:

− антивірусні движки, які використовуються на Virustotal, є консольними версіями, тому в залежності від продукту вони не будуть вести себе так само, як їх аналоги для настільних комп'ютерів. Наприклад, версія антивіруса для настільного комп'ютера може також використовувати проактивний захист і брандмауер, які підвищують ймовірність виявлення загроз;

− в консольних версіях антивірусів на Virustotal евристичний аналіз може бути більш агресивним і параноїдальним в порівнянні з антивірусами для настільних комп'ютерів, тому на Virustotal помилкових спрацьовувань може бути більше.

Недоліки сервісу:

− Статистика на сайті показується «як є», немає ні засобів, ні можливостей для її аналізу.

− API має обмеження щодо ліміту запитів та розміру файлу, як у безкоштовній версії, так і в корпоративній (платній) версії.

GitHub − найбільший веб-сервіс для хостингу IT-проектів і їх спільної розробки. Веб-сервіс заснований на системі контролю версій Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome). Сервіс безкоштовний для проектів з відкритим вихідним кодом і

(з 2019 роки) невеликих приватних проектів, надаючи їм всі можливості (включаючи SSL), а для великих корпоративних проектів пропонуються різні платні тарифні плани (рис. 3.19) [31][1)].

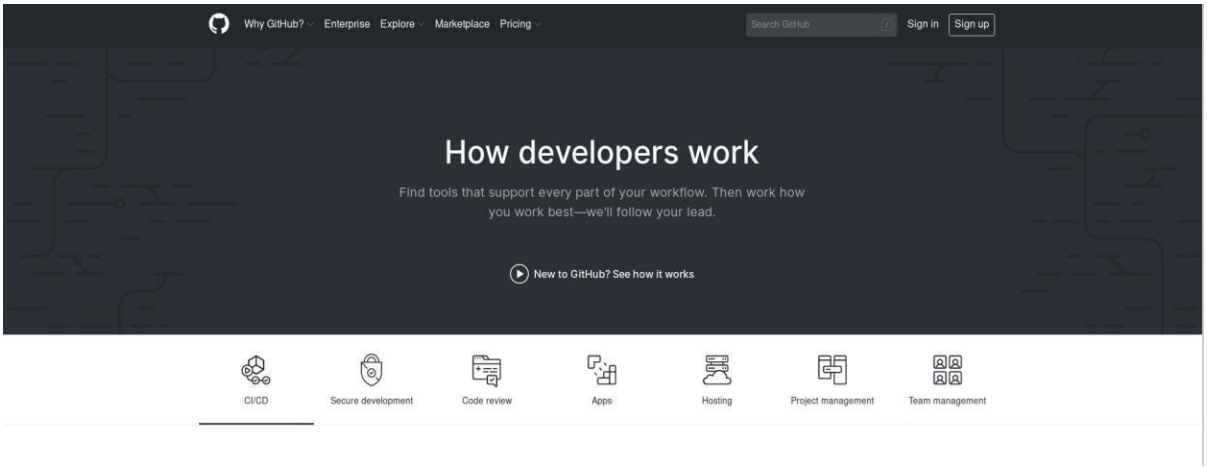

Рисунок 3.19 − Головна сторінка проекту GitHub

Слоган сервісу − «Social Coding» − на українську можна перекласти як «Пишемо код разом». На футболках ж друкують зовсім іншу фразу: «Fork you!» («Відгалужуйся!»). З одного боку, вона співзвучна з англомовним лайкою і натякає на неформальну атмосферу. З іншого, ці слова нагадують, що створювати нові Форк з Git можна легко і безболісно − традиційно, до створення гілок розробники проектів з відкритим вихідним кодом відносяться негативно − а також співзвучна назві однієї з можливостей GitHub - черги форків.

Талісманом GitHub обраний осьмікот (англ. Octocat), який, всупереч поширеній помилці, не має відношення до короткометражки «Octocat Adventure», а просто був знайдений Томом Престон-Вернером на сервісі iStock. 4 червня 2018 року Microsoft купила GitHub за 7,5 млрд доларів.

Творці сайту називають GitHub «соціальною мережею для розроб-

[1)] [31] The world's leading software development platform · GitHub. URL: https://github.com/. (Дата звернення 12.11.2019).

ників». Крім розміщення коду, учасники можуть спілкуватися, коментувати правки один одного, а також стежити за новинами знайомих.

За допомогою широких можливостей Git програмісти можуть об'єднувати свої репозиторії − GitHub пропонує зручний інтерфейс для цього і може відображати внесок кожного учасника у вигляді дерева.

Для проектів є особисті сторінки, невеликі Вікі і система стеження за вадами. Прямо на сайті можна переглянути файли проектів з підсвічуванням синтаксису для більшості мов програмування. Можна створювати приватні репозиторії, які будуть видні тільки вам і вибраним вами людям.

Раніше можливість створювати приватні репозиторії була платною. Є можливість прямого додавання нових файлів в свій репозиторій через веб-інтерфейс сервісу. Код проектів можна не тільки скопіювати через Git, а й скачати у вигляді звичайних архівів з сайту. Крім Git, сервіс підтримує отримання і редагування коду через SVN і Mercurial.

Також на сайті є pastebin-сервіс gist.github.com для швидкої публікації фрагментів коду.

Microsoft стала важливим користувачем GitHub, використовуючи його для розміщення проектів з відкритим кодом та інструментів розробки, таких як Chakra Core, PowerShell, Visual Studio Code та Windows Terminal, і підтримала інші проекти з відкритим кодом, такі як Linux, і розробила віртуальну файлову систему для Git (VFS for Git; раніше віртуальна файлова система Git або GVFS) − розширення Git для управління масштабними сховищами (і його прийняв GitHub).

Виникли занепокоєння розробників Кайла Сімпсона, тренера та автора JavaScript, та Рафаеля Лагуна, генерального директора Open-Xchange з приводу придбання Microsoft, посилаючись на занепокоєння щодо поводження з Microsoft попередніми придбаннями, такими як мобільний бізнес Nokia або Skype.

Деякі сприймали це як кульмінацію нещодавніх змін Microsoft у стратегії бізнесу під керівництвом компанії Сатья Наделла, яка розглядала

більшу увагу на продажі послуг хмарних обчислень як своєї основної сфери бізнесу, поряд із розвитком та внеском у відкритий код програмне забезпечення (наприклад, Linux), на відміну від операційної системи Microsoft Windows. Harvard Business Review стверджував, що Microsoft має намір придбати GitHub, щоб отримати доступ до своєї бази даних, тому він може бути використаний як лідер збитків для заохочення використання інших продуктів і послуг для розробки.

Занепокоєння щодо продажу посилило інтерес у конкурентів: Bitbucket (належить Atlassian), GitLab (комерційний продукт з відкритим кодом, який також працює з розміщеною сервісною версією) та SourceForge (належить компанії BIZX, LLC) повідомили, що вони побачили сплески нових користувачів, які мають намір перенести проекти з GitHub до відповідних служб. Організаційна структура − GitHub, Inc. спочатку була плоскою організацією без середніх менеджерів; Іншими словами, "кожен − менеджер". Співробітники могли обирати роботу над проектами, які їх зацікавили (відкритий розподіл), але зарплати визначав керівник. У 2014 році GitHub, Inc. представила шар середнього управління.

Фінанси − GitHub.com був початковим бізнесом, який в перші роки давав достатній дохід, щоб фінансуватись виключно трьома його засновниками та почати брати працівників. У липні 2012 року, через чотири роки після заснування компанії, Андреєссен Горовіц вклав 100 млн. Доларів у венчурний капітал. У липні 2015 року GitHub зібрав ще 250 мільйонів доларів венчурного капіталу в раунді серії B. Інвесторами були Sequoia Capital, Andreessen Horowitz, Thrive Capital та інші фонди венчурного капіталу. Станом на серпень 2016 року GitHub заробляв 140 мільйонів доларів щорічного періодичного доходу.

Талісман GitHub − це антропоморфізований "восьминог" з п'ятьма руками, схожими на восьминога. Octocat, талісман GitHub. Персонаж був створений графічним дизайнером Саймоном Окслі як графічне мистецтво для

продажу на iStock, веб-сайт, який дозволяє дизайнерам продавати цифрові зображення, безоплатні роялті (рис. 3.20).

GitHub зацікавився роботою Окслі після того, як Twitter вибрав птаха, якого він створив для власного логотипу. Ілюстрація, яку вибрав GitHub, була персонажем, якого Окслі назвав Октопосом. Оскільки GitHub хотів, щоб у логотипа Octopuss був логотип, вони домовилися з Oxley придбати ексклюзивні права на зображення. GitHub перейменував Octopuss на Octocat, і торгова марка символу разом з новим іменем. Пізніше GitHub найняв ілюстратора Кемерона Макфі, щоб адаптувати Octocat для різних цілей на веб-сайті та рекламних матеріалах; McEfee та різні користувачі GitHub з тих пір створили сотні варіацій персонажа, які доступні в The Octodex.

