

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ
ВРАЗЛИВОСТЕЙ СИСТЕМ ВІДЕОСПОСТЕРЕЖЕННЯ
SOFTWARE FOR DETECTING THE VULNERABILITIES OF VIDEO
SURVEILLANCE SYSTEM

Виконав: студент(ка) 4 курсу, групи КНТ-128
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Чабан І.С.

(прізвище та ініціали)

Керівник Табунщик Г. В.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Чабана Іллі Сергійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення програмного забезпечення для пошуку
вразливостей систем відеоспостереження Software for Detecting the
Vulnerabilities of Video Surveillance System

керівник проєкту (роботи) Табунщик Галина Володимирівна, к.т.н., професор,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 07 червня 2022 року

3. Вихідні дані до проєкту (роботи) Вразливості систем відеоспостереження,
виробники систем відеоспостереження, стандартні налаштування систем
відеоспостереження, користувачі системи: звичайні користувачі, якими
необхідно провести пошук вразливостей у системі відеоспостереження.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Огляд та аналіз предметної області. 2. Вибір та обґрунтування
структури системи, що проектується. 3. Основні рішення щодо реалізації
компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6.
Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Табунщик Г.В., професор		
Нормоконтролер	Белова А.В., асистент		

7. Дата видачі завдання “6” квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10-12 тиждень	Додатки
8	Нормоконтроль та рецензування.	13–14 тижні	
9	Захист роботи.	15 тиждень	

Студент

(підпис) **Чабан І.С.**
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) **Табунщик Г.В.**
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
154 с., 51 рис., 17 табл., 2 додатки, 11 джерел.

ПОШУК ВРАЗЛИВОСТЕЙ, СИСТЕМА ВІДЕОСПОСТЕРЕЖЕННЯ, IP-КАМЕРА, КІБЕРАТАКА, RTSP, IP, PYTHON, QML.

Об'єкт дослідження – вразливості систем відеоспостереження.

Предмет дослідження – програмне забезпечення для пошуку вразливостей систем відеоспостереження.

Мета роботи – розробка програмного забезпечення для пошуку вразливостей систем відеоспостереження.

Матеріали, методи та технічні засоби - IP-камера, мова програмування Python, декларативна мова програмування QML, запити протоколом RTSP, серіалізація та десеріалізація, багатопроцесорність.

Результати. Виявлено найпоширенішу вразливість, розроблено програмне забезпечення для виявлення вразливостей систем відеоспостереження.

Висновки. Розроблено програмне забезпечення, яке виконує пошук найпоширенішої вразливості систем відеоспостереження.

Галузь використання – пошук вразливостей систем відеоспостереження.

ABSTRACT

Explanatory note to the master's thesis project: 154 pages, 17 tables, 51 figures, 2 appendices, 11 sources.

VULNERABILITY DEVELOPMENT, VIDEO SURVEILLANCE SYSTEM, IP CAMERA, CYBER ATTACK, RTSP, IP, PYTHON, QML.

The object of study – the vulnerabilities of video surveillance systems.

The subject of research – software to search for vulnerabilities in video surveillance systems.

The purpose of the work is to develop software to search for vulnerabilities in video surveillance systems.

Materials, methods and technical means – IP-camera, Python programming language, declarative QML programming language, RTSP requests, serialization and deserialization, multiprocessor.

Results. The most common vulnerabilities have been identified, software has been developed to detect vulnerabilities in video surveillance systems.

Conclusions. Developed software that searches for the most common vulnerabilities in CCTV systems.

Scope - search for vulnerabilities in video surveillance systems.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП.....	9
1 ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1 Поняття системи відеоспостереження.....	11
1.2 Виникнення вразливостей систем відеоспостереження	11
1.3 Експлуатація вразливостей	12
1.4 Огляд актуальних вразливостей	14
1.5 Протокол RTSP	19
1.6 Аналіз вимог	27
1.7 Висновок за розділом 1	28
2 ВИБІР ТА ОБГРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЩО ПРОЕКТУЄТЬСЯ.....	31
2.1 Вибір середовища розробки.....	31
2.2 Вибір мови програмування	32
2.3 Вибір бібліотеки для створення інтерфейсу	33
2.4 Вибір бібліотеки для розподілу процесів	33
2.5 Вибір способу зберігання даних.....	34
2.6 Висновок за розділом 2	34
3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ ...	35
3.1 Опис файлів та папок програми	35
3.2 Алгоритм функціонування програми	36
3.3 Розгляд модуля DataManager.py	38
3.4 Розгляд модуля ProcessManager.py	42
3.5 Розгляд модуля UpdateAndBruteProcessManager.py	46
3.6 Розгляд модуля CameRAadar.py	51
3.7 Структура бази даних	57
4 КЕРІВНИЦТВО ПРОГРАМІСТА.....	59
4.1 Призначення й умови застосування програми.....	59

4.2 Характеристика програми.....	59
4.3 Звернення до програми.....	60
4.4 Початкові та вихідні дані	61
4.5 Повідомлення	61
5 КЕРІВНИЦТВО ОПЕРАТОРА	62
5.1 Призначення програми.....	62
5.2 Умови виконання програми.....	62
5.3 Виконання програми.....	63
5.4 Повідомлення оператору.....	71
ВИСНОВКИ.....	75
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	77
ДОДАТОК А Текст програми.....	78
ДОДАТОК Б Слайди презентації.....	146

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

RTSP – Real Time Streaming Protocol;

SDP – Session Description Protocol;

RTP – Real-time Transport Protocol;

GIL – Global Interpreter Lock;

QML – Qt Modeling Language;

IP – Internet Protocol;

URI – Uniform Resource Identifier;

URN – Uniform Resource Name;

URL – Uniform Resource Locator.

ВСТУП

Камери відеоспостереження стали невід'ємною частиною нашого життя, вони впроваджені у всі сфери діяльності людини в яких необхідно постійно вести спостереження для виявлення, фіксації та запобігання злочинних дій, чи то крадіжка, вандалізм чи шахрайські дії.

Люди встановлюють їх у будинках, на підприємствах, у бізнесі, довіряючи їм свою безпеку та конфіденційність одержуваної інформації, при цьому часто не змінюючи стандартні ім'я користувача та пароль для доступу до системи відеоспостереження, найчастіше це відбувається з кількох причин:

- низької цифрової грамотності;
- недбалості;
- недостатня мотивація із боку виробника.

Не змінюючи стандартні ім'я користувача та пароль для доступу до системи відеоспостереження, вона стає легкою здобиччю для кібератак, які можуть порушити її функціонування або призвести до витоку конфіденційної інформації.

Кібератаки на системи відеоспостереження часто проводяться за допомогою сканування діапазону ір-адрес країн чи міст і перебору стандартних або найпоширеніших імен користувачів та паролів для доступу до систем відеоспостереження.

Пошуком незахищених або слабо захищених систем відеоспостереження в мережі інтернет часто є люди не зацікавлені в тому, щоб заподіяти якусь шкоду системі відеоспостереження, їх метою є банальне бажання знайти щось цікаве в "безмежній" мережі інтернеті.

Переважає більшість сучасних системи відеоспостереження підтримують повне управління та трансляцію медіа контенту через інтернет, транслявання медіа контенту виконується за стандартизованим протоколом RTSP, дана функція включена за замовчуванням, це означатиме, що всі такі системи відео спостереження які підключені до інтернету виступають як

сервер, виявити який може кожен у кого є доступ в інтернет, доступ до трансляції захищається ім'ям користувача і паролем, які задані за замовчуванням, а в деяких випадках пароль взагалі відсутній, мається на увазі, що користувач змінить хоча б пароль, який надійно захищатиме їхню систему відеоспостереження від несанкціонованого доступу, але не всі люди надають цьому значення, не підозрюючи яких наслідків це може призвести.

Аналізуючи вище написане можна зробити висновок, що є досить велика кількість незахищених і слабозахищених систем відеоспостереження.

Метою дипломної роботи є розробка програмного забезпечення, яке дозволить людям перевірити свої системи відеоспостереження на наявність вразливості або для проведення лабораторних експериментів для демонстрації того, наскільки важливий надійний захист і що йому потрібно приділяти більше уваги.

1 ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття системи відеоспостереження

Система відеоспостереження – це комплекс охоронних пристроїв, призначений для постійного візуального спостереження над територією, що захищається. Система відеоспостереження включає набір камер, реєстратор з вбудованим пристроєм для запису, блок живлення, кабель відеоспостереження для передачі інформації, монітор і додаткові витратні матеріали, необхідні для монтажу системи [1].

1.2 Виникнення вразливостей систем відеоспостереження

1.1.1 Вразливості виробництва

Сюди належать всі проблеми, пов'язані з реалізацією функціональності IP-камер. Головна з них полягає в тому, що вартість апаратної частини камер значно менша, ніж витрати на розробку прошивки. Результатом прагнення компаній заощадити стають найдивніші рішення, наприклад:

- неоновлювані прошивки або прошивки без автоматичного оновлення;
- можливість отримати доступ до веб-інтерфейсу камери шляхом багаторазових спроб ввести неправильний пароль або натискання кнопки «Скасувати»;
- відкритий доступ до всіх камер із сайту виробника;
- невідключний заводський сервісний обліковий запис зі стандартним паролем або без нього (таку вразливість використовувала в 2016 хакерська група Lizard Squad для створення DDoS-ботнета з потужністю атак до 400 ГБ/с з камер відеоспостереження);
- можливість несанкціонованої зміни налаштувань навіть при включеній авторизації та забороні анонімного доступу;

- відсутність шифрування відеопотоку та/або передача облікових даних у відкритому вигляді;
- використання вразливого веб-сервера GoAhead;
- тиражування вразливих прошивок (поширене серед китайських виробників пристроїв);
- вразливості пристроїв для зберігання відео - наприклад, зовсім недавно була опублікована інформація про проникнення на мережні накопичувачі Synology і Lenovo Iomega шкідливих даних, що видаляють або шифрують файли і вимагають викуп [2].

1.1.2 Вразливості використання

Навіть якщо виробники камер виправлять усі помилки та випустять ідеальні з точки зору безпеки камери, це не врятує ситуацію, оскільки нікуди не подінуться рукотворні вразливості, причина яких у людях, які обслуговують обладнання:

- використання паролів за замовчуванням, якщо є можливість змінити їх;
- відключення шифрування або VPN, якщо камера дозволяє його використовувати (далеко не всі камери мають достатню продуктивність для такого режиму роботи);
- відключення автоматичного оновлення прошивки камери;
- забудькуватість або недбалість адміністратора, що не оновлює прошивку пристрою, який не вміє робити це автоматично [2].

1.3 Експлуатація вразливостей

Створення ботнетів — це група кількох комп'ютерів та/або пристроїв, об'єднаних під керуванням одного або кількох командних серверів. На пристрої встановлюється спеціальне програмне забезпечення, яке за командою із центру

виконує різні завдання. Це може бути розсилання спаму, проксіювання трафіку, DDoS і навіть майнінг крипто валюти [2].

Шантаж — захоплення контролю над камерами та відеореєстраторами дає можливість спостерігати за приватним життям людей, серед яких можуть бути державні діячі та просто відомі особи. Публікація відео за їх участю здатна викликати скандал і завдати шкоди репутації власника, а отже, загроза оприлюднення – серйозний мотив для вимоги викуп [2].

Приховування злочинів за допомогою протоколу RTSP. Щоб замінити запис на IP-камері, достатньо:

- отримати RTSP-посилання для камери, потік з якої потрібно замінити;
- використовуючи це посилання, записати потік у відеофайл;
- відтворити записаний відеореєстратор у режимі потокового мовлення;
- захиститися від вторинної заміни потоку.

Очевидно, що маючи записаний відеофайл, можна зробити з ним безліч цікавого, наприклад, прибрати “зайві” фрагменти або додати нові.

Записаний файл транслюємо у форматі RTSP, після чого розв'язуємо задачу виведення камери з ладу. Нижче наведено кілька варіантів, які відрізняються в залежності від об'єкта, який ми хочемо атакувати. Насправді способів набагато більше, розглянемо найголовніші. Перше, що нам потрібно, це потрапити в потрібну нам мережу.

Щоб підмінити трансляцію потрібний доступ до мережі wi-fi, в якому можна можна скористатися класичним методом заміни:

- за допомогою програм таких як arpspoof стати між камерою і сервером;
- за допомогою програм таких як ip_forward переадресовувати запити з сервера відеоспостереження на IP-камеру та навпаки;
- за допомогою програм таких як iptables переадресовувати всі запити по RTSP-порту на сервер відеоспостереження не з камери, а з атакуючого комп'ютера [2].

Нетсталкінг — це діяльність, здійснювана в межах мережі методом пошуку, спрямовану на виявлення маловідомих, малодоступних і маловідвідуваних об'єктів з їх можливим подальшим аналізом, систематизацією і зберіганням, з метою естетичного та інформаційного задоволення шукача. Під це визначення потрапляє пошук об'єктів глибокої мережі, у тому числі IoT-пристроїв, якими є сучасні системи відео спостереження [3].

1.4 Огляд актуальних вразливостей

1.4.1 Вразливість прошивки систем відеоспостереження Hikvision

Дана вразливість надає можливість отримання моментального знімка зображення і системної інформації з камери. Також дана вразливість дозволяє використовувати інструмент скидання пароля для захоплення камери шляхом встановлення нового пароля.

Дана вразливість міститься в одному рядку, який дозволяв миттєво отримати доступ до будь-якої камери, незалежно від пароля адміністратора. Все, що для цього необхідно, додати один рядок до http-запиту камери Hikvision: “?auth=YWRtaW46MTEK”.

Приклади використання цієї вразливості:

- отримати список усіх користувачів та їх ролей
(<http://camera.ip/Security/users?auth=YWRtaW46MTEK>);
- отримати знімок камери без аутентифікації (<http://camera.ip/onvif-http/snapshot?auth=YWRtaW46MTEK>);
- завантажити конфігурацію камери
(<http://camera.ip/System/configurationFile?auth=YWRtaW46MTEK>).

Таким же чином можна здійснювати решту викликів HikCGI, у тому числі ті, які додають нових користувачів або прошивають прошивку камери.

Оскільки більшість пристроїв Hikvision захищають лише образи прошивки шляхом обфускації, можна запустити довільний код або зробити сотні тисяч підключених пристроїв назавжди непридатними для використання лише

одним простим HTTP-запитом.

Будь-яка доступна камера Hikvision з вразливою прошивкою вразлива для повного захоплення або блокування. Hikvision випустила виправлення прошивки в березні 2017 року, хоча статистика IPVM показує, що більш ніж 60% камер Hikvision все ще вразливі.

Інструмент для скидання паролів користувачів (включно з адміністратором користувача) був випущений через кілька днів після оголошення експлойту. Програма скидання пароля Hikvision дозволяє користувачеві вводити IP-адресу камери, отримувати список користувачів і вибірково скидати пароль для будь-якого користувача. Вивчення вихідного коду цієї програми показує, що рядок `auth=YWRtaW46MTEK` використовується для зміни паролів користувачів.

За оцінками IPVM, більше 300 000 пристроїв Hikvision є вразливими, ґрунтуючись на результатах сканування Shodan [4].

1.4.2 Вразливість стандартних налаштувань систем відеоспостереження Dahua та Hikvision

Вразливість систем відеоспостереження виробника Dahua та Hikvision полягає в стандартних налаштуваннях, при яких дані системи виступає як сервер, до якого можна звернутися по 3777 порту для Dahua і 8080 для Hikvision.

Експлуатування вразливості полягає в брутфорсі стандартного та найбільш популярних імені користувача та пароля.

Ґрунтуючись на результатах пошукового запиту у Shodan до глобальної мережі інтернет підключено близько 291119 пристроїв з відкритим 8080 портом та відповіддю «401 Unauthorized» (рис. 1.1).



Рисунок 1.1 – Результатах пошукового запиту у Shodan по відкритому 8080 порту та відповіддю «401 Unauthorized»

А пристроїв з відкритим 3777 портом лише 434 (рис. 1.2).

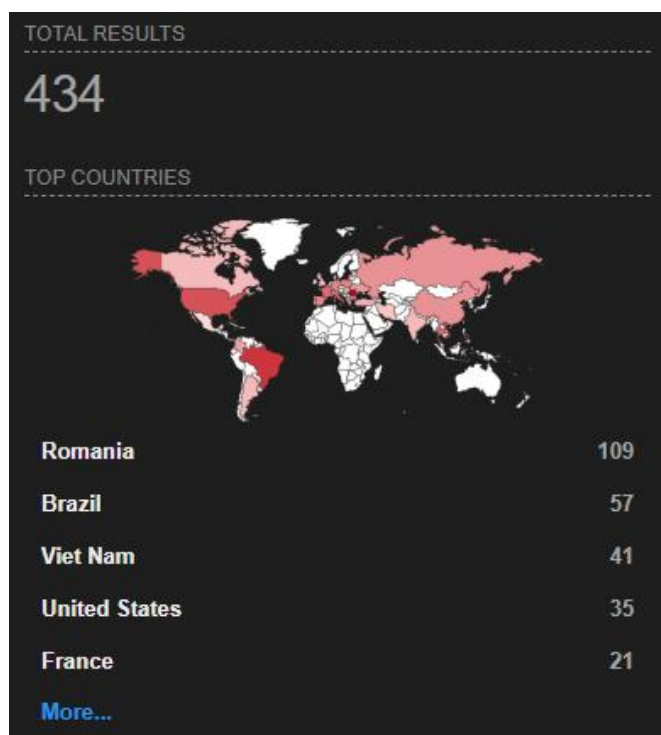


Рисунок 1.2 – Результатах пошукового запиту у Shodan по відкритому 3777 порту

1.4.3 Вразливість стандартних налаштувань систем відеоспостереження RTSP протоколу

Всі системи відеоспостереження, які підтримують доступ в інтернет, при не зміні стандартних налаштувань, виступають як сервер, до якого можна підключитися за протоколом RTSP для отримання відео та аудіо трансляції.

Доступ до трансляції здійснюється методом перебору стандартних імені користувача та паролів різних виробників та URN.

Стандартні імена користувача та паролі популярних виробників наведено у таблиці 1.1.

Таблиця 1.1 – Стандартні імена користувача та паролі популярних виробників

Виробник	Ім'я користувача	Пароль
ACTi	admin	123456
American Dynamics	admin	admin
Axis	root	pass
Dahua	admin	admin
Hikvision	admin	12345
Panasonic	admin	12345
GeoVision	admin	admin
LTS Security	admin	12345
Ubiquiti	ubnt	ubnt
Foscam	admin	-
Vivotek	root	-
Speco	admin	1234

URN популярних виробників наведено в таблиці 1.2.

Таблиця 1.2 – URN популярних виробників

Виробник	URN
ACTi	/stream1
American Dynamics	/primarystream
Axis	/axis-media/media.amp
Dahua	/cam/realmonitor?channel=1&subtype=00&authbasic=[AUTH]
Hikvision	/Streaming/Channels/101
Panasonic	/ONVIF/MediaInput
GeoVision	/Streaming/Channels/101
LTS Security	/Streaming/Channels/101
Ubiquiti	/live/ch00_0
Foscam	/videoMain
Vivotek	/live.sdp
Speco	/Streaming/Channels/101

Ґрунтуючись на результатах пошукового запиту у Shodan до глобальної мережі інтернет підключено близько 4439849 пристроїв з відкритим 554 портом (рис. 1.3).



Рисунок 1.3 – Результатах пошукового запиту у Shodan по відкритому 554 порту

З цих 4439849 пристроїв близько 1221387 повернули код “200 OK”, що може вказувати на повну відсутність будь-якого захисту, навіть стандартного імені користувача та пароля (рис. 1.4).



Рисунок 1.4 – Результатах пошукового запиту Shodan по відкритому порту 554 і коду відповіді 200

1.5 Протокол RTSP

За синтаксисом та операціями RTSP схожий на HTTP. Однак між протоколами RTSP та HTTP є ряд суттєвих відмінностей. Одне з основних у тому, що у першому сервер та клієнт можуть генерувати запити. Як приклад, відеосервер може надіслати запит для встановлення параметрів відтворення певного відеопотоку. Також RTSP протоколом передбачається, що керування станом чи зв'язком має виконуватися сервером, тоді як HTTP взагалі жодного відношення до цього не має. Нарешті, дані RTSP можуть передаватися іншими

протоколами поза основною смугою, наприклад RTP, що неможливо у разі HTTP.

RTSP-повідомлення надсилаються окремо від мультимедійного потоку. Для них використовується з'єднання за спеціальним портом, за замовчуванням з номером 554. Запит на сервер надсилається у текстовому вигляді у форматі: “<абсолютна_адреса>[/медіавміст] <версія_протоколу>”. Разом із запитом можуть бути передані додаткові службові поля (на нових рядках запиту).

Методи протоколу:

- describe, запит опису вмісту, наприклад, у форматі SDP;
- options, запит підтримуваних методів;
- play, запит початку мовлення;
- pause, запит тимчасової зупинки мовлення;
- record, запит на записування вмісту сервером;
- redirect, перенаправлення на інший вміст;
- setup, запит на встановлення транспортного механізму для вмісту;
- announce, оновлення даних опису вмісту;
- get_parameter, запит зазначених параметрів сервера;
- set_parameter, встановлення параметрів сервера;
- teardown, зупинка потоку та звільнення ресурсів [5].

1.5.1 Метод DESCRIBE

Запит DESCRIBE включає URL-адресу RTSP (rtsp://...) та тип даних відповіді, які можуть бути оброблені. Ця відповідь включає опис уявлення, зазвичай у форматі протоколу опису сеансу (SDP). Крім того, в описі презентації перераховані медіапотоки, керовані сукупною URL-адресою. У типовому випадку є один медіапоток для аудіо і відеопотоку. URL-адреси медіапотоку виходять безпосередньо з полів керування SDP, або шляхом додавання поля керування SDP до сукупної URL-адреси (рис. 1.5) [5].

```

C->S: DESCRIBE rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 2

S->C: RTSP/1.0 200 OK
      CSeq: 2
      Content-Base: rtsp://example.com/media.mp4
      Content-Type: application/sdp
      Content-Length: 460

      m=video 0 RTP/AVP 96
      a=control:streamid=0
      a=range:npt=0-7.741000
      a=length:npt=7.741000
      a=rtpmap:96 MP4V-ES/5544
      a=mimetype:string;"video/MP4V-ES"
      a=AvgBitRate:integer;304018
      a=StreamName:string;"hinted video track"
      m=audio 0 RTP/AVP 97
      a=control:streamid=1
      a=range:npt=0-7.712000
      a=length:npt=7.712000
      a=rtpmap:97 mpeg4-generic/32000/2
      a=mimetype:string;"audio/mpeg4-generic"
      a=AvgBitRate:integer;65790
      a=StreamName:string;"hinted audio track"

```

Рисунок 1.5 – Приклад запиту DESCRIBE і відповіді на нього

1.5.2 Метод OPTIONS

Запит OPTIONS повертає типи запитів, які сервер приймає (рис. 1.6) [5].

```

C->S: OPTIONS rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 1
      Require: implicit-play
      Proxy-Require: gzipped-messages

S->C: RTSP/1.0 200 OK
      CSeq: 1
      Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE

```

Рисунок 1.6 – Приклад запиту OPTIONS і відповіді на нього

1.5.3 Метод PLAY

Запит PLAY викликає відтворення одного або всіх медіапотоків. Запити на відтворення можна об'єднувати, надсилаючи кілька запитів на відтворення. URL-адреса може бути сукупною URL-адресою (для відтворення всіх медіапотоків) або URL-адресою одного медіапотоку (для відтворення лише цього потоку). Можна вказати діапазон. Якщо діапазон не вказано, потік відтворюється з початку і до кінця, або якщо потік поставлений на паузу, він відновлюється з того місця, де він був призупинений (рис. 1.7) [5].

```
C->S: PLAY rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 4
      Range: npt=5-20
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 4
      Session: 12345678
      RTP-Info: url=rtsp://example.com/media.mp4/streamid=0;seq=9810092;rtptime=3450012
```

Рисунок 1.7 – Приклад запиту PLAY і відповіді на нього

1.5.4 Метод PAUSE

Запит PAUSE тимчасово зупиняє один або всі медіапотоки, тому його можна відновити за допомогою запиту PLAY. Запит містить сукупну URL-адресу або URL-адресу медіапотоку. Параметр діапазону в запит PAUSE вказує, коли зробити паузу. Коли параметр діапазону опущений, пауза відбувається негайно та на невизначений термін (рис. 1.8) [5].

```
C->S: PAUSE rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 5
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 5
      Session: 12345678
```

Рисунок 1.8 – Приклад запиту PAUSE і відповіді на нього

1.5.5 Метод RECORD

Цей метод ініціює запис діапазону мультимедійних даних відповідно до опису презентації. Позначка часу відображає час початку та закінчення (UTC). Якщо часовий діапазон не вказано, використовуйте час початку або закінчення, зазначений у описі презентації. Якщо сеанс розпочався, негайно почніть запис. Сервер вирішує, чи зберігати записані дані під запитом URI або під іншим URI. Якщо сервер не використовує URI запит, відповідь повинна бути 201 і містити об'єкт, який описує стан запиту і посиляється на новий ресурс, а також заголовок Location (рис. 1.9) [5].

```
C->S: RECORD rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 6
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 6
      Session: 12345678
```

Рисунок 1.9 – Приклад запиту RECORD і відповіді на нього

1.5.6 Метод SETUP

Запит SETUP вказує, як повинен транспортуватись один медіапоток. Це необхідно зробити до надсилання запиту PLAY. Запит містить URL-адресу медіапотoku та специфікатор транспорту. Цей специфікатор зазвичай включає локальний порт для отримання даних RTP (аудіо або відео) та інший порт даних RTSP (метаінформація). Відповідь сервера зазвичай підтверджує вибрані параметри та заповнює відсутні частини, такі як вибрані сервером порти. Кожен медіапотік має бути налаштований за допомогою SETUP, перш ніж може бути надіслано запит на сукупне відтворення (рис. 1.10) [5].

```

C->S: SETUP rtsp://example.com/media.mp4/streamid=0 RTSP/1.0
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8000-8001

S->C: RTSP/1.0 200 OK
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8000-8001;server_port=9000-9001;ssrc=1234ABCD
      Session: 12345678

C->S: SETUP rtsp://example.com/media.mp4/streamid=1 RTSP/1.0
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8002-8003
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8002-8003;server_port=9002-9003;ssrc=1234ABCD
      Session: 12345678

```

Рисунок 1.10 – Приклад запиту SETUP і відповіді на нього

1.5.7 Метод ANNOUNCE

Метод ANNOUNCE служить двом цілям:

- при надсиланні з клієнта на сервер ANNOUNCE надсилає опис презентації або мультимедійного об'єкта, ідентифікованого URL-адресою запиту на сервер;
- при надсиланні з сервера на клієнт ANNOUNCE оновлює опис сеансу в режимі реального часу. Якщо до презентації додається новий медіапотік (наприклад, під час прямої трансляції), слід повторно надіслати весь опис презентації, а не лише додаткові компоненти, щоб компоненти можна було видалити (рис. 1.11) [5].

```

C->S: ANNOUNCE rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 7
      Date: 23 Jan 1997 15:35:06 GMT
      Session: 12345678
      Content-Type: application/sdp
      Content-Length: 332

      v=0
      o=mhandley 2890844526 2890845468 IN IP4 126.16.64.4
      s=SDP Seminar
      i=A Seminar on the session description protocol
      u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
      e=mjh@isi.edu (Mark Handley)
      c=IN IP4 224.2.17.12/127
      t=2873397496 2873404696
      a=recvonly
      m=audio 3456 RTP/AVP 0
      m=video 2232 RTP/AVP 31

S->C: RTSP/1.0 200 OK
      CSeq: 7

```

Рисунок 1.11 – Приклад запиту ANNOUNCE і відповіді на нього

1.5.8 Метод GET_PARAMETER

Запит GET_PARAMETER отримує значення параметра презентації або потоку, зазначеного в URI. Зміст відповіді та відповіді залишається на реалізацію. GET_PARAMETER без тіла об'єкта можна використовувати для перевірки працездатності клієнта або сервера ("ping") (рис. 1.12) [5].

```

S->C: GET_PARAMETER rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 9
      Content-Type: text/parameters
      Session: 12345678
      Content-Length: 15

      packets_received
      jitter

C->S: RTSP/1.0 200 OK
      CSeq: 9
      Content-Length: 46
      Content-Type: text/parameters

      packets_received: 10
      jitter: 0.3838

```

Рисунок 1.12 – Приклад запиту GET_PARAMETER і відповіді на нього

1.5.9 Метод SET_PARAMETER

Цей метод вимагає встановлення параметра для представлення або потоку, зазначеного URI (рис. 1.13) [5].

```

C->S: SET_PARAMETER rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 10
      Content-length: 20
      Content-type: text/parameters

      barparam: barstuff

S->C: RTSP/1.0 451 Invalid Parameter
      CSeq: 10
      Content-length: 10
      Content-type: text/parameters

      barparam

```

Рисунок 1.13 – Приклад запиту SET_PARAMETER і відповіді на нього

1.5.10 Метод TEARDOWN

Запит TEARDOWN використовується для завершення сеансу. Він зупиняє всі потоки мультимедіа та звільняє всі дані, пов'язані з сеансом, на

сервері (рис. 1.14) [5].

```
C->S: TEARDOWN rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 8
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 8
```

Рисунок 1.14 – Приклад запиту TEARDOWN і відповіді на нього

1.6 Аналіз вимог

При огляді предметної області було складено оптимальні нефункціональні та функціональні вимоги до програмного забезпечення, які дозволять йому тривалий час залишатися актуальним та конкурувати з можливим аналогам у майбутньому.

Нефункціональні вимоги:

- програмний продукт повинен підтримувати операційні системи Windows та Linux, 32 та 64 розрядностей;
- виконувати виявлення найпоширенішої вразливості, що зачіпає більшість виробників;
- мати простий та зрозумілий графічний інтерфейс.

Функціональні вимоги:

- приймати як одиночні ір-адреси та їх діапазони;
- вибір кількості потоків для виконання завдання виявлення вразливостей;
- пошук систем відеоспостереження та попереднє виявлення вразливостей;
- можливість поставити процес на паузу;
- можливість завершити процес із збереженням непройдених ір-адрес;
- відстеження результатів виконання процесу;

- можливість виведення зображення з вразливої системи відеоспостереження;
- перегляд результатів тестування;
- видалення результатів тестування;
- продовжити виявлення вразливостей у знайдених систем відеоспостереження;
- перевірити вразливі системи відеоспостереження на відсутність вразливості;
- редагування списку, за яким виконується виявлення вразливостей.

Розповсюджуватись програма повинна як проект з відкритим вихідним кодом для того, щоб користувачі могли переконатися в її безпеці перед використанням.

1.7 Висновок за розділом 1

У розділі було розглянуто поняття системи відеоспостереження, їх вразливості, експлуатація вразливостей, протокол RTSP та вимоги до програмного забезпечення.

З огляду вразливостей систем відеоспостереження можна виділити найпоширенішу, якою є використання імені користувача та пароля, що встановлюються виробниками на етапі виробництва. При огляді стандартних імені користувача та пароля популярних виробників наведених у таблиці 1.1, можна скласти їхній рейтинг наведений у таблиці 1.3.

Таблиця 1.3 – Рейтинг стандартних комбінацій імен користувачів та паролів

Місце	Ім'я користувача	Пароль
1	2	3
1	admin	admin

Продовження таблиці 1.3

1	2	3
2	admin	12345
3	admin	123456
4	admin	1234
5	admin	-
6	ubnt	ubnt
7	root	pass
8	root	-

Виконувати виявлення вразливості стандартного імені користувача та пароля найкраще за допомогою протоколу RTSP через запит “DESCRIBE”, оскільки цей протокол за замовчуванням для всіх систем відеоспостереження використовує 554 порти, необхідно тільки знати URN виробника. При огляді URN популярних виробників наведених у таблиці 1.2 та стандартних імені користувача та пароля наведених у таблиці 1.1, можна скласти рейтинг комбінацій який наведено в таблиці 1.4.

Таблиця 1.4 – Рейтинг комбінацій

Місце	Ім'я користувача	Пароль	URN
1	2	3	4
1	admin	12345	/Streaming/Channels/101
2	admin	admin	/Streaming/Channels/101
3	root	pass	/axis-media/media.amp
4	admin	admin	/cam/realmonitor?channel=1&subtype=00&authbasic=[AUTH]
5	admin	1234	/Streaming/Channels/101
6	ubnt	ubnt	/live/ch00_0
7	admin	123456	/stream1

Продовження таблиці 1.4

1	2	3	4
8	admin	admin	/primarystream
9	admin	12345	/ONVIF/MediaInput
10	admin	-	/videoMain

Такий спосіб виявлення вразливостей дозволяє спростити розробку та супровід програми, зробивши її більш універсальною, так при появі нових виробників або моделей достатньо буде просто внести нову комбінацію до списку, за допомогою якого виконується перебір.

2 ВИБІР ТА ОБГРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЩО ПРОЕКТУЄТЬСЯ

2.1 Вибір середовища розробки

Для створення програми буде використано інтегроване середовище розробки Microsoft Visual Studio 2022.