Рисунок 3.20 − Octocat, талісман GitHub

4 РОЗРОБКА РІШЕННЯ ДЛЯ МОНІТОРИНГУ МЕРЕЖІ

4.1 Специфікація функціональних та нефункціональних вимог

При розробці проекту були поставлені наступні вимоги, які були складені після ретельного аналізу сильних і слабких сторін розглянутих аналогів, розроблюваного програмного забезпечення.

4.1.1 Нефункціональні вимоги

− нескладний користувацький інтерфейс;

− можливість роботи на операційних системах без графічного інтерфейсу;

− використання сучасних інструментів розробки і мов програмування.

4.1.2 Функціональні вимоги

− клієнт-серверна архітектура, яка відокремлює користувацький інтерфейс від логіки;

− підтримка програмних модулів;

− підтримка розширення функціональності за допомогою плагінів;

− підтримка логування роботи програми, для виявлення помилок в роботі додаткових плагінів і модулів;

− підтримка багато потокового режиму роботи;

− підтримка конфігурацій, що зберігаються між сеансами сканування;

− знаходження IoT пристроїв і сервісів в заданій мережі;

− можливість тестування на слабкість парольної аутентифікації;

− можливість тестування на уразливості в мережевих сервісах і сервісах мережевих IoT пристроїв;

− можливість відновлення працездатності пошкоджених шкідливим ПЗ

мережевих IoT пристроїв і сервісів;

− генерація простих звітів по вразливих пристроям і типам їх вразливостей;

− збереження результатів сканування.

4.2 Проектування частин рішення на мові UML

Композиція головних модулів сканера, створених за допомогою середовища розробки UML діаграм, StarUML (рис. 4.1).

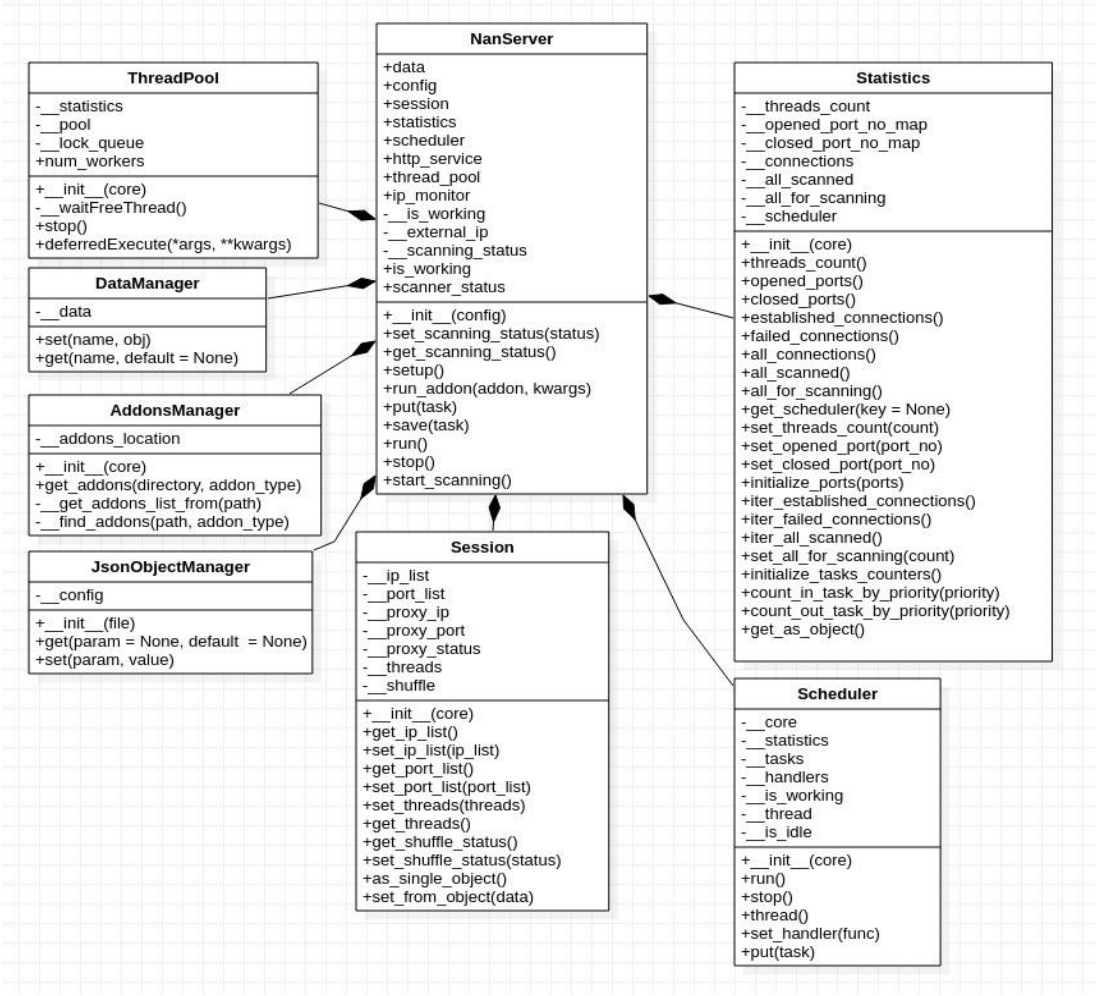

Рисунок 4.1 − UML діаграма об'єднання головних модулів мережевого сканера

4.3 Ключові компоненти рішення на мові Python

Як описано вище на UML діаграмах, що розробляється рішення складається з безлічі частин які об'єднуються принципом композиції в єдину систему здатну обробляти різні даних одержувані від інших частин системи.

4.3.1 Програмна реалізація ядра сканера

Точка входу в додаток знаходиться в класі NanServer, ця частина є ядром, ядро сканера це та частина програмного забезпечення через якові проходять всі дані, ця частина Керує всім процесом сканування розподіляючи завдання між потоками в яких працюють модулі.

Ініціалізація ядра сканера відбувається в його конструкторі NanServer.__init__, і починається з завантаження всіх модулів і даних необхідних для роботи сканера. Першим модулем завантажується менеджер даних який зберігає в собі різні дані необхідні для роботи мережевого сканера, а так само дані отримані в процесі сканування.

```
class NanServer(object):
    def __init__(self, config: str):
        self.data = DataManager()
        self.data.set('results', list())
        self.data.set('not-handled', list())
```

Завантаження шаблонів пошуку в вигляді регулярних виразів, що зберігаються в списках в форматі JSON.

```
        self.data.set("servers-regex", JsonObjectManager("servers-
regex.json"))
        self.data.set("titles-regex", JsonObjectManager("titles-
regex.json"))
        self.data.set("routers-regex", JsonObjectManager("routers-
regex.json"))
```

Після завантаження менеджера даних, завантажується конфігурація знаходиться в форматі JSON, за допомогою класу JsonObjectManager.

```
self.config = JsonObjectManager(config)
```

Менеджер сесії – Він керує конфігурацією поточної сесії сканування і є прошарком яка дозволяє обмінюватися конфігураціями внутрішніх систем з інтерфейсом користувача.

```
self.session = Session(self)
```

Менеджер статистики – він зберігає в собі дані про прогрес сканування, знайдених пристроях, портах на яких вони були знайдені, встановлених з'єднаннях і про кількостях завдань які на даний момент виконує планувальник а так само про пріоритети цих завдань.

```
self.statistics = Statistics(self)
```

Планувальник – Управляє завданнями, сортує їх відповідно до пріоритетів, призначає виконує доповнення відповідно до типу завдання і виконує запуск в виділеному потоці для роботи сканера в багатопотоковому режимі.

```
self.scheduler = Scheduler(self)
```

Сервер HTTP API – призначений для взаємодії з клієнтами по протоколу HTTP, так само підтримує технологію WebSocket, це дозволяє бачити користувачеві результати його дій і дій сканера майже миттєво.

```
self.http_service = HttpApi(self)
```

Менеджер потоків – використовується планувальником для паралельного виконання різних завдань, підтримувати запуск завдань в асинхронному режимі що дозволяє ефективно використовувати потоки для виконання поставлених завдань.

```
self.thread_pool = ThreadPool(self)
```