Особливості Visual Studio 2022:

- дозволяє швидко та легко створювати хмарні програми за допомогою Azure, який надає можливість розробки, виконання програм та зберігання даних на серверах, розташованих у розподілених дата-центрах;
- колаборація в реальному часі за допомогою Live Share, яка відкриває нові можливості для співпраці, обміну ідеями, парного програмування та аналізу коду;
- підсистема AI IntelliCode допомагає робити правильні дії в потрібному місці в потрібний час, поміщає найкорисніші методи на початок списку завершення, тим самим заощаджуючи час;
- включає потужну підтримку Git та GitHub, фіксація коду, відправлення запитів та злиття гілок [11];
- велика бібліотека розширень, наприклад Qt Visual Studio Tools для зручності роботи з файлами розширення qml;
- зручне управління пакетами PyPi.

Порівняння середовищ розробки наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння середовищ розробки

	MS Visual Studio	Qt Creator IDE	Eclipse IDE	Spider IDE
1	2	3	4	5
Вбудований відладчик	Так	Так	Так	Так

Продовження таблиці 2.1

1	2	3	4	5
Переносимість середовища на різні платформи	Так	Так	Так	Так
Встановлене ПЗ на підприємстві та наявність ліцензії	Так	Так	Ні	Ні
Досвід роботи	Так	Так	Ні	Ні
Безкоштовний доступ	Так	Так	Так	Так

2.2 Вибір мови програмування

Програма буде реалізована на мові Python.

Переваги мови Python:

- швидкість розробки є дуже високою, порівняно зі строго типізованими мовами, такими як C, C++ та Java або компілюючими, Python у багато разів підвищує продуктивність праці розробника. Об'єм програмного коду Python майже у всіх випадках становить третину або навіть п'яту частину еквівалентного програмного коду мовою Java або C++. Це дозволяє зменшити обсяг введення з клавіатури, зменшити витрати часу на налагодження та об'єм трудовитрат на супровід. Також, програми на Python запускаються відразу, уникаючи затяжних етапів компіляції та зв'язування, необхідні в деяких інших мовах програмування, що ще більше збільшує продуктивність праці програміста;

- переносність програм, більшість програм на мові Python виконується без змін на всіх основних платформах. Перенесення програмного коду з операційної системи Windows у Linux зазвичай полягає у звичайному копіюванні файлів програм з одного комп'ютера на інший. Більше того, Python надає велику кількість можливостей зі створення графічних інтерфейсів, програм для роботи з базами даних, веб-додатків і багатьох інших типів

програм. Навіть інтерфейси операційних систем, включаючи спосіб запуску програм та обробку каталогів у мові Python реалізовані переносним способом;

– бібліотеки підтримки Python включають велику кількість зібраних і переносимих функціональних можливостей, відомих як стандартна бібліотека, ця бібліотека надає велику кількість можливостей, затребуваних у програмі, починаючи від мережеских функцій і закінчуючи пошуком тексту за шаблоном. Також Python підтримує розширення як за рахунок бібліотек створених сторонніми розробниками, так і за рахунок власних бібліотек. Серед сторонніх розробок можна назвати програмування математичних обчислень, інструменти створення веб-сайтів, розробку ігрових програм, доступ до послідовного порту та багато іншого [6].

2.3 Вибір бібліотеки для створення інтерфейсу

Інтерфейс програми буде реалізовано за допомогою декларативної мови програмування QML, бібліотеки PyQt5.

QML – це мова специфікації інтерфейсу користувача і розмітки, який дозволяє розробникам і дизайнерам створювати високопродуктивні, плавні анімаційні та візуально привабливі програми.

QML надає добре читаний, декларативний, JSON-подібний синтаксис, який підтримує імперативні вирази JavaScript у поєднанні з динамічним зв'язуванням атрибутів [7].

Використання JavaScript дозволяє обробляти деякі дії в самому інтерфейсі, що дозволяє мінімізувати звернення до нього з коду Python.

2.4 Вибір бібліотеки для розподілу процесів

Процес сканування діапазону ір-адрес та виявлення вразливостей шляхом перебору значень буде поділятися на асинхронні процеси за допомогою бібліотеки “multiprocessing”.

Переваги бібліотеки multiprocessing:

- окремий простір пам'яті;
- використовує переваги кількох процесорів та ядер;
- обходить обмеження GIL для cPython;
- дочірні процеси можна переривати.

Найголовнішим плюсом використання процесів замість потоків є обхід обмеження GIL, яке по черзі передає управління одному потоку, а всі інші блокує тим самим лише збільшуючи час виконання.

2.5 Вибір способу зберігання даних

Зберігання даних виконуватиметься за допомогою серіалізації та десеріалізації класів, засобами бібліотеки “pickle”.

Переваги такого зберігання даних:

- швидке читання та запис;
- дані зберігаються із потрібними типами даних, зчитуються як готовий об'єкт.

Єдиним недоліком такого зберігання даних можна вважати прив'язку до мови Python, але його можна не враховувати, оскільки специфіка інформації, що зберігається, не передбачає її поширення та тривале зберігання.

2.6 Висновок за розділом 2

У розділі було проаналізовано та обрано:

- середовище розробки Visual Studio 2022;
- мову програмування Python;
- бібліотеку PyQt5 для створення інтерфейсу за допомогою QML;
- зберігання даних шляхом серіалізації та десеріалізації об'єктів за допомогою бібліотеки pickle.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Опис файлів та папок програми

Файлова структура програмного забезпечення наведена рисунку 3.1.

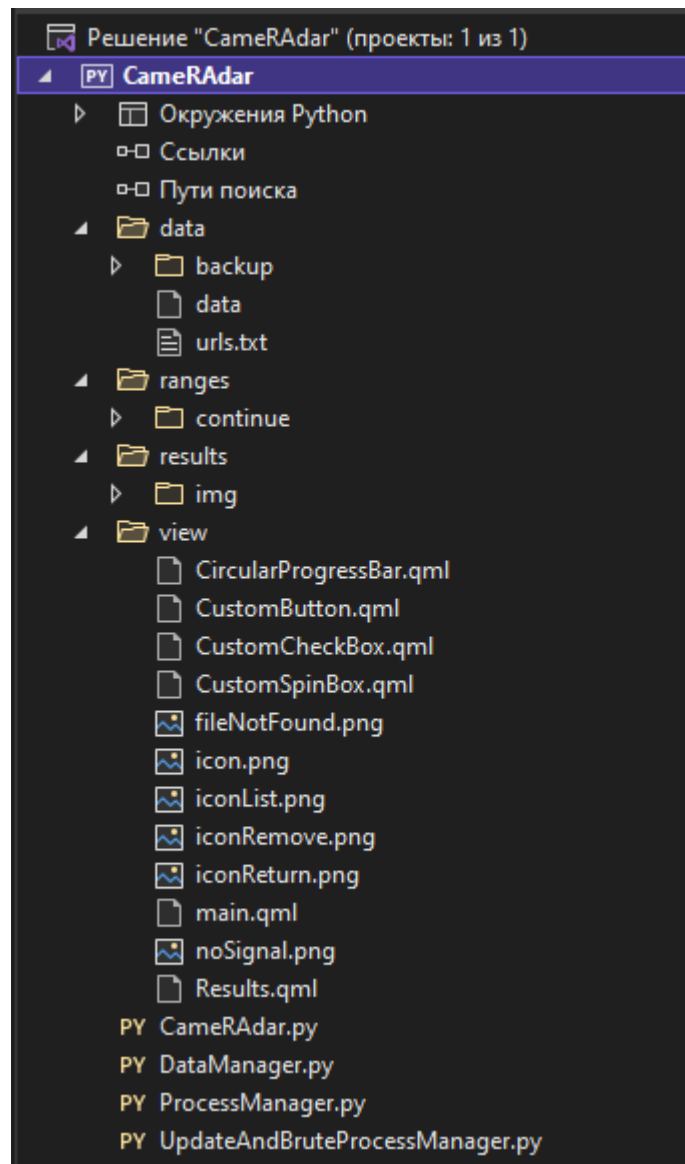


Рисунок 3.1 – Структура файлів проекту

CameRAadar.py - основний файл програми що виконує запуск інтерфейсу та взаємодію з ним.

Модулі програми:

- DataManager.py – модуль для роботи з даними;

- ProcessManager.py – модуль для управління процесами пошуку систем відеоспостереження та виявлення в них вразливостей;
- UpdateAndBruteProcessManager.py – модуль для управління процесами перевірки усунених вразливостей та продовження їх виявлення.

Інтерфейс:

- main.qml – головне вікно за допомогою якого виконується керування процесом виявлення вразливостей у системах відеоспостереження;
- Results.qml – друге вікно за допомогою якого виконується перевірка актуальності виявлених вразливостей, продовження їх виявлення, перегляд результатів та видалення результатів.

Модулі інтерфейсу:

- CircularProgressBar.qml – круглий прогресбар;
- CustomButton.qml – стилізована кнопка;
- CustomCheckBox.qml – стилізований чекбокс;
- CustomSpinBox.qml – стилізований спінбокс.

data – файл для збереження результатів.

urls.txt – файл із комбінаціями.

Знімки із систем відеоспостереження зберігаються в папці “results\img”.

Непройдені ір-адреси при зупинці процесу зберігаються в папці “ranges\continue”.

3.2 Алгоритм функціонування програми

Програма націлена на користувачів, яким необхідно провести перевірку своєї системи відеоспостереження на наявність у ній вразливості.

Алгоритм функціонування програми зображений на рисунку 3.2

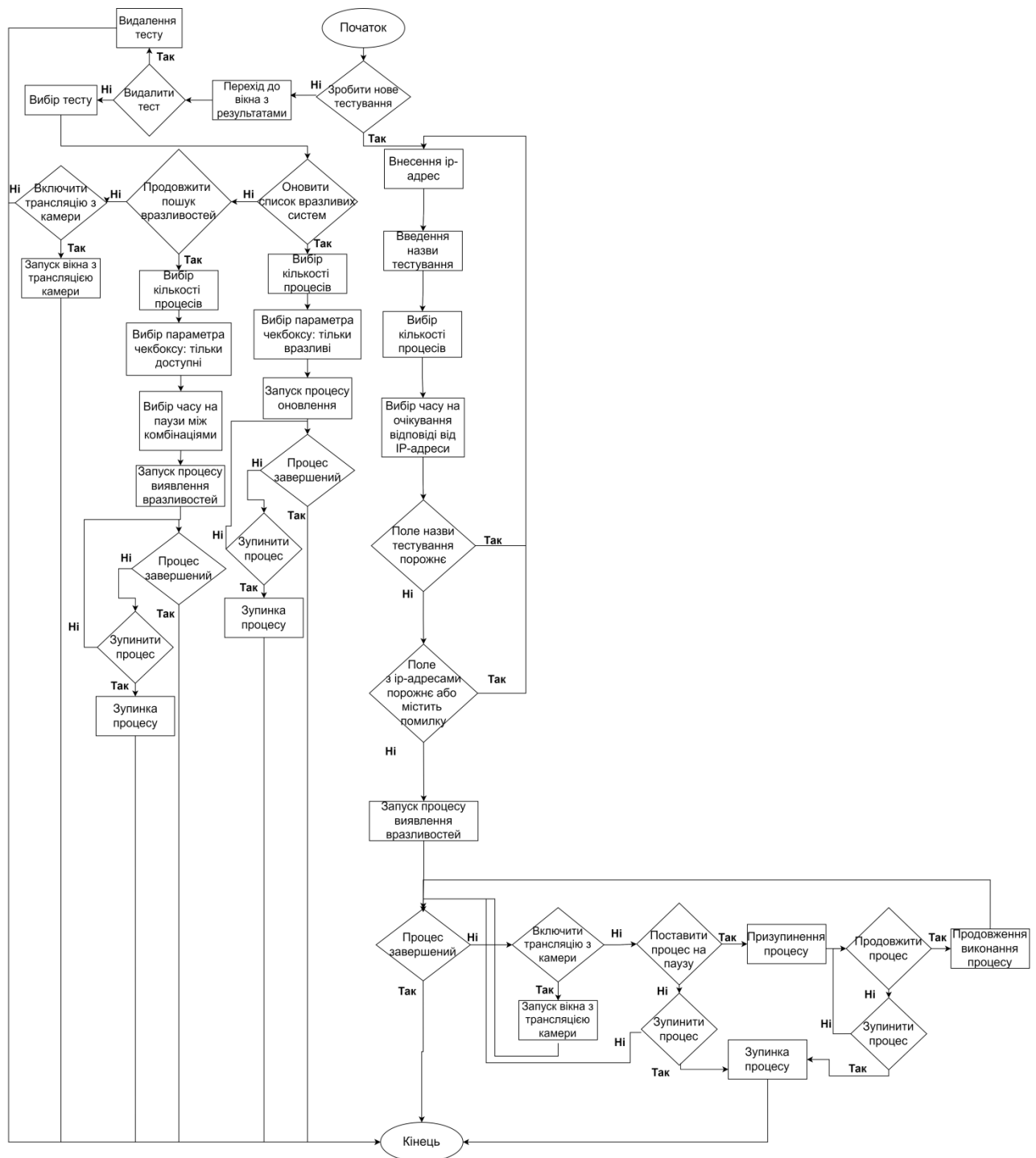


Рисунок 3.2 – Діаграма алгоритму функціонування програми

Точною входу є запуск файлу CameRAadar.py після чого користувачеві пропонується внести список ір-адрес як поодинокі, так і у форматі діапазону "від-до", задати назву тестування що проводиться та запустити процес.

У ході виконання процесу, користувач може поставити його на паузу, після чого продовжити його або завершити, також за наявності результату у

вигляді знімка з камери, користувач може включити трансляцію клікнувши по зображенню.

Для подальшої взаємодії з результатами користувач може перейти до вікна Results, в якому він може:

- видаляти результати тестування;
- переглядати результати тестування;
- перевірити, чи усунено вразливість;
- продовжити виявлення вразливостей;
- переглядати трансляцію.

3.3 Розгляд модуля DataManager.py

Модуль DataManager.py надає інтерфейс для роботи з даними, збереження, видалення, зміна.

Модуль DataManager.py містить у собі три класи, які формують структуру даних, опис яких наведено в таблиці 3.1.

Таблиця 3.1 – Опис класів модуля DataManager.py які формують структуру даних

Клас	Атрибути	Опис
1	2	3
CamInBrute	ip	IP-адреса системи відеоспостереження.
	block	Встановлюється True, за відсутності з'єднання або відповіді.
	urls	Список пройдених комбінацій.

Продовження таблиці 3.1

1	2	3
Result	ip	IP-адреса системи відеоспостереження.
	url	Комбінація за допомогою якої вдалося отримати доступ.
	img	Встановлюється True при вдалому отриманні зображення.
	alive	Встановлюється False за відсутності з'єднання або відповіді.
TestBlock	name	Ім'я тесту.
	inBrute	Список класів CamInBrute.
	results	Список класів Result.

Опис методів основного класу DataManager, модуля DataManager.py наведено на таблиці 3.2.

Таблиця 3.2 – Опис методів основного класу DataManager

Метод	Аргументи	Опис
1	2	3
SetUp		Перевіряє наявність необхідних директорій та файлів даних, за їх відсутності створює їх.
ReadUrls		Повертає список комбінацій із файлу.
addTestBlock	name	Додає новий об'єкт "TestBlock" із зазначеною назвою у файл даних, якщо блоку з таким ім'ям немає.
addToBrute	testName, ip, url, block = False,	Створює та додає клас CamInBrute до списку inBrute класу TestBlock за вказаним ім'ям тесту та зберігає зміни.

Продовження таблиці 3.2

1	2	3
addPassedUrl	testName, ip, url	Додає пройдену комбінацію для зазначеної ір-адреси за вказаною назвою тесту та зберігає зміни.
addResult	testName, ip, url, img = True, alive = True	Створює та додає клас Result до списку results класу TestBlock та зберігає зміни.
setBlock	testName, ip	Встановлює значення False атрибуту block класу CamInBrute за вказаною ір-адресою та назвою тесту та зберігає зміни.
setNotAlive	testName, ip	Встановлює значення False атрибуту alive класу Result за вказаною ір-адресою та назвою тесту та зберігає зміни.
setYseImg	testName, ip	Встановлює значення True атрибуту img класу Result за вказаною ір-адресою та назвою тесту та зберігає зміни.
setAlive	testName, ip	Встановлює значення True атрибуту alive класу Result за вказаною ір-адресою та назвою тесту та зберігає зміни.

Продовження таблиці 3.2

1	2	3
getResultIpWithImg	testName, alive = True, yesImg = True, getQuantity = False, totallmg = False	Залежно від значень аргументів повертає: - результати із зображенням та можливістю доступу; - результати без зображення але з доступом; - результати із зображенням але без з доступу; - результати без зображення та доступу; - кількість, якщо аргумент getQuantity дорівнює True; - загальна кількість всіх елементів, при аргументі totallmg що дорівнює True.
getResultAlive	testName, yes = True, getQuantity = False	Повертає всі результати до яких є доступ або їх кількість, якщо аргумент getQuantity дорівнює True.
getResultUrl	testName, ip	Повертає URL вказаної ip-адреси.
getResultAll	testName	Повертає всі результати за назвою тесту.
getResultQuantity	testName	Повертає кількість результатів за назвою тесту.