Установка прапорів управління - Ці прапори керують роботою сканера дозволяючи сканера знаходиться в різних режимах роботи. Це дозволяє відстежувати роботу сканера і надавати динячі про це користувачеві.

```
# за замовчуванням сканер знаходиться в режимі очікування.
self._scanning_status = ScanningStatus.STANDBY
```

Прапор завершення роботи сканера - він застосовується для завершення роботи сканера, що в свою чергу викликає зупинку всіх підсистем: менеджера потоків, планувальника і HTTP API сервера.

```
self._is_working = True
```

Далі відбувається виклик методу NanServer.run який виконує метод NanServer.setup і запускає головний цикл очікування завершення роботи програми, а так само після завершення роботи повинен зупинити всі виділені потоки.

```
def run(self):
    self.setup()
    logger.warning(f"Press '^C' to stop.")
    while self.__is_working:
        try:
            sleep(1)
        except KeyboardInterrupt:
```

Завершення роботи модулів по перериванню користувачем, ці модулі необхідно завершити викликавши методи завершення роботи, що в свою чергу зупинить виділені потоки всередині яких вони запущені.

```
        self._is_working = False
        # зупинка менеджера потоків
        self.thread_pool.stop()
        # зупинка планувальника
        self.scheduler.stop()
        # зупинка HTTP API сервера
        self.http_service.stop()
        # вихід
```

Виклик NanServer.setup − завантажує всі доповнення успадковані від класу ManagerAddonBase і встановлює їх в планувальник як оброблювачів у яких є типи завдань для яких вони призначені, далі за допомогою планувальника завдання відповідно до типу будуть розподіляться між доповненнями. Так само NanServer.setup запускає всі модулі які працюють в неблокуючих режимі будучи виділеними потоками.

```
def setup(self):
    # завантаження доповнень
    addons = AddonsManager(self).get_addons("managers",
ManagerAddonBase)
    # встановлення додатків в якості оброблювачів в планувальник
    for _, addon in addons.items():
        self.scheduler.setHandler(addon(self))
    # запуск потоку планувальника
    self.scheduler.run()
    # запуск потоку HTTP API сервера
    self.http_service.run()
```

Так само клас NanServer містить і інші методи без яких його повноцінна робота буде неможлива але докладний опис всього коду знаход-

иться в проекті зробить мою магістерську роботу занадто об'ємною, з цієї причини деякі дрібні подробиці опущені.

4.3.2 Програмна реалізація планувальника

Планувальник – один з основних компонентів без котрих коректна робота сканера неможлива він керує завданнями, сортує ці завдання відповідно до їх пріоритетів, призначає виконує доповнення відповідно до типу завдання і виконує запуск в виділеному потоці який йому надає клас ThreadPool, цей клас забезпечує роботу сканера в багатопотоковому режимі.

Ініціалізація планувальника – ініціалізація виконується в конструкторі Scheduler._init__, куди в свою чергу передається посилання на ядро сканера для надання зворотного зв'язку планувальника з ядром.

```
class Scheduler(object):
    def __init__(self, core):
        # Посилання на ядро сканера
        self._core = core
```

Установка планувальником посилання на модуль статистики, для надання статистики про завдання оброблені планувальником, до користувача.

```
        self._statistics: Statistics = core.statistics
```

Створення черги з пріоритетами, для зберігання завдання в відсортованому вигляді згідно їх пріоритетам

```
        self._tasks = PriorityQueue()
        # Обробники завдань, доповнення менеджери
        self._handlers = {}
```

Прапор завершення роботи, управляти виділений потоком планувальника.

```
self._is_working = True
```

Створення виділеного потоку планувальника дозволяє працювати сканера в неблокуючому режимі

```
self._thread = Thread(target=self.thread, name="scheduler")
```

Метод Scheduler.run – виконує запуск виділеного потоку планувальника

```
def run(self):
    self.__thread.start()
```

Метод Scheduler.stop – завершує роботу планувальника встановлюючи прапор завершення роботи в потрібне значення, а так само смакота виділений потік з головним потоком.

```
def stop(self):
    # Зупинка виділеного потоку
    self._is_working = False
    # Єднання з головним потоком
    self.__thread.join()
```

Метод Scheduler.thread – являє собою метод який буде виконуватися в виділеному потоці, виконуючи завдання які надходять в чергу.

```
def thread(self):
    while True:
        # Зупинка циклу очікування по прапору
        if not self.__is_working:
```

```
            break
```

Гілка очікування нових завдань приходять в планувальник, згідно їх пріоритетам.

```
        if self.__tasks.empty():
            sleep(1)
            continue
```

Частина планувальника, яка займається обробкою завдання, якщо в черзі є завдання. При виконанні завдання воно передається по типу в потрібне доповнення "менеджер"

```
        task = self.__tasks.get(block=False, timeout=2)
        # Підрахунок статистики виконаних завдань
        self._statistics.count_out_task_by_priority(task.priority)
        # Передача управління доповненню менеджеру
        self.__handlers[task.type](task)
```

Метод Scheduler.setHandler – додає в планувальник нове доповнення яке буде застосовуватися для обробки завдань. Додатки передаються у вигляді функторів.

```
def setHandler(self, func: object):
    self._handlers[func.type] = func
```

Метод Scheduler.put – додає завдання в пріорітизовану чергу, а так само збирає статистику про отримані завданнях.

```
def put(self, task: Task):
    self._statistics.count_in_task_by_priority(task.priority)
    self.__tasks.put(task)
```

4.3.3 Програмна реалізація менеджера потоків

Менеджер потоків клас ThreadPool призначений для роботи спільно з планувальником для паралельного виконання різних завдань, він так-же управляє кількістю одноразово запущених потоків за допомогою додаткової черги котра застосовується для блокування менеджера потоків при досягненні граничної кількості потоків.

Ініціалізація менеджера потоків – ініціалізація виконується в конструкторі ThreadPool._init_, як і в інших модулях в цьому теж є посилання на модуль статистики.

```
class ThreadPool(object):
    def __init__(self, core):
        # Посилання на модуль статистики
        self._statistics: Statistics = core.statistics
        # кількість робочих потоків, отримані з файлу конфігурації
        num_workers: int = core.config.get("nios.workers")
        # пулл потоків
        self._pool: Pool = Pool(num_workers)
        # Черга блокування потоків
        self._lock_queue = Queue(num_workers)
```

Метод ThreadPool.__waitFreeThread – призначений для очікування вільного потоку виконання.

```
def __waitFreeThread(self):
    # Перевірка заповнення черги блокування
    while self.__lock_queue.full():
        # Отримання результату виконуваного потоку
        item: ApplyResult = self._lock_queue.get()
        # Перевірка готовності результату
        if not item.ready():
            # Очікування якщо не готовий
            item.wait(1)
        # Якщо після очікування не готовий
```

```
        if not item.ready():
            # повернути в чергу
            self.__lock_queue.put(item)
```

Метод ThreadPool.deferredExecute – призначений для відкладення запуску завдання в вільного потоку виконання якщо такий є інакше очікувати за допомогою методу ThreadPool._waitFreeThread наступного вільного потоку.

```
def deferredExecute(self, *args, **kwargs):
    # Метод очікування вільного потоку
    self.__waitFreeThread()
    # Запуск виконання в виділеному потоці
    result = self.__pool.apply_async(*args, **kwargs)
    # Додавання результату в чергу блокування
    self.__lock_queue.put(result)
    # Отримання статистики про кількість виконуваних завдань
    self.__statistics.set_threads_count(self.__lock_queue.qsize())
```

Метод ThreadPool.stop – призначений для зупинки менеджера потоків, завершення всіх потоків і об'єднання їх з головним потоком.

```
def stop(self):
    # Закриття пулу потоків
    self.__pool.close()
    # Зупинка пулу потоків
    self.__pool.terminate()
    # Об'єднання потоків з головним потоком
    self.__pool.join()
```

4.3.4 Реалізація механізму доповнень

Багато програм підтримують так звані плагіни (доповнення, розширення і т.п.), за допомогою яких можна розширювати функціональність програми. На Python робити програми, що підтримують плагіни особливо легко і приємно. Тому що з одного боку в якості плагіна можуть виступати повноцінні класи, а з іншого боку завдяки платформ мови Python плагіни так само залишаються платформозалежні. Давайте подивимося що потрібно зробити, щоб ваша програма теж підтримувала плагіни.

Для реалізації модульної архітектури були використані базові класи доповнень, це дозволить за допомогою механізмів імпорту модулів вбудованого в мову програмування Python виконати імпорт необхідних класів по їх батьківського класу, таким чином можна імпортувати тільки необхідні в якості модулів класи не зачіпаючи інші утилітарні та інші класи.