Продовження таблиці 3.2

1	2	3
getBruteWithNoPassedUrl	testName, url, availableOnli = True	Повертає ір-адреси систем відеоспостереження, які не були перевірені зазначеною комбінацією, за замовчуванням повертає ір-адреси до яких є доступ.
getBruteQuantity	testName	Повертає кількість систем відеоспостереження в яких не було виявлено вразливостей.
getNoPassedUrlList	testName, urlList	Повертає список непройдених комбінацій.
getBruteBlockForCheck	testName	Повертає недоступні IP-адреси з першою непройденою комбінацією.
removeTestBlock	testName	Видаляє результати тестування.
backup		Виконує копіювання файлу даних у папку backup.

3.4 Розгляд модуля ProcessManager.py

Модуль ProcessManager.py надає інтерфейс для створення та управління процесами, які виконують пошук систем відеоспостереження та виявлення в них вразливостей.

Модуль ProcessManager.py має десять функцій, призначених для запуску в окремих процесах, їх опис наведено в таблиці 3.3.

Таблиця 3.3 – Опис функцій модуля ProcessManager.py

Функція	Аргументи	Опис
1	2	3
genmsg_DESCRIBE	url,seq,userAgent, authSeq	Генерує рядок запиту DESCRIBE, протоколу RTSP.
generateAuthString	username,password, realm, method, uri, nonce	Генерує рядок з даними для авторизації.
connect	ip, log, uri, timeout, step	Перевіряє відкритість 554 порту, якщо порт відкритий виконує спробу авторизації та обробку відповіді.
capture	ip, uri	Підключається до системи відеоспостереження для захоплення та збереження зображення.
resCatcher	testName, queueRes, queueResImg, cOpen, cAuthorized, cCaptured	Приймає та обробляє отримані результати.
stopCatcher	testName, queueStop	При зупинці процесу сканування користувачем, зберігає непройдені іп-адреси у файл txt з назвою тестування в папку “ranges\continue”.
timeLeftCal	t, resT, countIp, stopPauseSignal, totalIp	Розраховує час що залишився до кінця.

Продовження таблиці 3.3

1	2	3
sendIp	ips, queueIp, processQuantity	Відправляє ip-адреси зі списку ips у чергу queueIp, при досягненні кінця списку ips додає в чергу queueIp вказану в змінній processQuantity кількість значень None.
scan	urls, queueRes, queueStop, queueIp, counterPassedIps, stopPauseSignal, t, timeout	Виконує пошук систем відеоспостереження та виявлення в них уразливостей.

Опис атрибутів класу ProcessManager наведено у таблиці 3.4.

Таблиця 3.4 – Опис атрибутів класу ProcessManager

Атрибут	Опис
1	2
stopPauseSignal	Загальна для всіх процесів змінна для відстеження ними сигналу про припинення або повне зупинення.
counterPassedIps	Загальна змінна для всіх процесів scan для підрахунку пройдених IP-адрес.
counterOpen	Загальна змінна для процесу resCatcher для підрахунку ip-адрес систем відеоспостереження в яких не було виявлено вразливостей.
counterAuthorized	Загальна змінна для процесу resCatcher для підрахунку ip-адрес систем відеоспостереження в яких була виявлена вразливість.

Продовження таблиці 3.4

1	2
counterCaptured	Загальна змінна для процесу resCatcher для підрахунку ір-адрес систем відеоспостереження з яких вдалося захопити зображення.
resT	Загальна змінна для процесу timeLeftCal якій надається результат розрахунку часу, що залишився.
timeProc	Змінна для зберігання процесу timeLeftCal.
sendIpProc	Змінна для зберігання процесу sendIp.
processList	Список для зберігання процесів scan.
resultsCatcherProc	Змінна для зберігання процесу resCatcher.
queueRes	Черга для передачі результатів з процесів scan в процес resCatcher.
queueResImg	Черга для передачі об'єктів з процесу resCatcher в основний процес для виведення отриманих зображень в інтерфейс користувача.
queueStop	Черга для передачі непройдених ір-адрес з процесів scan в процес stopCatcher.
queueIp	Черга для передачі ір-адрес із процесу sendIp процесам scan.
stopProc	Змінна для зберігання процесу stopCatcher.
ipsNum	Зберігає кількість переданих IP-адрес.

Опис методів класу ProcessManager наведено у таблиці 3.5.

Таблиця 3.5 – Опис методів класу ProcessManager

Метод	Аргументи	Опис
1	2	3
clear		Очищає всі змінні.
getImgData		Повертає елемент із черги queueResImg.

Продовження таблиці 3.5

1	2	3
<code>__init__</code>	<code>testName,</code> <code>nProcess,</code> <code>timeout,</code> <code>ips, urls</code>	Конструктор класу створює і запускає всі необхідні процеси.
<code>getTimeLeft</code>		Повертає час у хвилинах до закінчення проходження всіх ір-адрес.
<code>getOpen</code>		Повертає кількість ір-адрес з відкритим 554 портом.
<code>getCaptured</code>		Повертає кількість систем відеоспостереження, з яких було отримано зображення.
<code>getAuthorized</code>		Повертає кількість вразливих систем відеоспостереження.
<code>getIpLeft</code>		Повертає залишкову кількість ір-адрес.
<code>getProgress</code>		Повертає прогрес процесу.
<code>pause</code>		Встановлює сигнал паузи для процесів.
<code>cont</code>		Прибирає сигнал паузи для процесів.
<code>stop</code>		Зупиняє всі процеси.
<code>isImg</code>		Повертає True якщо у черзі <code>queueResImg</code> є елементи.

3.5 Розгляд модуля `UpdateAndBruteProcessManager.py`

Модуль `UpdateAndBruteProcessManager.py` надає інтерфейс для створення та керування процесами оновлення результатів та продовження виявлення вразливостей.

Модуль `UpdateAndBruteProcessManager.py` має вісім функцій, призначених для запуску в окремих процесах, їх опис наведено в таблиці 3.6.

Таблиця 3.6 – Опис функцій модуля UpdateAndBruteProcessManager.py

Функція	Аргументи	Опис
1	2	3
processPutGetImgUpdate	testName, numProcess, queueImg, queueImgUpdate, queueResImgUpdate, aliveOnly, resultImgUpdateProgress, endSignal	Передає дані систем відеоспостереження процесам, що виконують перевірку актуальності вразливості, після чого приймає результат.
processImgUpdate	queueImgUpdate, queueResImgUpdate, stopSignal	Передає дані систем відеоспостереження процесам, що виконують перевірку актуальності вразливості, після чого приймає та обробляє результат.
processPutGetBrute	testName, numProcess, timeout, queueImg, queueBrute, queueResBrute, availableOnly, brute_available, brute_block, brute_combinations_left, brute_total, stopSignal, endSignal	Передає ip-адреси систем відеоспостереження із зазначеною комбінацією процесам, що виконують запит, після чого приймає та обробляє результат.

Продовження таблиці 3.6

1	2	3
processBrute	queueBrute, queueResBrute	Приймає ір-адреси із зазначеною комбінацією, виконує запит та передає результат.
genmsg_DESCRIBE	url,seq,userAgent,authSeq	Генерує рядок запиту DESCRIBE, протоколу RTSP.
generateAuthString	username,password,realm,method,uri,nonce	Генерує рядок з даними для авторизації
capture	ip, uri, checkIsAlive = True	Підключається до системи відеоспостереження для захоплення та збереження зображення.
connect	ip, log, uri	Перевіряє відкритість 554 порту, якщо порт відкритий виконує спробу авторизації та обробку відповіді.

Опис атрибутів класу UpdateAndBruteProcessManager наведено у таблиці 3.7.

Таблиця 3.7 – Опис атрибутів класу UpdateAndBruteProcessManager

Атрибути	Опис
1	2
bruteProcessList	Список для зберігання процесів processBrute.

Продовження таблиці 3.7

1	2
queueImg	Черга для передачі об'єктів з процесу processPutGetImgUpdate в основний процес для виведення отриманих зображень в інтерфейс користувача.
queueResImgUpdate	Черга для передачі об'єктів з процесу processImgUpdate в процес processPutGetImgUpdate.
queueResBrute	Черга для передачі об'єктів з процесу processBrute в processPutGetBrute.
queueImgUpdate	Черга для передачі об'єктів з процесу processPutGetImgUpdate до процесу processImgUpdate.
queueBrute	Черга для передачі об'єктів із процесу processPutGetBrute в процес processBrute.
stopSignal	Загальна для всіх процесів змінна для передачі сигналу про повну зупинку.
endSignal	Після завершення всіх процесів встановлюється значення 1.
imgUpdateProcessList	Список для зберігання процесів processImgUpdate.
imgUpdatePutGetProcess	Змінна для зберігання процесу processPutGetImgUpdate.
brutePutGetProcess	Змінна для зберігання процесу processPutGetBrute.
brute_available	Загальна змінна для процесу processBrute для підрахунку доступних IP-адрес систем відеоспостереження.
brute_block	Загальна змінна для процесу processBrute для підрахунку недоступних IP-адрес систем відеоспостереження.

Продовження таблиці 3.7

1	2
brute_combinations_left	Загальна змінна для процесу processBrute, для підрахунку комбінацій, що залишилися.
brute_total	Загальна змінна для процесу processBrute, для підрахунку ір-адрес систем відеоспостереження в яких не виявлено вразливостей.

Опис методів класу UpdateAndBruteProcessManager наведено у таблиці 3.8.

Таблиця 3.8 – Опис методів класу UpdateAndBruteProcessManager

Метод	Аргументи	Опис
1	2	3
updateImgRun	testName, numProcess, aliveOnly = True	Запускає процеси для перевірки усунення вразливостей.
bruteRun	testName, numProcess, timeout, availableOnly = True	Запускає процеси перебору комбінацій для виявлення вразливостей.
updateImgStop		Зупиняє процеси перевірки усунення вразливостей.
updateImgIsEnd		Повертає True якщо процеси перевірки усунення вразливостей завершені.

Продовження таблиці 3.8

1	2	3
bruteStop		Зупиняє процеси перебору комбінацій для виявлення вразливостей.
bruteIsEnd		Повертає True якщо процеси перебору комбінацій завершені.
getBruteAvailable		Повертає кількість систем відеоспостереження доступних для виявлення вразливостей.
getBruteBlock		Повертає кількість систем відеоспостереження недоступних для виявлення вразливостей.
getBruteCombinationsLeft		Повертає кількість комбінацій, що залишилися.
getBruteTotal		Повертає кількість систем відеоспостереження в яких не було виявлено вразливостей.
isImg		Повертає True якщо у черзі queueImg є елементи.
getImgData		Повертає елемент із черги queueResImg.

3.6 Розгляд модуля CameRAadar.py

Модуль CameRAadar.py основний файл програми виконує запуск інтерфейсу, переходи між ними, запуск та управління процесами, оновлення значень інтерфейсу.

Функції модуля CameRAadar.py описані у таблиці 3.9.

Таблиця 3.9 – Опис функцій модуля CameRAdar.py

Назва функції	Аргументи	Опис
formattingNumber	numStr	Розділяє символи числа пробілом через кожні три символи, приклад: з 150545 робить 150 545.
isCorrectIpFormat	s	Повертає True якщо ір-адреса коректна.
separateSingleIpFromRange	l	Відокремлює одиночні ір-адреси від діапазону ір-адрес.
ipRangeScan	ranges	Повертає список ір-адрес за переданими діапазонами.
checkInternetConnection	-	Перевіряє підключення до інтернету, повертає False якщо інтернету немає та True якщо є.

Опис класів модуля CameRAdar.py наведено у таблиці 3.10.

Таблиця 3.10 – Опис класів модуля CameRAdar.py

Клас	Опис
1	2
MainWindowValueChange(QtCore.QObject)	Потрібен для створення та запуску окремого від вікна main потоку для запуску в ньому процесу виявлення вразливостей.

Продовження таблиці 3.10

1	2
ResultWindowValueChanged(QtCore.QObject)	Відстежує зміни змінних: name, result_total, result_alive, result_alive_with_img, result_img, brute_total, brute_available, brute_block, brute_combinations_left для оновлення їх у вікні Results.
MainWindowQThread(QThread, QObject)	Потрібен для створення окремого потоку, який виконує: запуск процесу виявлення вразливостей, управління процесом, оновлення інформації у вікні Main.
ResultWindowQThread(QThread, QObject)	Потрібен для створення та запуску окремого від вікна Results потоку для запуску в ньому процесу виявлення вразливостей, перевірку актуальності виявлених раніше вразливостей.
PlayVideoQThread(QThread, QObject)	Потрібен для створення окремого потоку для виведення зображення із системи відеоспостереження.
WindowManager()	Виконує завантаження вікон Main та Results.

Опис методів класу MainWindowQThread наведено у таблиці 3.11.

Таблиця 3.11 – Опис методів класу MainWindowQThread

Метод	Аргументи	Опис
1	2	3
<code>__init__</code>	<code>res, qmlObj</code>	Конструктор класу приймаючий об'єкт класу <code>MainWindowValueChanged</code> для оновлення результатів у графічному інтерфейсі та об'єкт, що представляє вікно графічного інтерфейсу для передачі в нього об'єктів та виклику функцій JavaScript.
<code>run</code>	-	Перевизначений метод успадкованого класу <code>QThread</code> , в якому виконується запуск процесу виявлення вразливостей, вилучення з нього результатів та передача їх графічному інтерфейсу.
<code>end</code>	-	Викликається методом <code>run</code> після закінчення процесу або при сигналі про завершення від інтерфейсу. Виконує зупинку процесу та повернення початкових значень змінним.
<code>getImg</code>	-	Витягує інформацію про знайдені вразливі системи відеоспостереження з процесу та надсилає їх до списку результатів графічного інтерфейсу.
<code>setValue</code>	<code>testName, rangeIp, threads, timeout</code>	Необхідний для передачі значень з графічного інтерфейсу атрибутам класу, які використовуватимуться запущеним потоком.
<code>pause</code>	-	Необхідний для передачі сигналу про призупинення з графічного інтерфейсу.

Продовження таблиці 3.11

1	2	3
cont	-	Необхідний для передачі сигналу про продовження виконання процесу з графічного інтерфейсу.
stop	-	Необхідний для передачі сигналу про зупинку з графічного інтерфейсу.
loadResultsWindow	-	Необхідний для передачі сигналу про перехід до вікна з результатами з графічного інтерфейсу.

Опис методів класу ResultWindowQThread наведено у таблиці 3.12.

Таблиця 3.12 – Опис методів класу ResultWindowQThread

Метод	Аргументи	Опис
1	2	3
__init__	res, qmlObj	Конструктор класу приймаючий об'єкт класу ResultWindowValueChange для оновлення результатів у графічному інтерфейсі та об'єкт, що представляє вікно графічного інтерфейсу для передачі в нього об'єктів та виклику функцій JavaScript.
run	-	Перевизначений метод успадкованого класу QThread, в якому виконується запуск процесу перевірки усунення вразливостей або перебору комбінацій, вилучення з нього результатів та передача їх графічному інтерфейсу.

Продовження таблиці 3.12

1	2	3
getImg	-	Витягує інформацію про знайдені вразливі системи відеоспостереження з процесу та надсилає їх до списку результатів графічного інтерфейсу.
loadImg	testName	Завантажує список вразливих систем відеоспостереження вказаного тестування до графічного інтерфейсу.
loadTests	-	Завантажує список тестувань в графічний інтерфейс.
getTestData	testName	Завантажує результати вказаного тестування у графічний інтерфейс.
runUpdateImg	testName, treads, aliveOnly	Необхідний для передачі значень з графічного інтерфейсу атрибутам класу, які використовуватимуться запущеним потоком, встановлює значення для виконання процесу перевірки усунення вразливостей.
runBrute	testName, treads, timeout, aliveOnly	Необхідний для передачі значень з графічного інтерфейсу атрибутам класу, які будуть використовуватися запущеним потоком, встановлює значення для виконання процесу перебору комбінацій.
stopBrute	-	Необхідний для передачі сигналу про зупинку процесу перебору значень з графічного інтерфейсу.
stopUpdateImg	-	Необхідний для передачі сигналу про зупинення процесу перевірки усунення вразливостей з графічного інтерфейсу.

Продовження таблиці 3.12

1	2	3
removeTestBlock	testName	Необхідний для передачі сигналу про видалення результатів зазначеного тестування з графічного інтерфейсу.
loadHomeWindow	-	Необхідний для передачі сигналу про перехід до початкового вікна з графічного інтерфейсу.

3.7 Структура бази даних

Збереження даних здійснюється за допомогою серіалізації та десеріалізації класів за допомогою бібліотеки pickle, структура даних зображена на рисунку 3.3.

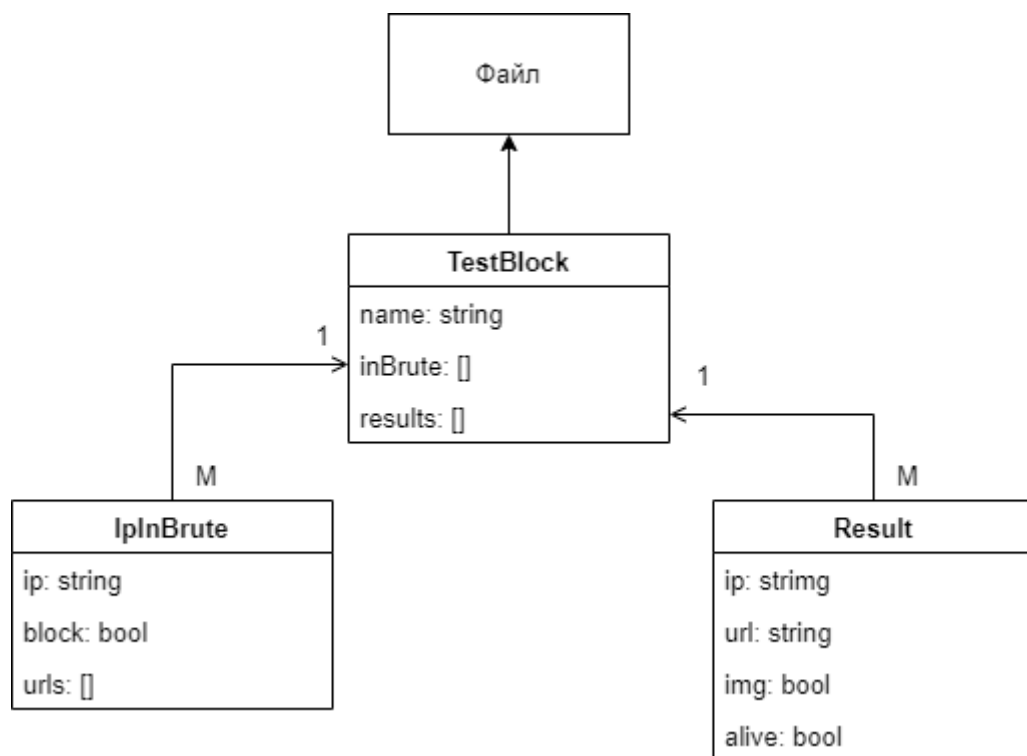


Рисунок 3.3 – Діаграма зв'язків таблиць бази даних

Структура даних складається із трьох об'єктів:

- Result - інформація про вразливу систему відео спостереження, містить таку інформацію: ір-адреса, підібрана комбінація, чи є зображення, чи можливо підключиться;
- CamInBrute - інформація про систему відеоспостереження, яка знаходиться в процесі виявлення вразливостей, містить таку інформацію: ір-адреса, список пройдених комбінацій, чи можливо підключиться;
- TestBlock - основний елемент, що містить всю інформацію з тестування, містить списки елементів Result та CamInBrute.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

ПЗ «CameRAadar» призначене для виявлення вразливостей систем відеоспостереження, якими володіє користувач або до яких дозволено доступ з боку власника.

Технічні вимоги:

- операційна система – Windows 7 і вище або Linux 2010 року та вище;
- вільне місце на жорсткому диску не менше ніж 10 мб;
- оперативна пам'ять щонайменше 1 гб;
- процесор не слабший за Intel Pentium 1500MHz;
- наявність wi-fi або ethernet.

Програмні вимоги:

- встановлений інтерпретатор Python 3.10;
- встановлені бібліотеки netaddr, opencv-python та PyQt5.

4.2 Характеристика програми

Робоче ім'я програми "CameRAadar".

Початковий обсяг файлів програми має розмір приблизно 153 кб.

Програма розроблена за допомогою мови програмування Python у середовищі розробки Visual Studio 2022, інтерфейс реалізований за допомогою бібліотеки PyQt5 та декларативної мови QML.

Основним файлом, що запускає програму, є «CameRAadar.py».

Директорії програми:

- "\results\img" - необхідна для збереження знімків із систем відеоспостереження;
- "\ranges\continue" - необхідна для збереження непройдених ір-адрес;
- "\data" - необхідна для зберігання результатів та комбінацій;

– "\\data\\backup" - необхідна зберігання резервної копії файлу з результатами.

Модулі програми:

- DataManager.py;
- ProcessManager.py;
- UpdateAndBruteProcessManager.py.

Файли, які представляють вікна графічного інтерфейсу, знаходяться в папці “view”:

- main.qml;
- Results.qml.

Модулі графічного інтерфейсу знаходяться у папці “view”:

- CustomSpinBox.qml;
- CustomCheckBox.qml;
- CustomButton.qml;
- CircularProgressBar.qml.

Файли даних, що знаходяться в папці "\\data":

- urls.txt – містить комбінації;
- data – містить результати тестувань.

4.3 Звернення до програми

Для звернення до програми необхідно встановити Python 3.10 і Visual Studio 2022 після чого необхідно перейти в папку з проектом “CameRAadar” (рис. 4.1) та запустити файл “CameRAadar.sln” який запустить проект у середовищі розробки, після чого можна редагувати програму, щоб запустити програму необхідно натиснути кнопку “Пуск”.

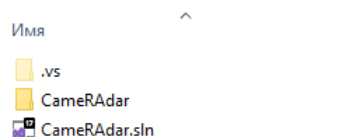


Рисунок 4.1 – Вміст папки проекту “CameRAadar”

4.4 Початкові та вихідні дані

Початковими даними є:

- ір-адреси як поодиноці так і у вигляді діапазону;
- назва тестування;
- кількість потоків;
- час очікування відповіді на запит від ір-адреси;
- комбінації для перебору, перші 4 зі списку.

Вихідними даними є:

- кількість ір-адрес з відкритим 554 портом;
- кількість вразливих систем відеоспостереження;
- кількість вразливих систем відеоспостереження, з яких вдалося отримати зображення;
- зображення з вразливих систем відеоспостереження;
- відео трансляція з камери системи відеоспостереження;
- кількість вразливих систем відеоспостереження, вразливість яких було усунуто;
- кількість знайдених систем відеоспостереження, які більше не відповідають на запити;
- кількість знайдених систем відеоспостереження, виявлення вразливостей яких можна продовжити;
- кількість комбінацій, що залишилися.

4.5 Повідомлення

Програма може виводити консольні повідомлення про непередбачені помилки. Також програма виводить консольні логи про завершення процесів, необхідні для спрощення проведення налагоджувальних робіт та додаткового відстеження коректного завершення всіх процесів.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Розроблене програмне забезпечення «CameRAдар» надає можливість користувачам проводити тестування безпеки своїх систем відеоспостереження шляхом виявлення вразливостей пов'язаних з використанням стандартних налаштувань та стандартного імені користувача та пароля. Також програма може використовуватись для проведення лабораторних експериментів для перевірки або демонстрації безпеки стандартних налаштувань різних виробників систем відеоспостереження.

Функції, які виконує програма:

- пошук систем відеоспостереження у зазначених ір-адресах з попереднім виявленням вразливостей, перебиранням перших 4 комбінації зі списку;
- вибір кількості потоків;
- вибір часу очікування відповіді від ір-адреси на запит;
- підключення до камери вразливої системи відеоспостереження для отримання зображення;
- можливість встановити процес пошуку на паузу;
- можливість зупинити процес пошуку із збереженням непройдених ір-адрес;
- перегляд та видалення результатів проведених тестувань;
- перевірка вразливих систем відеоспостереження на усунення вразливості;
- продовжити виявлення вразливостей знайдених систем відеоспостереження.

5.2 Умови виконання програми

Технічні вимоги:

– операційна система – Windows 7 і вище або Linux 2010 року та вище;

- вільне місце на жорсткому диску не менше ніж 10 мб;
- оперативна пам'ять щонайменше 1 гб;
- процесор не слабший за Intel Pentium 1500MHz;
- наявність wi-fi або ethernet.

Програмні вимоги:

- встановлений інтерпретатор Python 3.10;
- встановлені бібліотеки netaddr, opencv-python та PyQt5.

5.3 Виконання програми

Для запуску програми необхідно запустити файл CameRAadar.py інтерпретатор Python, після чого відкриється головне вікно програми (рис. 5.1).

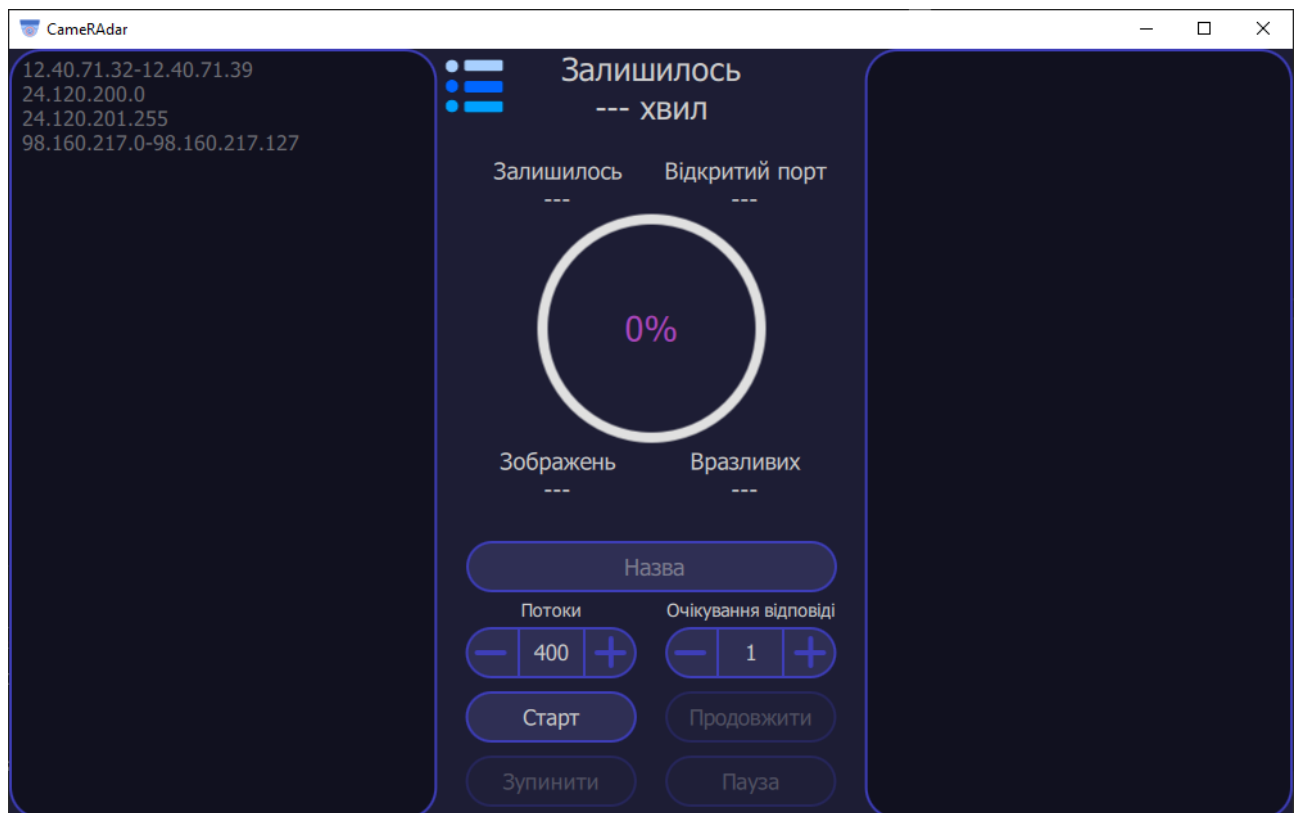


Рисунок 5.1 – Головне вікно програми

Для запуску процесу пошуку систем відеоспостереження з попереднім виявленням вразливостей, необхідно задати ір-адреси в текстове поле, що знаходиться зліва і назву тестування, після чого можна вибрати кількість потоків, час очікування відповіді на запит відкритості 554 порту у ір-адреси в секундах і натиснути кнопку “Старт”.

Після натискання кнопки “Старт” за умови правильно заданих ір-адрес, заданої назви та наявності інтернету, буде виконано запуск усіх необхідних процесів, це може зайняти кілька секунд, залежить це від кількості заданих потоків. Від обраної кількості потоків залежатиме час виконання процесу сканування ір-адрес. При активному процесі сканування будуть активні кнопки “Пауза” та “Зупинити” (рис. 5.2).

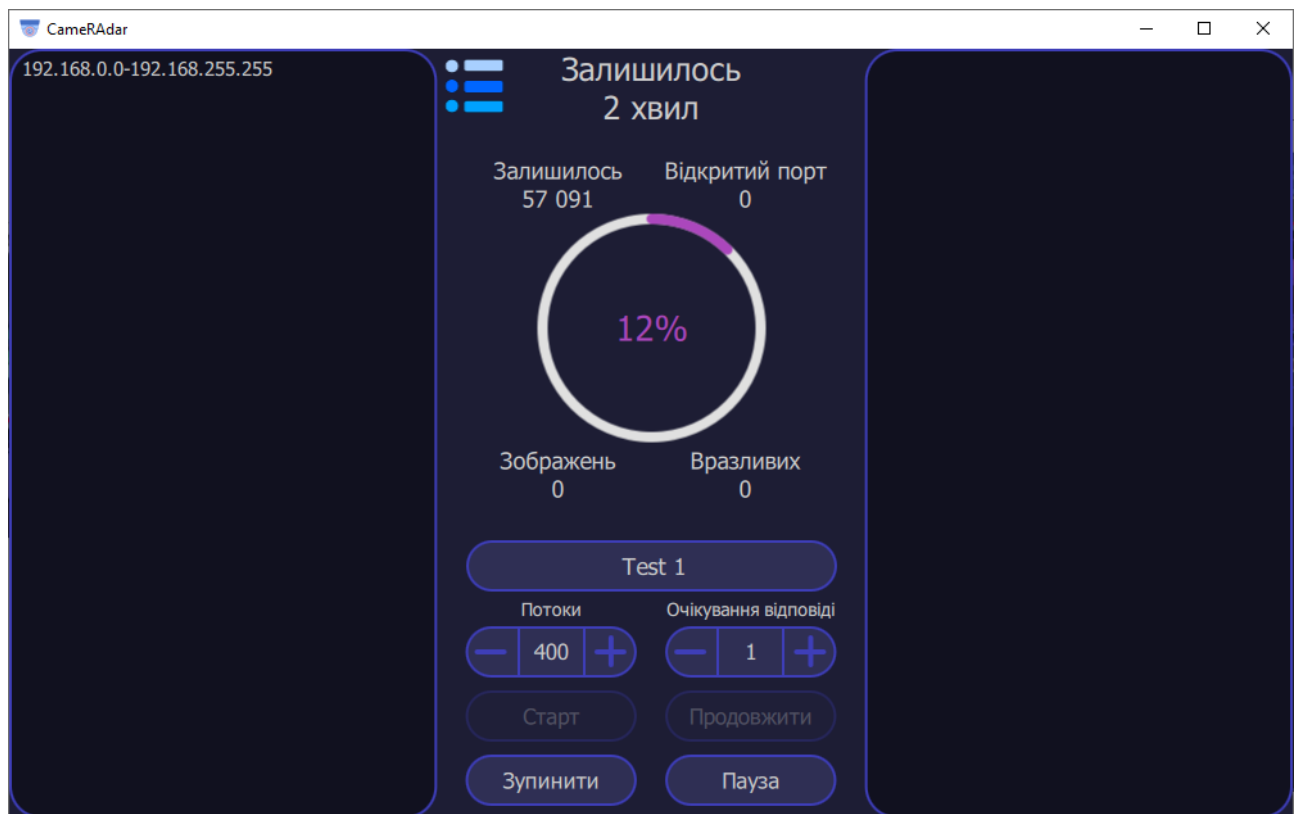


Рисунок 5.2 – Запущений процес пошуку вразливостей систем відеоспостереження

Справа знаходиться список з вразливими системами відеоспостереження, до яких можна підключитися, якщо зображення не містить

напис «No signal» (рис. 5.3).