4.3.5 Програмна реалізація базового класу доповнення “робочого”

Доповнення робочі призначені для виконання в певних завдань всередині потоку, доповнення робочі створюються і передаються на виконання доповненнями менеджерами, які повинні знайти для доповнення відповідну задачу і запустити виконання робочого в виділеному потоці.

Кожне доповнення робочий складається з декількох обов'язкових методів і полів:

− поле − addon__, унікальний ідентифікатор використовується ідентифікації доповнення в словнику з завантаженими доповненнями;

− метод − init , конструктор який повинен бути перевизначений на додаток спадкоємця якщо це необхідно;

− метод − is_accept, метод який перевіряє чи може поточний додаток обробити завдання;

– метод – __call__, є перевантаженість оператором виклику, в доповненні в ньому відбувається основна робота;

– метод – on_complete, приймає результати які повертає метод___call__, перевіряє їх валідність і передає в чергу для обробки іншими доповненнями;

– метод – on_error, виводить в лог повідомлення про помилку якщо вона виникла в момент виконання методу___call__.

```
class WorkerAddonBase(object):
    __addon___= None
    def __init__(self, **kwargs):
        pass
    @staticmethod
    def is_accept(task):
        raise NotImplementedError
    def __call__(self, task):
        raise NotImplementedError
    def on_complete(self, *args, **kwargs):
        raise NotImplementedError
    def on_error(self, *args, **kwargs):
        logger.exception(f"{__name_}")
        print(*args, **kwargs)
```

4.3.6 Програмна реалізація базового класу доповнення "менеджера"

Доповнення менеджери призначені для пошуку відповідних завдань для доповнень робочих, а так само для запуску екземпляра доповнення робочого в виділеному потоці.

Кожне доповнення «Менеджер» складається з декількох обов'язкових методів і полів:

– поле – addon__, унікальний ідентифікатор використовується для ідентифікації доповнення в словнику з завантаженими доповненнями;

– метод – __call__, є перевантаженість оператором виклику, в доповненні в ньому відбувається основна робота;

– поле – __manager_type__, зберігає тип менеджера який використовується для прямої передачі завдань певним менеджерам;

– метод – manager_type, повертає значення приватного поля __manager_type__;

– метод – name, повертає ім'я типу менеджер;

– метод – type, повертає тип менеджера.

```
class ManagerAddonBase(object):
    __addon___= None
    __manager_type__ = ManagerType.UNKNOWN
    @classproperty
    def manager_type(self):
        return self.__manager_type___
    def __init__(self, **kwargs):
        pass
    @property
    def name(self):
        if self.__manager_type__ == ManagerType.UNKNOWN:
            raise AttributeError("__manager_type___is not set")
            return self.__manager_type_.name
    @property
    def type(self):
        if not self.__manager_type__:
        raise AttributeError("__manager_type___is not set")
        return self.__manager_type___
    def __call__(self, **kwargs):
        raise NotImplementedError
```

4.3.7 Програмна реалізація менеджера доповнень

Менеджер доповнень – цей клас призначений для завантаження доповнення з заданих директорій, після виконання процедури завантаження, він повертає словник містить всі завантажені модулі, де ключем є ім'я модуля а

значенням клас модуля з якого пізніше може бути створений екземпляр модуля.

```
class AddonsManager(object):
    def __init__(self, core):
```

Отримання директорії в якій зберігаються всі додатки, з раніше завантаженого конфігураційного файлу в форматі JSON.

```
        self._addons_location =
core.config.get("core.addons.location")
```

Отримання словника з усіма доповненнями з заданої директорії і заданого типу, виконується це за допомогою викликом методу для пошуку всіх доповнень відповідних типу.

```
     def get_addons(self, directory: str, addon_type):
        path = os.path.join(self._addons_location, directory)
        return AddonsManager._find_addons(path, addon_type)
```

Отримання всіх доповнень з заданої директорії, отримання всіх класів успадкування від класу переданого в addon_type та збереження класу доповнення в словник і повернення всіх знайдених доповнень.

```
    @staticmethod
    def __get_addons_list_from(path: str):
        addons = map(lambda x: os.path.splitext(x)[0],
os.listdir(path))
        addons = map(lambda x: os.path.join(path, x), addons)
        addons = map(lambda x: x.replace("/", '.'), addons)
        return list(addons)
    @staticmethod
    def __find_addons(path: str, addon_type):
        addons = AddonsManager.__get_addons_list_from(path)
```

```
founded_addons = {}
for addon in addons:
    # імпортування python модулів
    importlib.import_module(addon)
for addon in addon_type.__subclasses_():
    if addon.__module___not in addons:
        continue
       founded_addons[addon.__addon__] = addon
return founded_addons
```

4.4. Розробка діаграм варіантів використання

Основне призначення діаграми − опис функціональності і поведінки, що дозволяє замовнику, кінцевому користувачеві і розробнику спільно обговорювати проектовану або існуючу систему. Ці діаграми описують варіанти використання всіх доступних елементів управління системою сканування. Описані діаграми знаходяться на рис. 4.2 – рис. 4.5.

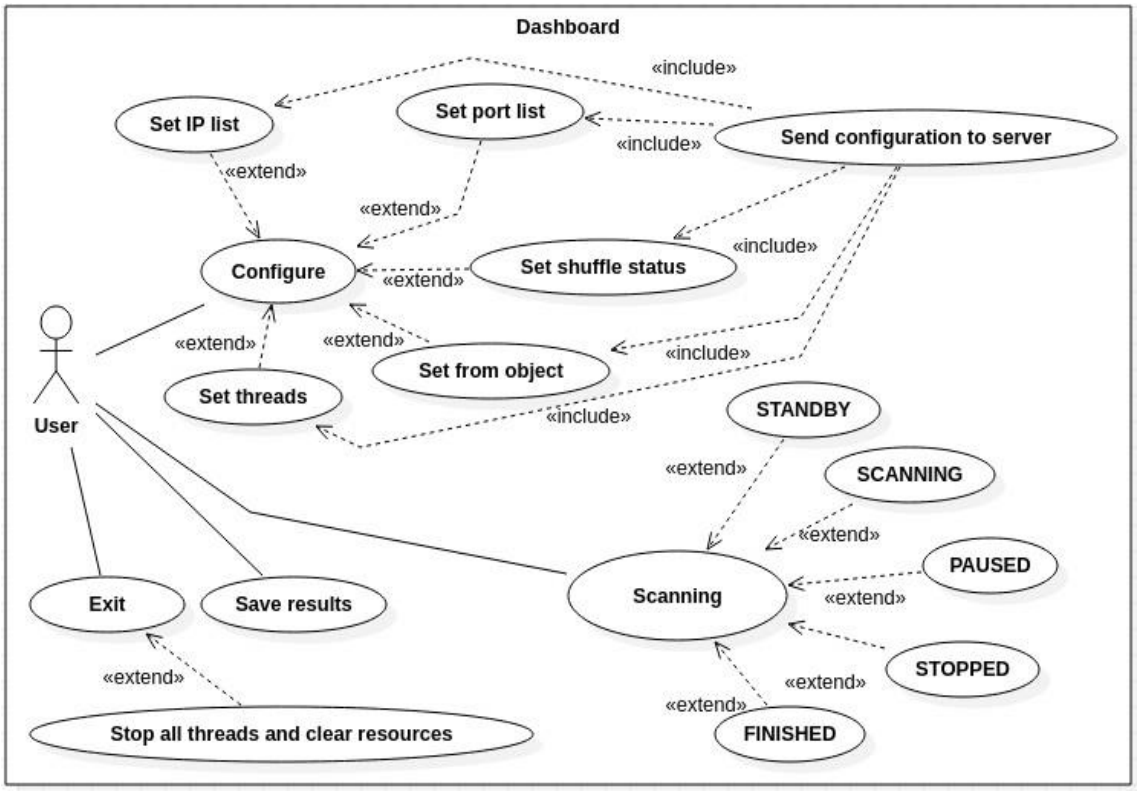

Рисунок 4.2 − Діаграма використання сторінки Dashboard.

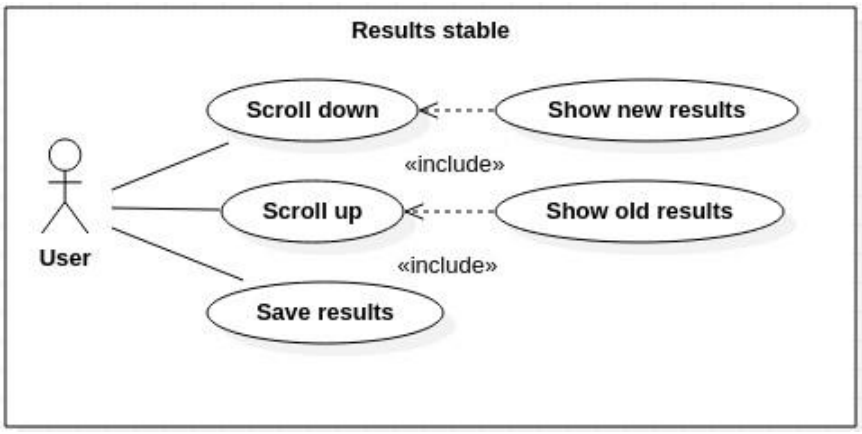

Рисунок 4.3 − Діаграма використання сторінки Results table.