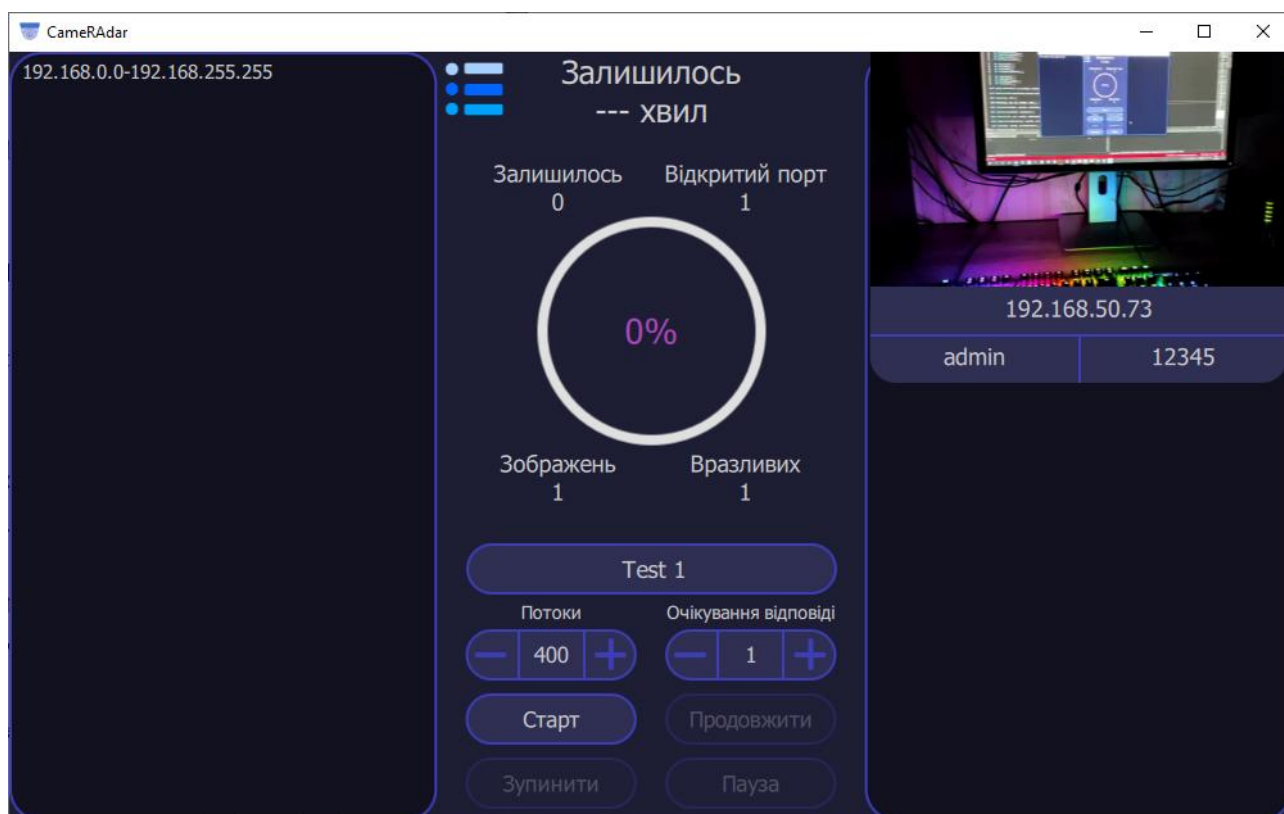


Рисунок 5.3 – Завершений процес з однією вразливою системою відеоспостереження у списку результатів

Під зображенням вказана ір-адреса системи відеоспостереження, зліва під ір-адресою вказано ім'я користувача, а праворуч пароль (рис. 5.4).

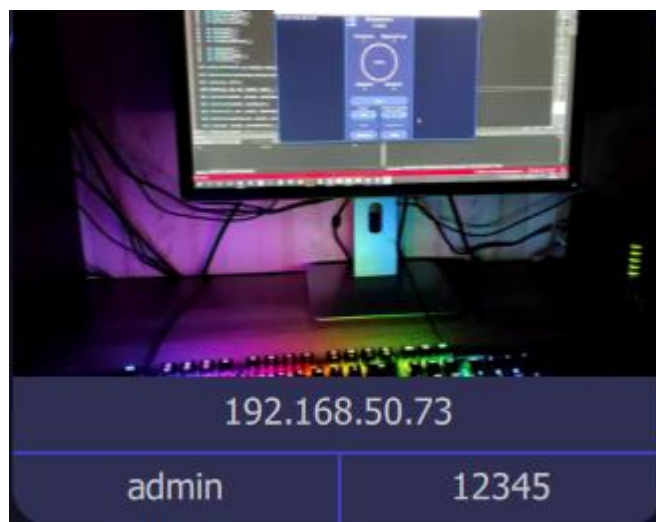


Рисунок 5.4 – Елемент зі списку вразливих систем відеоспостереження

Для підключення до вразливої системи відеоспостереження необхідно натиснути на зображення, після чого відкриється вікно з трансляцією (рис. 5.5) яке можна закрити натисканням кнопки «q» на клавіатурі.

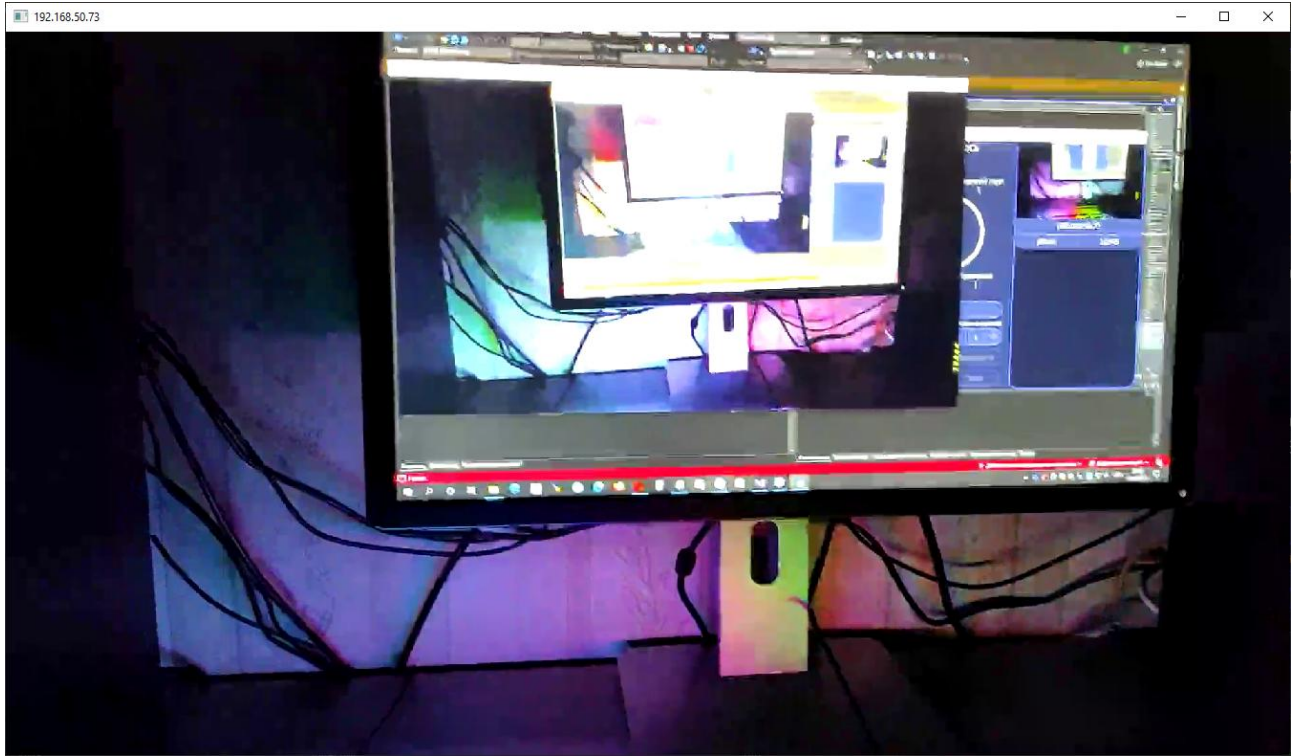


Рисунок 5.5 – Вікно з трансляцією з камери

При натисканні на кнопку "Зупинити" процес буде поставлений на паузу і будуть активні кнопки "Зупинити" та "Продовжити" (рис. 5.6), при натисканні на кнопку "Продовжити", процес буде продовжений, при натисканні на кнопку "Зупинити", процес буде зупинено а непройдені ір-адреси збережені в текстовий файл файл з назвою тестування на шляху "\ranges\continue" (рис. 5.7-5.8).

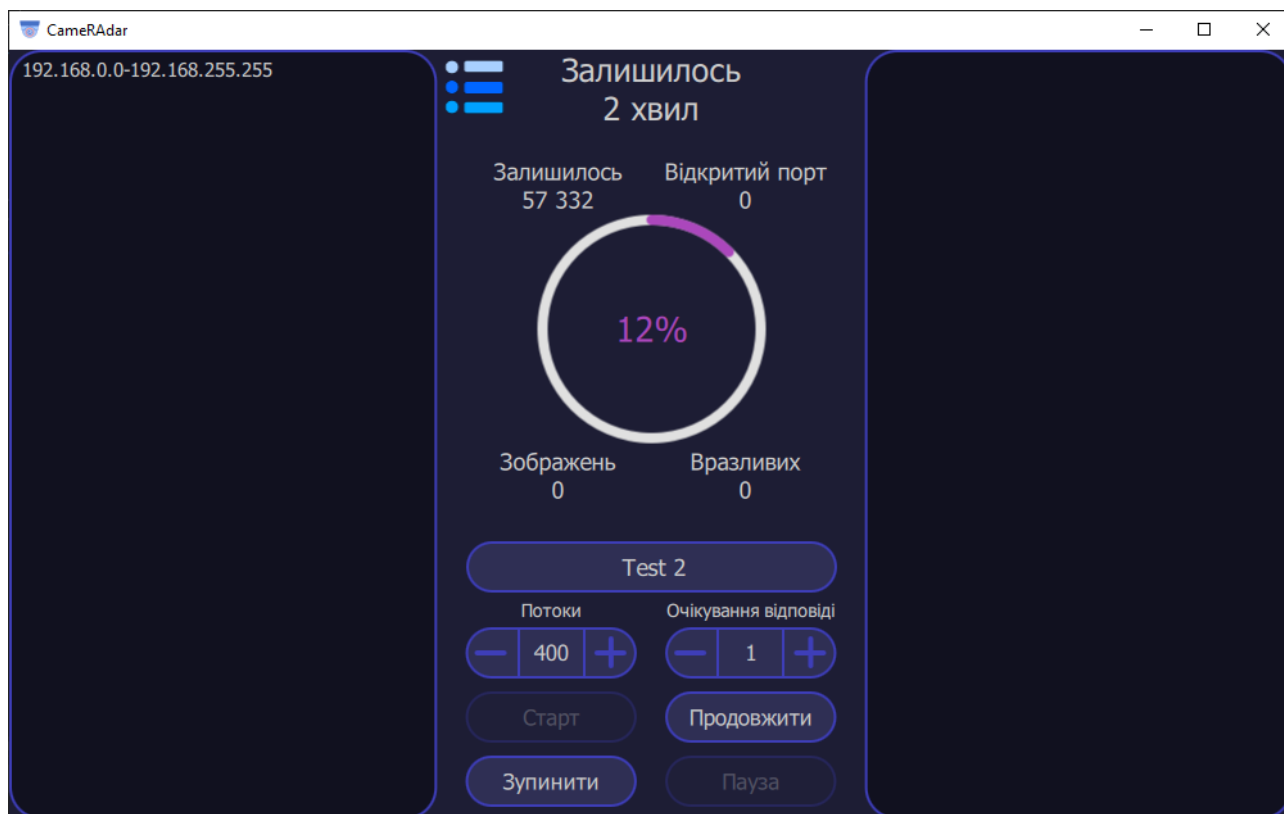


Рисунок 5.6 – Процес поставлений на паузу



Рисунок 5.7 – Файл із непройденими ір-адресами

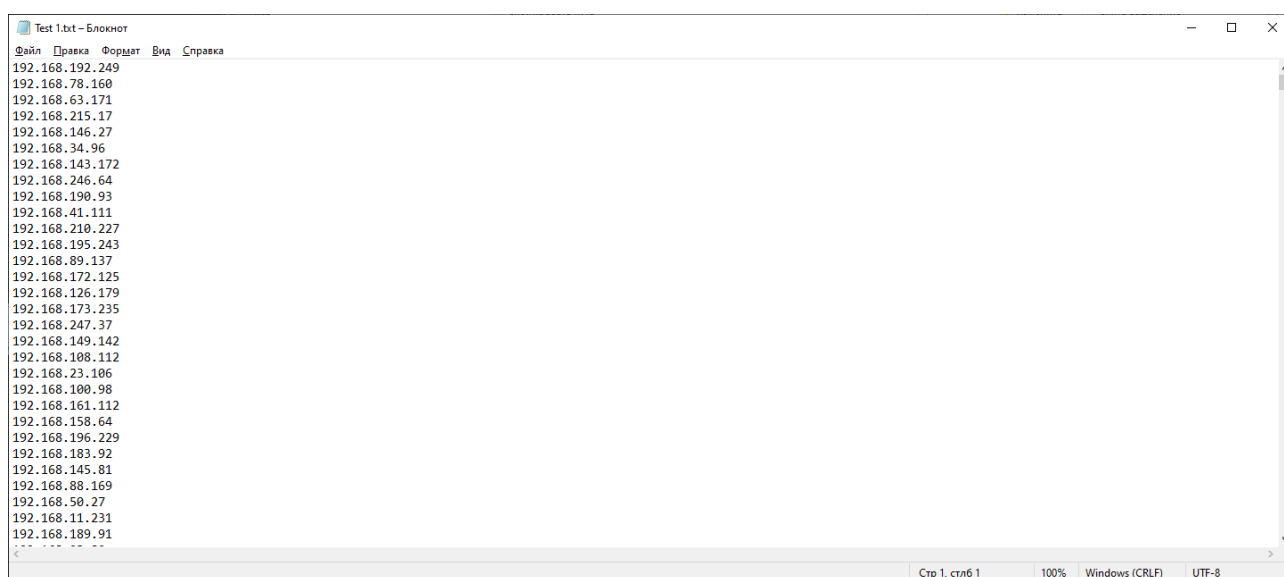


Рисунок 5.8 – Вміст файлу з непройденими ір-адресами

Після зупинки процесу прогрес у відсотках дорівнюватиме нулю, а очікуваний час “---- хв” (рис. 5.3) та оновлено файл резервної копії у папці “\\data\\backup”.

Для переходу до результатів тестувань необхідно натиснути на іконку, що зображає список (рис. 5.9), після чого відкриється вікно з результатами (рис. 5.10).



Рисунок 5.9 – Іконка для переходу до вікна результатів

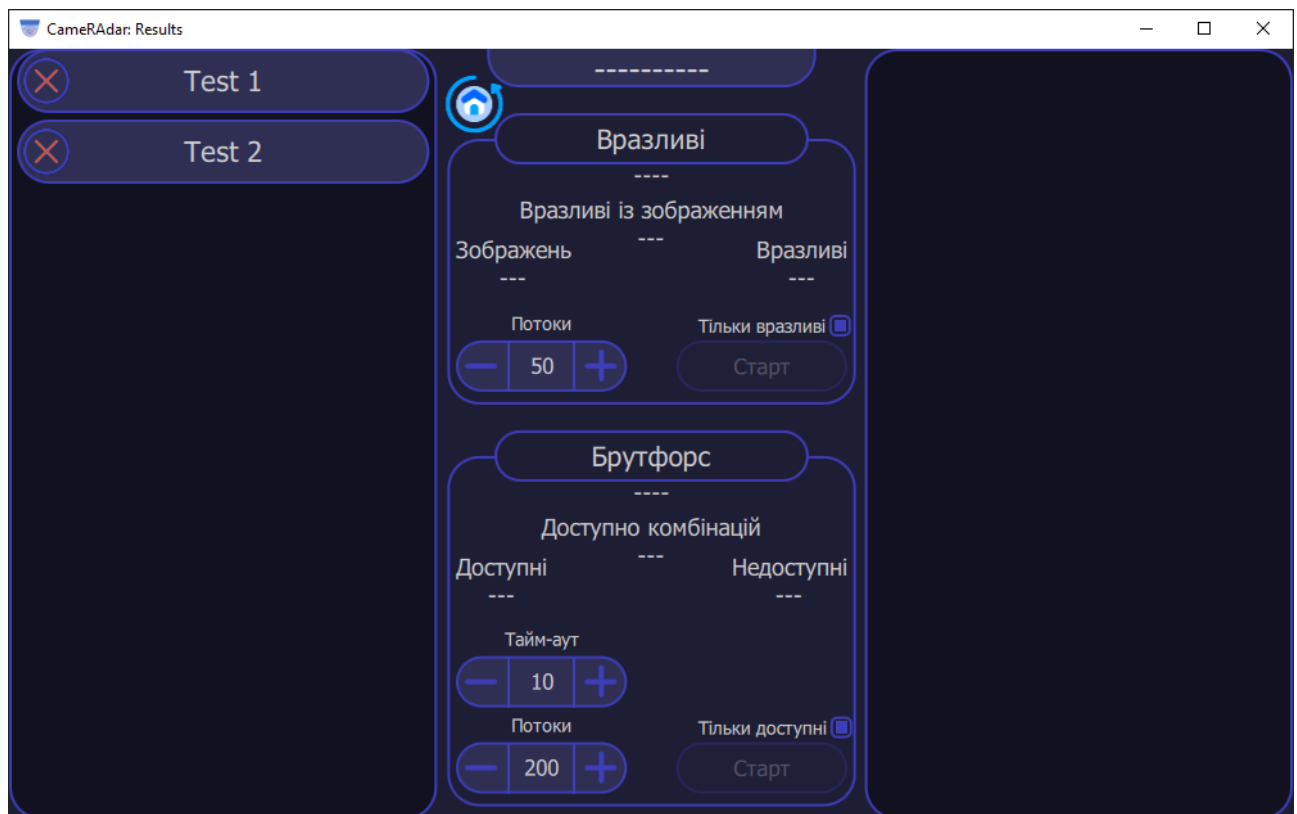


Рисунок 5.10 – Вікно з результатами

Зліва знаходиться список тестувань, що проводяться, при натисканні на елемент виводиться інформація про тестування яка розташована по центру вікна і містить у собі два розділи: "Вразливі" і "Брутфорс" (рис. 5.11).