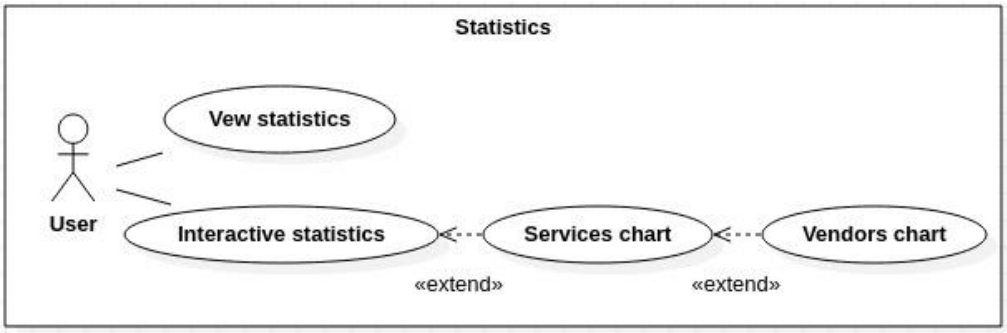

Рисунок 4.4 − Діаграма використання сторінки Statistics.

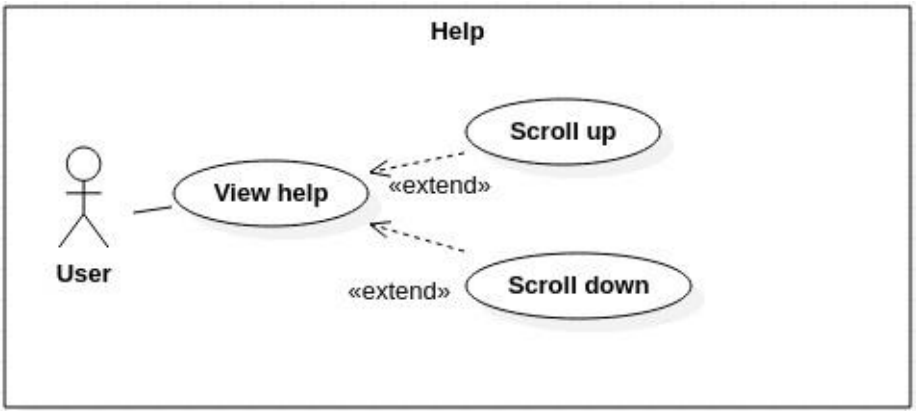

Рисунок 4.5 − Діаграма використання сторінки Help.

ВИСНОВКИ

З кожним роком все більше зростає пропускна здатність мереж. До того ж з появою в Україні 3G і 4G LTE, а також високому поширенні WI-FI мереж, тепер кожен пристрій, який за допомогою цих мереж отримує доступ в Інтернет може виступати як джерелом, так і ціллю DDoS-атак, також слід не забувати про атаки націлені на компрометацію і викрадення даних. Будь-які нові технології несуть з собою нові ризики. Нові технології дають однозначні переваги того, хто ними володіє і вміє їх безпечно використовувати. Перехід в цифрову епоху стає новим двигуном для бізнесу та економіки за умови дотримання достатнього рівня кібербезпеки. Для вирішення проблем з мережевою безпекою необхідно складне програмне забезпечення з модульною архітектурою, яке буде легко розширювати і доповнювати потрібними елементами для вирішення відповідних завдань.

Більшість мережевих пристроїв IoT вдають із себе «закриті системи». Його користувачі не зможуть додавати програмне забезпечення безпеки після того, як пристрої покинуть завод. Таке втручання анулює гарантію, а часто просто не представляється можливим. З цієї причини, захисні функції повинні бути спочатку вбудовані в пристрої IoT, щоб вони були безпечними за своєю архітектурою. Дані, пов'язані з IoT, повинні зберігатися в безпеці.

Вході роботи над дипломним проектом були вивчені такі питання:

− розробка мережевих додатків;

− аналіз безпеки мереж, мережевих пристроїв і сервісів;

− робота з середовищем розробки для мови Python;

− моделювання UML діаграм;

− мережева та інформаційна безпека;

− стан мережевої безпеки в Україні.

В результаті роботи розроблено мережевий сканер NaN який виконує наступні функції:

− управління системою плагінів (модуль. менеджер плагінів);

− виконання паралельного сканування (модуль. управління багатопотоковістю);

− виявлення та аналіз вразливостей в мережевих пристроях та сервісах (модуль. виявлення та аналізу вразливостей);

− об'єднання всіх модулів разом і розподіл завдань між модулями модуль. ядро).

Аналіз отриманих результатів дозволяє зробити наступні висновки:

− проблема з безпекою мереж, мережевих пристроїв і сервісів, є актуальними на даний момент, захищеність IoT пристроїв вкрай низька;

− проблеми з захищеністю IoT вимагають виправлення;

− використання новітніх засобів розробки таких як системи управління версій і середовища розробки прискорюють розробку у рази.

− подальший розвиток проекту передбачає використання даного програмного забезпечення для аналізу захищеності мереж у м. Одеса, як дослідження щоб показати рівень захищеності мережевих пристроїв. Планується робота з місцевими мережевими провайдерами які допоможуть провести дане дослідження, результати дослідження планується надати мережевим провайдерам для поліпшення якості захищеності мереж і мережевих пристроїв її сервісів підключених до мережі, а так само мережевих пристроїв користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Фальстарт або розвідка боєм: як мобільні оператори впроваджують інтернет речей. URL: https://mind.ua/ru/publications/20194010-falstart-ili-razvedka-boem-kak-mobilnye-operatory-vnedryayut-internet-veshchej. (Дата звернення 13.11.2019).

2. Як забезпечити IoT-безпеку в епоху "Connected Everything". URL: https://www.anti-malware.ru/analytics/Threats_Analysis/How-to-secure-IoT-devices-with-nist-recommendations. (Дата звернення 10.10.2019).

3. П'ять актуальних практик кіберзахисту для корпорацій. URL: https://delo.ua/special/pjat-aktualnyh-praktik-kiberzaschity-dlja-korpor-351219/. (Дата звернення 10.10.2019).

4. Kaspersky: IoT-зловредів проводять 20 тисяч атак за 15 хвилин. URL: https://threatpost.ru/kapersky-on-iot-state-of-security-in-h1-2019/34556/. (Дата звернення 10.10.2019).

5. Shodan is the world's first search engine for Internet-connected devices. URL: https://www.shodan.io/. (Дата звернення 01.11.2019).

6. Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit. URL: https://www.metasploit.com/. (Дата звернення 01.11.2019).

7. Nmap: the Network Mapper - Free Security Scanner. URL: https://nmap.org/. (Дата звернення 01.11.2019).

8. Zenmap - Official cross-platform Nmap Security Scanner GUI. URL: https://nmap.org/zenmap/. (Дата звернення 01.11.2019).

9. NmapSI4 Security Interface. URL: https://web.archive.org/web/20180324022230/http://nmapsi4.org/. (Дата звернення 01.11.2019).

10. XSpider. URL: https://www.ptsecurity.com/ru-ru/products/xspider/. (Дата звернення 01.11.2019).

11. OpenVAS - Open Vulnerability Assessment Scanner. URL: http://www.openvas.org/. (Дата звернення 01.11.2019).

12. Nessus Vulnerability Assessment | Tenable®. URL: https://www.tenable.com/products/nessus. (Дата звернення 01.11.2019).

13. The Python Wiki. URL: https://wiki.python.org/moin/. (Дата звернення 12.11.2019).

14. Documentation for Visual Studio Code. URL: https://code.visualstudio.com/docs. (Дата звернення 12.11.2019).

15. JavaScript. URL: https://en.wikipedia.org/wiki/JavaScript. (Дата звернення 15.11.2019).

16. Interactive JavaScript charts. URL: https://www.highcharts.com/. (Дата звернення 15.11.2019).

17. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/. (Дата звернення 15.11.2019).

18. Normalize.css: Make browsers render all elements more consistently. URL: https://necolas.github.io/normalize.css/. (Дата звернення 15.11.2019).