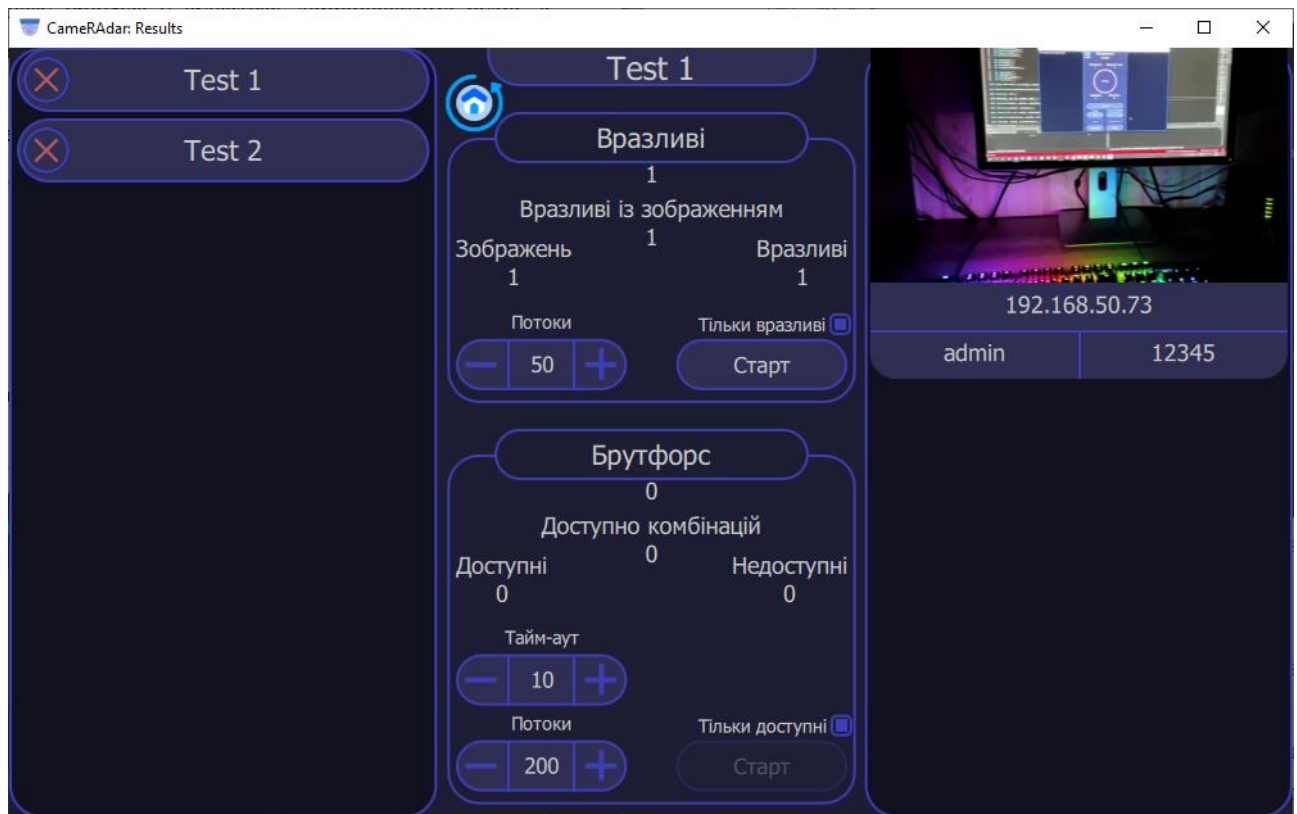


Рисунок 5.11 – Результати тестування «Test 1»

Опис розділів "Вразливі" та "Брутфорс".

"Вразливі". В цьому розділі відображається загальна кількість систем відеоспостереження, що потрапили в цю категорію, загальна кількість систем відеоспостереження з яких вдалося отримати зображення, кількість вразливих, кількість вразливих із зображенням. Також у цьому розділі можна провести перевірку на усунення вразливості, обравши кількість потоків, додатково можна перевірити невразливі, прибравши галочку з чекбоксу "Тільки вразливі" та натиснувши кнопку "Старт", після чого кнопка змінить свій стан на "Стоп". При натисканні на кнопку "Стоп" процес завершить свою роботу.

"Брутфорс". У цьому розділі відображається загальна кількість систем відеоспостереження, що потрапили в цю категорію, кількість недоступних систем відеоспостереження, кількість доступних і кількість комбінацій, що залишилися. Для запуску процесу перебору необхідно вибрати час для перерви між комбінаціями «Тайм-аут», кількість потоків, додатково можна перевірити недоступні, прибравши галочку з чекбоксу "Тільки доступні" та натиснути

кнопку "Старт", після чого кнопка змінить свій стан на "Стоп". При натисканні на кнопку "Стоп" процес завершить свою роботу.

Після завершення процесу перевірки або брутфорсу буде оновлено файл "data" резервної копії в папці «\data\backup».

Видалення результатів тестування виконується натисканням на червоний хрестик у кружечку, який знаходиться ліворуч від назви тестування.

Справа знаходиться список з вразливими системами відеоспостереження, до яких можна підключитися, якщо зображення не містить напис "No signal" (рис. 5.11).

Повернутись на головну можна клікнувши по іконці будиночка (рис. 5.12).

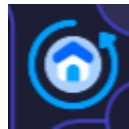


Рисунок 5.12 – Іконка для повернення до головного вікна

Редагування списку комбінацій здійснюється через редагування текстового файлу urls.txt будь-яким текстовим редактором, формат комбінації «{ім'я користувача}:{пароль}__ {URN}» (рис. 5.13).

```

urls.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
admin:12345__ /h264_pcm.sdp
admin:12345__ /11
admin:12345__ /mpeg4/ch01/main/av_stream
admin:12345__ /Streaming/Channels/101
admin:admin__ /cam/realmonitor?channel=1&subtype=0
admin:1234__ /11
root:pass__ /axis-media/media.amp
admin:123456__ /onvif-stream1
ubnt:ubnt__ /live/ch00_0

```

Рисунок 5.13 – Вміст файлу urls.txt

5.4 Повідомлення оператора

Кордони поля для введення ір-адрес стало червоним (рис. 5.14) – це означає, що користувач залишив поле порожнім або припустився помилки.

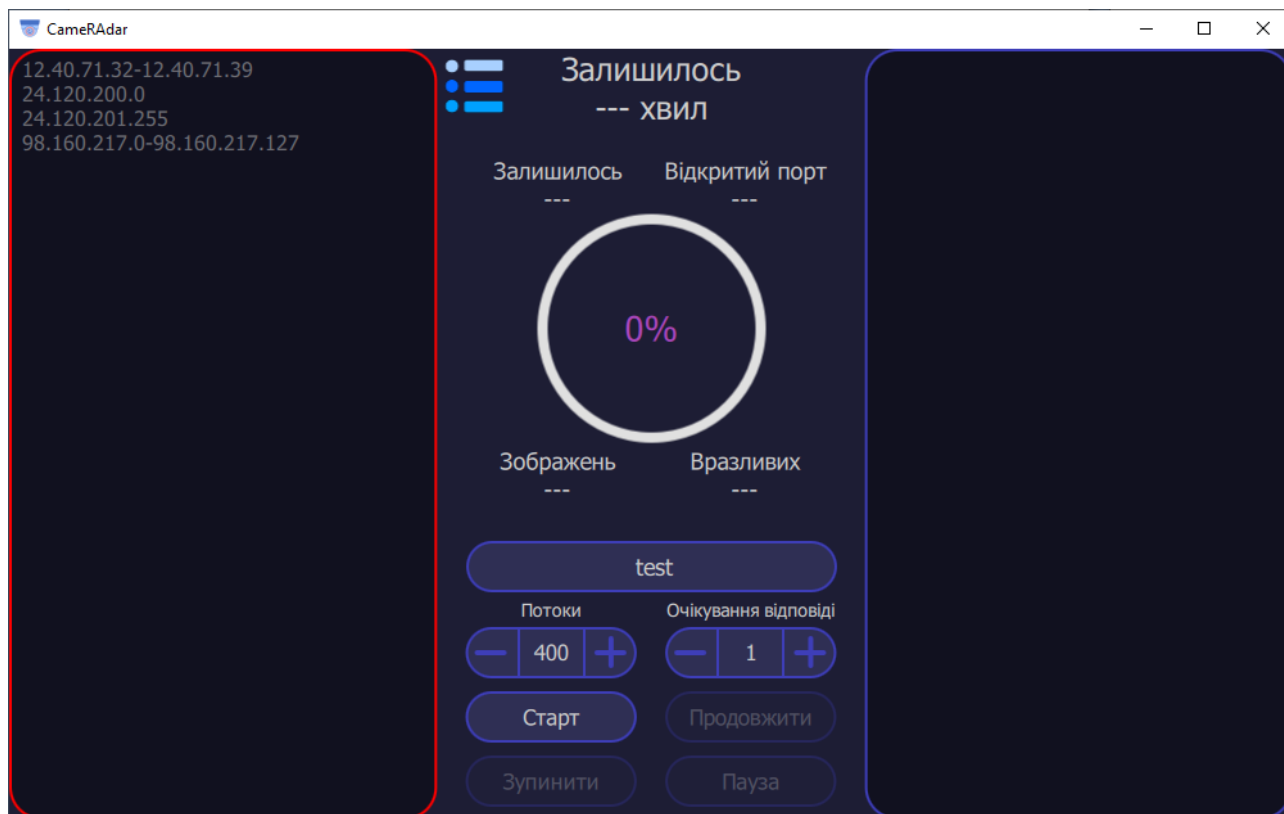


Рисунок 5.14 – Повідомлення про те що користувач залишив поле ір-адрес порожнім або припустився помилки

Кордони поля для введення назви тестування стало червоним (рис. 5.15) – це означає, що користувач залишив це поле порожнім.

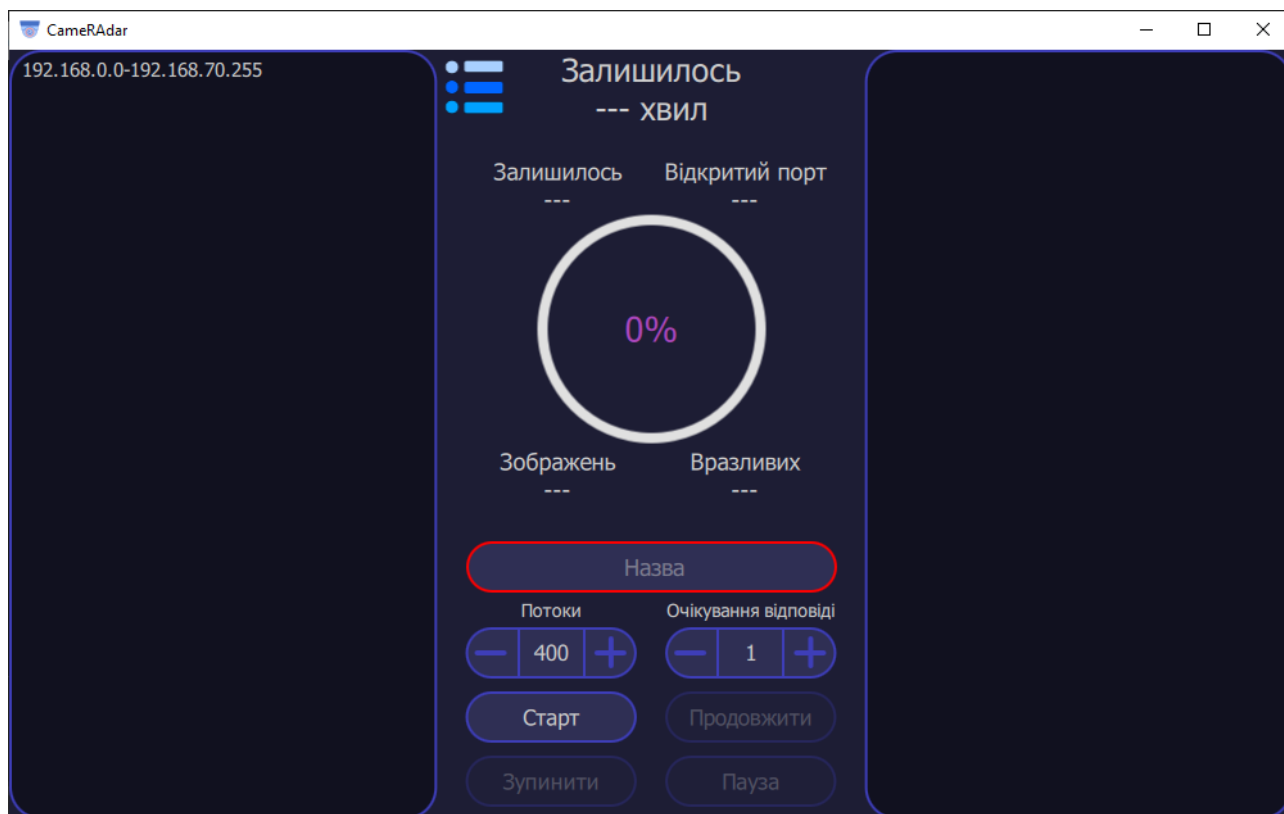


Рисунок 5.15 – Повідомлення про те що користувач залишив поле з назвою порожнім

У списку вразливих систем відеоспостереження є зображення з написом "No signal" (рис. 5.16) - це означає, що не вдалося отримати зображення з камери.

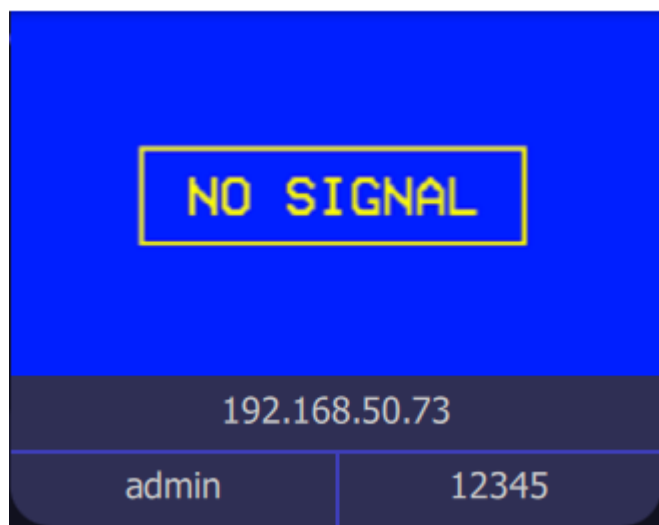


Рисунок 5.16 – Система Повідомлення про те, що з системи відеоспостереження не вдалося отримати зображення

У списку вразливих систем відеоспостереження є зображення з написом "FILE NOT FOUND" (рис. 5.17) - це означає, що в папці "results\img" відсутнє необхідне зображення.

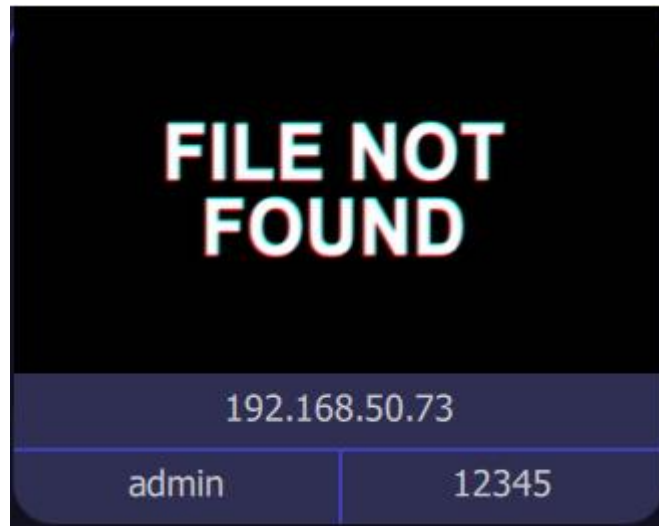


Рисунок 5.17 – Система Повідомлення про те, що в папці "results\img" відсутнє необхідне зображення

"Немає доступу до інтернету" - виникає при спробі запустити процес без доступу до інтернету.