19. jQuery. URL: https://jquery.com/. (Дата звернення 15.11.2019).

20. Regular-Expressions.info - Regex Tutorial, Examples and Reference - Regexp Patterns. URL: https://www.regular-expressions.info/. (Дата звернення 15.11.2019).

21. JavaScript Object Notation. URL: https://www.json.org/. (Дата звернення 22.11.2019).

22. Система контролю версій Git. URL: https://git-scm.com/. (Дата звернення 22.11.2019).

23. Linux. URL: https://en.wikipedia.org/wiki/Linux. (Дата звернення 20.11.2019).

24. Unified Modeling Language. URL: https://en.wikipedia.org/wiki/Unified_Modeling_Language. (Дата звернення 20.11.2019).

25. Середовище розробки на мові UML, StarUML. URL: http://staruml.io/. (Дата звернення 22.11.2019).

26. WebSocket, протокол зв'язку поверх TCP-з'єднання. URL: https://en.wikipedia.org/wiki/WebSocket. (Дата звернення 20.11.2019).

27. Мова розмітки, HTML5. URL: https://en.wikipedia.org/wiki/HTML5. (Дата звернення 20.11.2019).

28. Cascading Style Sheets. URL: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. (Дата звернення 20.11.2019).

29. PyPI · The Python Package Index. URL: https://pypi.org/. (Дата звернення 08.11.2019).

30. Сервіс перевірки підозрілих файли файлів, VirusTotal. URL: https://www.virustotal.com/. (Дата звернення 08.11.2019).

31. The world's leading software development platform GitHub. URL: https://github.com/. (Дата звернення 12.11.2019).

Д О Д А Т К И

ДОДАТОК А

Веб інтерфейс управління мережевим сканером

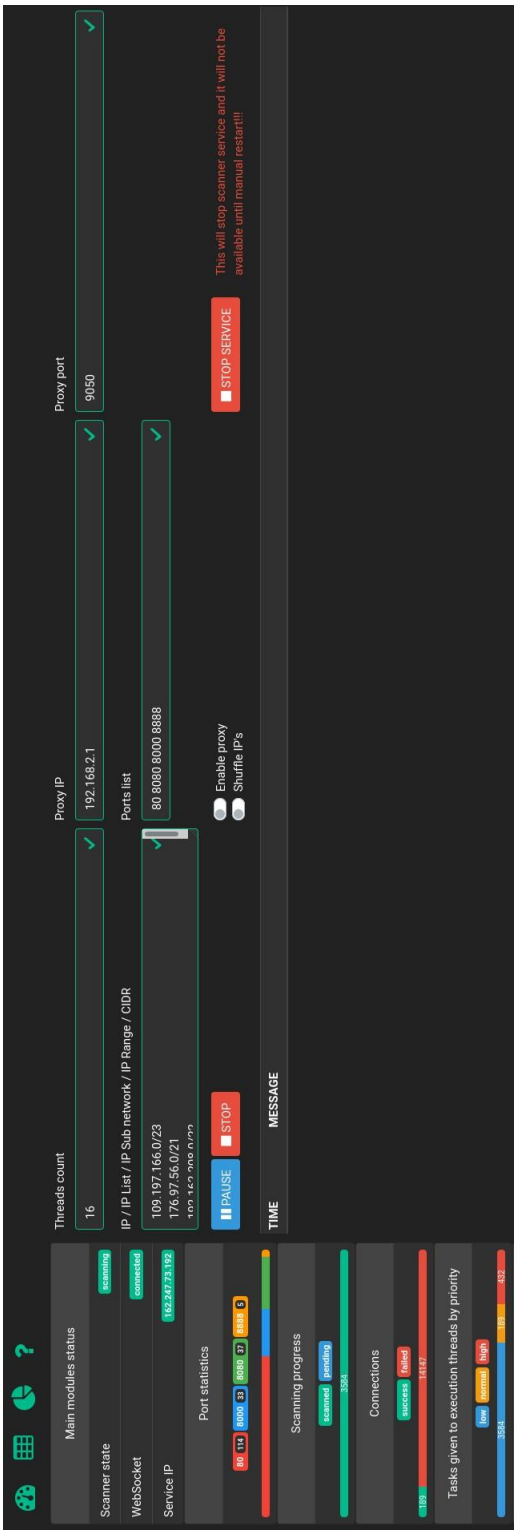

Рисунок А.1 Сторінка панель управління

#	IP	PORT	TYPE	COLLECTED
1	192.162.208.182	80	ROUTER	{'ip': '192.162.208.182', 'port': 80, 'type': <ServiceType.ROUTER: 1>, 'vendor': 'TP-Link', 'detailed': False, 'auth': 'Basic', 'realm': 'TP-LINK Wireless N Router WR841N', 'model': 'WR841N'}
2	176.97.61.45	80	UNKNOWN	{'ip': '176.97.61.45', 'port': 80, 'type': <ServiceType.UNKNOWN: 5>}
3	192.162.210.53	8080	DVR	{'ip': '192.162.210.53', 'port': 8080, 'type': <ServiceType.DVR: 3>, 'title': 'WEB SERVICE', 'vendor': 'Dahua', 'detailed': True, 'cap_js': {'talkTypes': ['2', '4', '1'], 'devType': '27', 'userInfo': 'This is user info!', 'streamCap': '19', 'channelNames': ['KAM 1', 'KAM 2', 'KAM 3', 'KAM 4', 'KAM 5', 'KAM 6', 'KAM 7', 'KAM 8'], 'capTcpPort': '37777'}, 'web_cap_config': {'Support_NEWRESOLUTION': 1, 'demand': '', 'multiPreview': 0, 'noCopyPtzLink': True, 'supportDecMultiP': 0, 'userNameLength': 15, 'userPasswordLength': 16, 'videoLineType': 0}}
4	192.162.211.192	80	DVR	{'ip': '192.162.211.192', 'port': 80, 'type': <ServiceType.DVR: 3>, 'title': 'WEB SERVICE', 'vendor': 'Dahua', 'detailed': True, 'cap_js': {'talkTypes': ['2', '1', '4'], 'devType': 'DHI-XVR5116HS-S2', 'userInfo': 'This is user info!', 'streamCap': '19', 'rtspport': '554', 'ClientType': '0', 'capTcpPort': '37777', 'radius': 'false', 'httpPort': '80'}, 'web_cap_config': {'DockUser': ['Onvif'], 'IntelliVersion': 1, 'PasswdSerialNo': '3K01AD0PAZ7A51F', 'SetReduceNoiseMode': 0, 'SupportNetSNMP': False, 'acutanceGroup': 1, 'ahdChnGroup': 1, 'b_3GKeepAlive': False, 'b_AudioSingle': True, 'b_FaceConfigOneChn': True, 'b_SupportDeviceInit': True, 'b_cableType': True, 'b_compelPassword': True, 'b_guiResolutionConflictVideoSpot': ['3840*2160', '2560*1440'], 'b_multiPreviewNotUseSplit': True, 'b_supportCloudUpdate': True, 'b_supportNetAlarm': False, 'b_supportPirAlarm': True, 'changeDenoiseLabel': True, 'chnTypeAnalyse': True, 'customChanelSplit': 16, 'defAccount': 'admin', 'demand': '', 'guideEnable': False, 'loginType': [0, 4, 3], 'multiPreview': 1, 'svcNotModify': True, 'tcp_FastDownload': False, 'vendor': 'Private', 'videoLineType': 0, 'weakPasswordTip': False}}
5	109.197.166.8	8080	ROUTER	{'ip': '109.197.166.8', 'port': 8080, 'type': <ServiceType.ROUTER: 1>, 'vendor': 'Asus', 'detailed': True}
6	176.97.58.81	8080	ROUTER	{'ip': '176.97.58.81', 'port': 8080, 'type': <ServiceType.ROUTER: 1>, 'vendor': 'ZyXEL', 'detailed': False, 'auth': 'Digest', 'realm': 'ZyXEL Keenetic 4G III', 'qop': 'auth', 'model': 'Keenetic 4G III'}
7	176.97.61.18	80	SERVER	{'ip': '176.97.61.18', 'port': 80, 'type': <ServiceType.SERVER: 4>, 'vendor': 'Apache', 'version': '2.4.18', 'os': 'Ubuntu'}
8	192.162.209.143	80	DVR	{'ip': '192.162.209.143', 'port': 80, 'type': <ServiceType.DVR: 3>, 'title': 'WEB SERVICE', 'vendor': 'Dahua', 'detailed': True, 'cap_js': {'talkTypes': ['2', '1', '4'], 'devType': 'DHI-XVR4116HS', 'userInfo': 'This is user info!', 'streamCap': '19', 'rtspport': '554', 'ClientType': '0', 'capTcpPort': '37777', 'radius': 'false', 'httpPort': '80'}, 'web_cap_config': {'IntelliVersion': 1, 'SetReduceNoiseMode': 0, 'acutanceGroup': 4, 'ahdChnGroup': 1, 'b_FaceConfigOneChn': False, 'b_cableType': True, 'b_compelPassword': True, 'changeDenoiseLabel': True, 'chnTypeAnalyse': False, 'customChanelSplit': 16, 'demand': '', 'loginType': [0, 4, 3], 'multiPreview': 1, 'svcNotModify': True, 'videoLineType': 0, 'weakPasswordTip': False}}
9	176.97.57.249	80	ROUTER	{'ip': '176.97.57.249', 'port': 80, 'type': <ServiceType.ROUTER: 1>, 'auth': 'Digest', 'realm': 'ZyXEL Keenetic Start', 'qop': 'auth', 'nonce': '13c0b', 'algorithm': 'MD5', 'vendor': 'ZyXEL', 'detailed': False, 'model': 'Keenetic Start'}
10	192.162.208.131	80	ROUTER	{'ip': '192.162.208.131', 'port': 80, 'type': <ServiceType.ROUTER: 1>, 'title': 'RouterOS router configuration page', 'vendor': 'Mikrotik', 'detailed': True, 'version': '6.45.7'}
11	109.197.167.8	80	UNKNOWN	{'ip': '109.197.167.8', 'port': 80, 'type': <ServiceType.UNKNOWN: 5>}
12	176.97.58.40	80	SERVER	{'ip': '176.97.58.40', 'port': 80, 'type': <ServiceType.SERVER: 4>, 'vendor': 'Apache', 'version': '2.4.7', 'os': 'Ubuntu'}

Рисунок А.2 Сторінка таблиця результатів

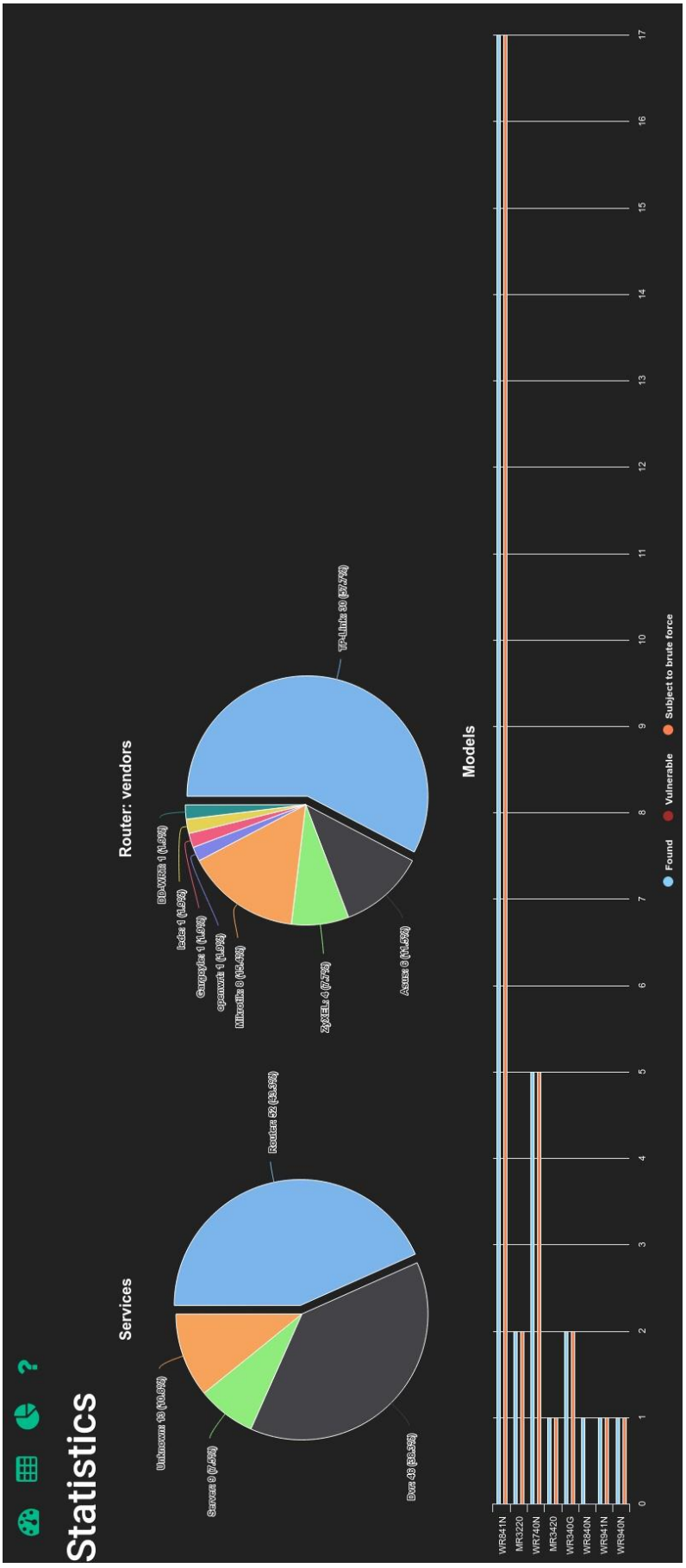

Рисунок А.3 Сторінка статистики

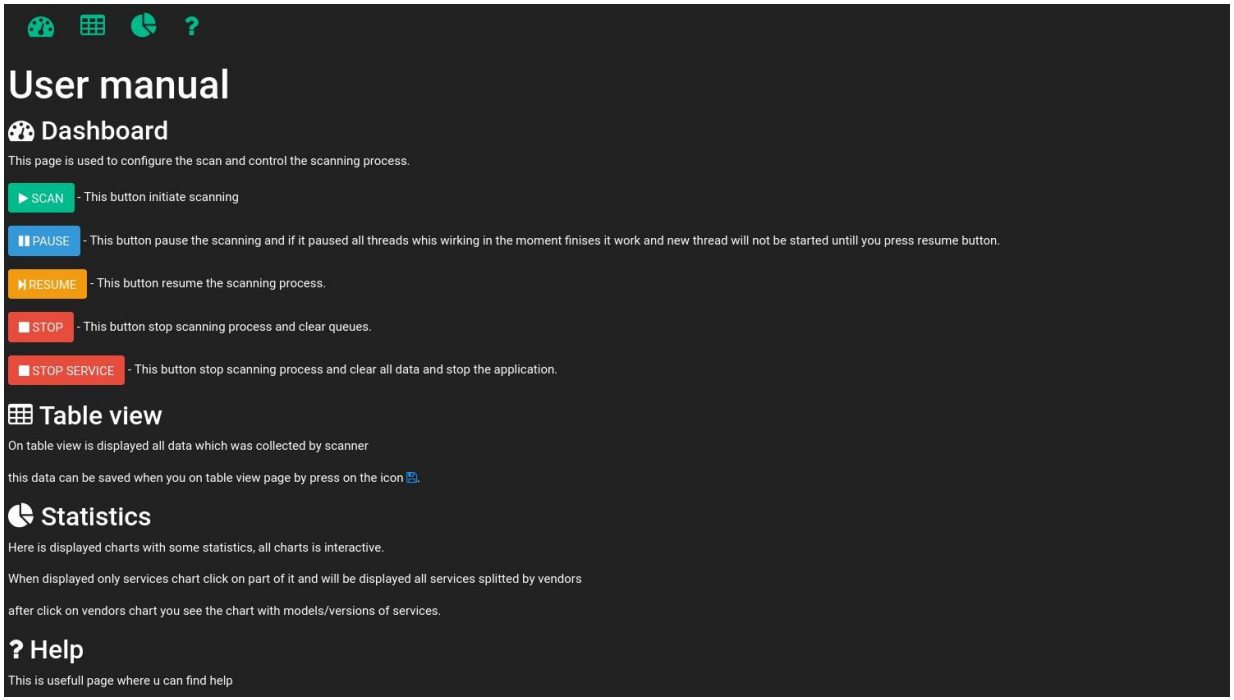

Рисунок А.4 Сторінка допомоги

ДОДАТОК Б

Програмний код головного класу

```
#-*- coding: utf-8 -*-

import socket
import sys
from multiprocessing.dummy import Pool
from multiprocessing.pool import ApplyResult
from queue import Queue
from random import shuffle
from threading import Thread
from time import sleep

import iptools
import socks
from loguru import logger

from app.auxiliary import safe_request

from app.base.addons import ManagerAddonBase
from app.components import (AddonsManager, JsonObjectManager,
Scheduler,
                            ThreadPool, DataManager, Statistics,
Session, ServiceIpMonitor)

from app.data import ManagerType, Task, TaskPriority, ScanningStatus
from .HttpApi import HttpApi

class NanServer(object):

    def __init__(self, config: str):
        self.data = DataManager()
        self.data.set('results', list())
        self.data.set('not-handled', list())

        self.config = JsonObjectManager(config)
        self.session = Session(self)

        self.statistics = Statistics(self)
        self.scheduler = Scheduler(self)
        self.http_service = HttpApi(self)
        self.thread_pool = ThreadPool(self)
        self.ip_monitor = ServiceIpMonitor(self)

        self._is_working  =  True
        self._external_ip = 'unknown'
        self._scanning_status = ScanningStatus.STANDBY
```

```
        self.data.set("servers-regex",
JsonObjectManager("config/servers-regex.json").get(None, []))
        self.data.set("titles-regex",
JsonObjectManager("config/titles-regex.json").get(None, []))
        self.data.set("routers-regex",
JsonObjectManager("config/routers-regex.json").get(None, []))

    def set_external_ip(self, ip):
        self._external_ip = ip

    def get_external_ip(self):
        return self.__external_ip

    def set_scanning_status(self, status):
        self._scanning_status = status

    def get_scanning_status(self):
        return self.__scanning_status

    def enable_proxy(self):
        pxy_ip, pxy_port = self.session.get_proxy()

        if not pxy_ip or not pxy_port:
            logger.error(
                f"Proxy can't be set, no ip ({pxy_ip}) or port
({pxy_port})"
            )

        logger.warning(f"*** Setup proxy ({pxy_ip}:{pxy_port}) ***")
        socks.setdefaultproxy(socks.PROXY_TYPE_SOCKS5, pxy_ip,
int(pxy_port))
        socket.socket = socks.socksocket

    def  disable_proxy(self):
        logger.warning("*** Proxy disabled ***")

        socks.setdefaultproxy(None)
        socket.socket = socks.socksocket

    def setup(self):
        addons = AddonsManager(self).get_addons("managers",
ManagerAddonBase)

        for _, addon in addons.items():
            self.scheduler.setHandler(addon(self))

        self.scheduler.run()
        self.ip_monitor.run()
        self.http_service.run()
```

```
    def run_addon(self, addon, kwargs: dict):
        try:
            self.thread_pool.deferredExecute(addon,
                error_callback=addon.on_error,
                callback=addon.on_complete,
                kwds=kwargs,
            )
        except ValueError as e:
            return

    def put(self, task: Task):
        self.scheduler.put(task)

    def save(self, task: Task):
        results: list = self.data.get('results', [])
        results.append(task.data)

    def run(self):
        self.setup()

        logger.warning(f"Press '^C' to stop.")
        while self.__is_working:
            try:
                sleep(1)
            except KeyboardInterrupt:
                self._is_working = False

        self.thread_pool.stop()
        logger.debug(f"{__name__}: thread_pool.stop()")
        self.scheduler.stop()
        logger.debug(f"{__name__}: scheduler.stop()")
        self.ip_monitor.stop()
        logger.debug(f"{__name__}: ip_monitor.stop()")
        self.http_service.stop()
        logger.debug(f"{__name__}: http_service.stop()")

    def stop(self):
        self._is_working = False

    def start_sca
        ip_list =
list(iptools.IpRangeList(*self.session.get_ip_list()))
        self.statistics.set_all_for_scanning(len(ip_list))

        if self.session.get_shuffle_status():
            shuffle(ip_list)

        for ip in ip_list:
            self.put(Task(ManagerType.SERVICE_DISCOVERY, {
                'ip': ip, 'ports': ports
            }, TaskPriority.LOW))
```

```
        logger.warning("*** start sequence complete
***")enable_proxy()
        else:
            self.disable_proxy()

        ports = self.session.get_port_list()
        self.statistics.initialize_ports(ports)

        ip_list =
list(iptools.IpRangeList(*self.session.get_ip_list()))
        self.statistics.set_all_for_scanning(len(ip_list))

        if self.session.get_shuffle_status():
            shuffle(ip_list)

        for ip in ip_list:
            self.put(Task(ManagerType.SERVICE_DISCOVERY, {
                'ip': ip, 'ports': ports
            }, TaskPriority.LOW))

        logger.warning("*** start sequence complete ***")
```

ДОДАТОК В

Програмний код менеджера доповнень

```
#-*- coding: utf-8 -*-

import os
from loguru import logger
import importlib

IGNORED_ADDONS = [
    " pycache__",
    " init  "
]

class AddonsManager(object):

    def __init__(self, core):
        self._addons_location =
core.config.get("core.addons.location")

    def get_addons(self, directory: str, addon_type):
        path = os.path.join(self._addons_location, directory)
        return AddonsManager._find_addons(path, addon_type)

    @staticmethod
    def __get_addons_list_from(path: str):
        addons = map(lambda x: os.path.splitext(x)[0],
os.listdir(path))
        addons = filter(lambda x: x not in IGNORED_ADDONS, addons)
        addons = map(lambda x: os.path.join(path, x), addons)
        addons = map(lambda x: x.replace("/", '.'), addons)
        return list(addons)

    @staticmethod
    def __find_addons(path: str, addon_type):
        addons = AddonsManager._get_addons_list_from(path)
        founded_addons = {}

        for addon in addons:
            importlib.import_module(addon)

        for addon in addon_type.__subclasses_():
            if addon.__module___not in addons:
                continue

            founded_addons[addon._addon_] = addon

        return founded_addons
```

ДОДАТОК Г

Програмний код одного з доповнень ідентифікації сервісів

```
# -*- coding: utf-8 -*-

from bs4 import BeautifulSoup
from loguru import logger
import requests
import re

from app.data import Task
from app.data import  ManagerType
from app.data import  ServiceType
from app.data import  ManagerType
from app.data import TcpConnectResult
from app.data import TaskPriority

from app.base.addons import WorkerAddonBase
from app.base.addons import ManagerAddonBase

from app.NanServer import NanServer
from app.auxiliary import safe_request

OPENWRT_URL_REGEX_LIST = [
    re.compile(r"(?P<url>cgi-bin\/luci)", re.I)
]

REGEX_LIST_VERSIONS = [
    re.compile(r"(?P<vendor>LEDE) (?P<codename>Reboot)
(?P<version>[\d.]+)", re.I),
    re.compile(r"(?P<vendor>OpenWrt) (?P<codename>Chaos
Calmer|Attitude Adjustment|Barrier Breaker) (?P<version>[\d.]+(?:[-
]rc\d)?)", re.I)
]

class OpenWrt(WorkerAddonBase):
    __addon___= "router-openwrt"

    @staticmethod
    def is_accept(task: Task) -> bool:
        content = task.data.get('http', {}).get('content', None)

        return content and bool(OpenWrt.get_web_ui_url(content))

    @staticmethod
    def get_web_ui_url(content: str):

        for regex in OPENWRT_URL_REGEX_LIST:
```

```
            result = regex.search(content.decode())
            if result:
                return result.groupdict()

    def __init__(self, core: NanServer):
        self._core = core

    def __call__(self, task: Task):
        content = task.data['http']['content']

        url = OpenWrt.get_web_ui_url(content).get('url', None)
        if not url:
            logger.error("if not url")
            return

        tcp: TcpConnectResult = task.data['tcp']

        timeout: float = self.__core.config.get("timeout.http", 2.0)

        url = f"http://{tcp.ip}:{tcp.port}/{url}"

        r = safe_request("GET", url, timeout=timeout)

        dom = BeautifulSoup(r.content, 'lxml')
        if not dom:
            logger.error("if not dom")
            return

        footer = dom.find("footer")
        hostinfo = dom.find("div", {"class": "hostinfo"})

        info = (footer or hostinfo)

        if not info:
            logger.error("if not info")
            return

        result = None
        for regex in REGEX_LIST_VERSIONS:
            result = regex.search(info.text)
            if result:
                break

        if not result:
            logger.error("if not result")
            return

        data = result.groupdict()
        task.data["collected"].update({
            "vendor": str(data["vendor"]).lower(),
            "version": str(data["version"]).lower(),
```

```
            'type': ServiceType.ROUTER
            })

        if 'codename' in data:
            task.data["collected"].update({
                'codename': str(data["codename"]).lower()
                })

        return task

    def on_complete(self, result: Task):
        if not result:
            logger.error(f"{result}")
            # work finished
            return

        result.type = ManagerType.SERVICE_VULNERABILITY
        result.priority = TaskPriority.HIGH
        self.__core.put(result)
```