"У файлі комбінацій є порушення формату" - виникає при порушенні формату комбінацій у файлі "urls.txt".

"Файл комбінацій порожній" - виникає якщо файл "urls.txt" порожній.

"Виявлено відсутність інтернету, процес було поставлено на паузу" - виникає при зникненні інтернет з'єднання під час виконання процесу сканування.

"Виявлено відсутність інтернету, процес був зупинений" - виникає при зникненні інтернет з'єднання під час виконання процесу перевірки усунення вразливості або брутфорсу.

"Втрачено з'єднання з камерою!" - виникає у разі втрати з'єднання з камерою під час трансляції.

"Не вдалося підключитися до камери!" - виникає при невдалій спробі підключитися до камери.

"Ви впевнені, що хочете видалити результати тестування?" - Виникає при спробі видалити результати тестування, додатково запитуючи користувача перед видаленням (рис. 5.18).

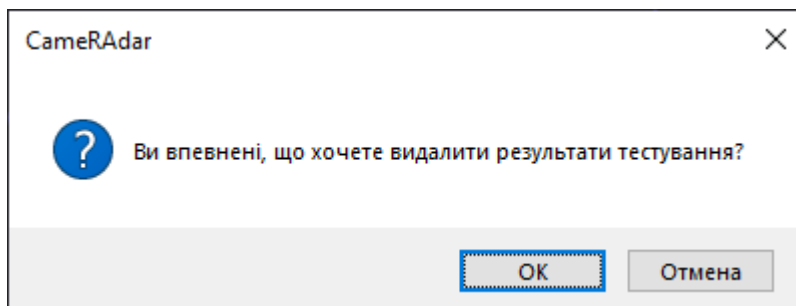


Рисунок 5.18 – Повідомлення: "Ви впевнені, що хочете видалити результати тестування?"

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено програмне забезпечення для пошуку вразливостей систем відеоспостереження, основою розробки якого став огляд актуальних вразливостей систем відеоспостереження та способи їх експлуатації.

Були розглянуті такі актуальні вразливості систем відеоспостереження:

- вразливість прошивки систем відеоспостереження Hikvision;
- вразливість стандартних налаштувань систем відеоспостереження Dahua та Hikvision;
- вразливість стандартних налаштувань систем відеоспостереження RTSP протоколу.

При огляді актуальних вразливостей, найпоширенішою виявилася вразливість пов'язана з використанням стандартних налаштувань, які встановлюються виробниками на етапі виробництва, а саме стандартне ім'я користувача, пароль та мовлення в інтернет за протоколом RTSP.

Були розглянуті стандартні імена користувачів та паролі популярних виробників таких як: ACTi, American Dynamics, Axis, Dahua, Hikvision, Panasonic, GeoVision, LTS Security, Ubiquiti, Foscam, Vivotek та Speco. Після чого було розглянуто URN для доступу до трансляції по RTSP протоколу тих самих виробників та складено рейтингову таблицю комбінацій.

При огляді протоколу RTSP було обрано метод DESCRIBE для виконання запитів підготовленими комбінаціями для пошуку вразливості.

Як середовище розробки була використана Visual Studio 2022, яка завдяки своєму широкому функціоналу та можливості встановлювати різні розширення спрощує роботу з кодом та проектом загалом.

Для створення програми була використана мова програмування Python, таке рішення було обумовлено його високою швидкістю розробки, у порівнянні з іншими мовами, переносимістю програми без необхідності компіляції та великої кількості бібліотек.

Графічний інтерфейс був створений декларативною мовою програмування QML, вибір такого рішення дозволив створити красивий і сучасний інтерфейс, деякий функціонал якого був реалізований всередині самого інтерфейсу за допомогою мови JavaScript.

Зберігання даних було реалізовано за допомогою серіалізацією та десеріалізацією класів, які виступають у ролі об'єкта в базі даних. Таке рішення було обумовлено відсутністю необхідності довготривалого зберігання та розповсюдження даних. Такий спосіб зберігання даних дозволив прискорити зчитування та запис даних.

Написано керівництво програміста та оператора, що дозволить швидко та легко ознайомитись із програмою.

Розроблене програмне забезпечення призначене для користувачів, що володіють однією або декількома системами відеоспостереження, яким необхідно перевірити їх на наявність вразливості. Також розроблене програмне забезпечення може бути використане у проведенні лабораторних експериментів, метою яких може бути перевірка або демонстрація надійності стандартних установок систем відеоспостереження різних виробників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Поняття системи відеоспостереження [Електронний ресурс]. – Access mode : http://rfcmd.ru/know_how/chto-takoe-videonabljudenuje.
2. Які уразливості є у систем відеоспостереження [Електрон. ресурс]. – Режим доступу : <https://habr.com/ru/company/trendmicro/blog/465513/>.
3. Переслідування мережі [Електрон. ресурс]. – Режим доступу : <https://ru.wikipedia.org/wiki/Нетсталкинг>.
4. Hikvision Backdoor Exploit [Electronic resource]. – Access mode : <https://ipvm.com/reports/hik-exploit>.
5. Real Time Streaming Protocol [Electronic resource]. – Access mode : https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol.
6. Чому програмісти використовують Python? [Electronic resource]. – Access mode : <http://python-3.ru/page/pochemu-programmisty-ispolzujut-python>.
7. Різниця між QtWidgets, QML, Qt Quick [Електронний ресурс]. – Режим доступу : <https://russianblogs.com/article/9267206791/>.
8. Рейтинг систем відеоспостереження [Електронний ресурс]. – Режим доступу : <https://control.ua/rejting-sistem-videonablyudeniya.html>.
9. Default Username - Password - IP Address for Security Cameras [Електронний ресурс]. – Режим доступу : <https://www.a1securitycameras.com/technical-support/default-username-passwords-ip-addresses-for-surveillance-cameras/>.
10. Camera Connection Database [Електронний ресурс]. – Режим доступу : <http://www.ispyconnect.com/cameras>.
11. Visual Studio 2022 [Електронний ресурс]. – Режим доступу : <https://habr.com/ru/company/microsoft/blog/553442/>.

ДОДАТОК А
Текст програми

```

# CameRAadar.py
db = dataManager()
db.SetUp()

def checkInternetConnection():
    return True
    try:
        request = requests.get('https://www.google.com', timeout=2)
        return True
    except (requests.ConnectionError, requests.Timeout) as exception:
        return False

def formattingNumber(numStr):
    res = ''
    ni = len(numStr) // 3
    temp = []
    for i in range(ni):
        temp.append(' ' + numStr[-3:])
        numStr = numStr[:-3]
    temp.append(' ' + numStr)
    for i in range(len(temp)):
        res += temp[-1]
        del temp[-1]
    if res[0] == ' ':
        res = res[1:]
    return res

def isCorrectIpFormat(s):
    nums = s.split(sep='.')
    if not nums[0].isdigit() or int(nums[0]) == 0:
        return False
    if len(nums) != 4:
        return False
    else:
        for n in nums:
            if not n.isdigit() or int(n) < 0 or int(n) > 255:
                return False
        return True

def separateSingleIpFromRange(l):
    singleIpL = []
    rangeIpL = []

    for i in l:
        if '-' in i:
            temp = i.split(sep='-')
            if not isCorrectIpFormat(temp[0]) or not isCorrectIpFormat(temp[1]):

```

```

        return None, None
    rangeIpL.append(i)
    continue
    if not isCorrectIpFormat(i):
        return None, None
    singleIpL.append(i)
return singleIpL, rangeIpL

def ipRangeScan(ranges):
    ips = []
    for i in ranges:
        generator = iter_iprange(i.split(sep="-")[0], i.split(sep="-")[1], step=1)
        for x in generator:
            ips.append(str(x))
    return ips

class MainWindowValueChange(QQtCore.QObject):
    ipsChanged = QtCore.pyqtSignal()
    openChanged = QtCore.pyqtSignal()
    timeChanged = QtCore.pyqtSignal()
    capturedChanged = QtCore.pyqtSignal()
    progressChanged = QtCore.pyqtSignal()
    authorizedChanged = QtCore.pyqtSignal()

    def __init__(self, parent=None):
        super(MainWindowValueChange, self).__init__(parent)
        self._ips = '---'
        self._open = '---'
        self._time = '--- хв\л'
        self._captured = '---'
        self._authorized = '---'
        self._progress = 0

    @QtCore.pyqtProperty(str, notify=ipsChanged)
    def ips(self):
        return self._ips
    @ips.setter
    def ips(self, v):
        self._ips = v
        self.ipsChanged.emit()

    @QtCore.pyqtProperty(str, notify=openChanged)
    def open(self):
        return self._open
    @open.setter
    def open(self, v):

```

```

        self._open = v
        self.openChanged.emit()

@QtCore.pyqtProperty(str, notify=timeChanged)
def time(self):
    return self._time
@time.setter
def time(self, v):
    self._time = v
    self.timeChanged.emit()

@QtCore.pyqtProperty(str, notify=capturedChanged)
def captured(self):
    return self._captured
@captured.setter
def captured(self, v):
    self._captured = v
    self.capturedChanged.emit()

@QtCore.pyqtProperty(float, notify=progressChanged)
def progress(self):
    return self._progress
@progress.setter
def progress(self, v):
    self._progress = v
    self.progressChanged.emit()

@QtCore.pyqtProperty(str, notify=authorizedChanged)
def authorized(self):
    return self._authorized
@authorized.setter
def authorized(self, v):
    self._authorized = v
    self.authorizedChanged.emit()

class ResultWindowValueChange(QtCore.QObject):
    nameChanged = QtCore.pyqtSignal()

    result_totalChanged = QtCore.pyqtSignal()
    result_aliveChanged = QtCore.pyqtSignal()
    result_alive_with_imgChanged = QtCore.pyqtSignal()
    result_imgChanged = QtCore.pyqtSignal()

    brute_totalChanged = QtCore.pyqtSignal()
    brute_availableChanged = QtCore.pyqtSignal()
    brute_blockChanged = QtCore.pyqtSignal()
    brute_combinations_leftChanged = QtCore.pyqtSignal()

```

```

def __init__(self, parent=None):
    super(ResultWindowValueChange, self).__init__(parent)
    self._name = '-----'

    self._result_total = '----'
    self._result_alive = '---'
    self._result_alive_with_img = '---'
    self._result_img = '---'

    self._brute_total = '----'
    self._brute_available = '---'
    self._brute_block = '---'
    self._brute_combinations_left = '---'

@QtCore.pyqtProperty(str, notify=nameChanged)
def name(self):
    return self._name
@name.setter
def name(self, v):
    self._name = v
    self.nameChanged.emit()

@QtCore.pyqtProperty(str, notify=result_totalChanged)
def result_total(self):
    return self._result_total
@result_total.setter
def result_total(self, v):
    self._result_total = v
    self.result_totalChanged.emit()

@QtCore.pyqtProperty(str, notify=result_aliveChanged)
def result_alive(self):
    return self._result_alive
@result_alive.setter
def result_alive(self, v):
    self._result_alive = v
    self.result_aliveChanged.emit()

@QtCore.pyqtProperty(str, notify=result_alive_with_imgChanged)
def result_alive_with_img(self):
    return self._result_alive_with_img
@result_alive_with_img.setter
def result_alive_with_img(self, v):
    self._result_alive_with_img = v
    self.result_alive_with_imgChanged.emit()

```

```

@QtCore.pyqtProperty(str, notify=result_imgChanged)
def result_img(self):
    return self._result_img
@result_img.setter
def result_img(self, v):
    self._result_img = v
    self.result_imgChanged.emit()

@QtCore.pyqtProperty(str, notify=brute_availableChanged)
def brute_available(self):
    return self._brute_available
@brute_available.setter
def brute_available(self, v):
    self._brute_available = v
    self.brute_availableChanged.emit()

@QtCore.pyqtProperty(str, notify=brute_totalChanged)
def brute_total(self):
    return self._brute_total
@brute_total.setter
def brute_total(self, v):
    self._brute_total = v
    self.brute_totalChanged.emit()

@QtCore.pyqtProperty(str, notify=brute_blockChanged)
def brute_block(self):
    return self._brute_block
@brute_block.setter
def brute_block(self, v):
    self._brute_block = v
    self.brute_blockChanged.emit()

@QtCore.pyqtProperty(str, notify=brute_combinations_leftChanged)
def brute_combinations_left(self):
    return self._brute_combinations_left
@brute_combinations_left.setter
def brute_combinations_left(self, v):
    self._brute_combinations_left = v
    self.brute_combinations_leftChanged.emit()

class MainWindowQThread(QThread, QObject):
    def __init__(self, res, listEl, parent=None):
        super().__init__()
        QObject.__init__(self)
        self.res = res
        self.listEl = listEl
        self.process = None

```

```

self.stopSig = False
self.pauseSig = False
self.testName = None
self.rangeIp = None
self.treads = None
self.timeout = None
self.valuesPg2 = None
self.resultWindowQThread = None
self.loadSecond = False

def run(self):
    db.SetUp()

    if self.testName == '':
        QMetaObject.invokeMethod(self.listEl, "nameError")
        return
    QMetaObject.invokeMethod(self.listEl, "nameNotError")

    if self.rangeIp == '':
        QMetaObject.invokeMethod(self.listEl, "rangeError")
        return

    self.rangeIp = self.rangeIp.split(sep='\n')

    while '' in self.rangeIp:
        self.rangeIp.remove('')

    temp, self.rangeIp = separateSingleIpFromRange(self.rangeIp)
    if temp == None:
        QMetaObject.invokeMethod(self.listEl, "rangeError")
        return

    QMetaObject.invokeMethod(self.listEl, "rangeNotError")

    if not checkInternetConnection():
        QMetaObject.invokeMethod(self.listEl, "end")
        messagebox.showerror("CameRAadar", 'Немає доступу до інтернету!')
        return

    urls = db.ReadUrls()
    if urls == None:
        QMetaObject.invokeMethod(self.listEl, "end")
        messagebox.showerror("CameRAadar", 'У файлі комбінацій є порушення формату!')
        return

    if len(urls) == 0:
        QMetaObject.invokeMethod(self.listEl, "end")
        messagebox.showerror("CameRAadar", 'Файл комбінацій порожній!')

```

```

        return

    ips = ipRangeScan(self.rangeIp)
    ips += temp
    ips = list(set(ips))

    self.process = processManager(self.testName, self.treads, self.timeout, ips,
    urls[:4])

    while True:
        time.sleep(0.3)
        progr = self.process.getProgress()
        self.res.progress = progr
        self.res.ips = formattingNumber(self.process.getIpLeft())
        self.res.time = self.process.getTimeLeft()
        self.res.open = formattingNumber(self.process.getOpen())
        self.res.captured = self.process.getCaptured()
        self.res.authorized = self.process.getAuthorized()

        self.getImg()
        if not self.pauseSig and not checkInternetConnection():
            QMetaObject.invokeMethod(self.listEl, "pauseF")
            messagebox.showwarning("CameRAadar", 'Виявлено відсутність інтернету,
процес було поставлено на паузу!')
            self.pauseSig = True

        if progr == 1 or self.stopSig:
            self.end()
            break

    db.backup()
    print('END RUN!')
    QMetaObject.invokeMethod(self.listEl, "end")
    self.exit()
    return

def end(self):
    self.process.stop()
    self.getImg()
    self.process.closeQueueImg()
    self.res.progress = self.process.getProgress()
    self.res.ips = formattingNumber(self.process.getIpLeft())
    self.res.time = '--- хвЛ'
    self.res.open = formattingNumber(self.process.getOpen())
    self.res.captured = self.process.getCaptured()
    self.res.authorized = self.process.getAuthorized()
    self.res.progress = 0

```

```

self.process.clear()
self.stopSig = False
self.pauseSig = False
self.process = None
def getImg(self):
    while not self.process.isImg():
        imgData = self.process.getImgData()
        if imgData[2] == 2:
            log = imgData[1].split(sep='___')[0].split(sep=':')[0]
            password = imgData[1].split(sep='___')[0].split(sep=':')[1]
            value = {"ipValue": imgData[0], "loginValue": log, "passwordValue":
password, "sourceImg": 'noSignal.png', "urlValue":imgData[1].split(sep='___')[1]}
            QMetaObject.invokeMethod(self.listEl, "append", Q_ARG(QVariant, value))
            continue
        log = imgData[1].split(sep='___')[0].split(sep=':')[0]
        password = imgData[1].split(sep='___')[0].split(sep=':')[1]
        value = {"ipValue": imgData[0], "loginValue": log, "passwordValue": password,
"sourceImg": self.getImgSource(imgData[0]), "urlValue":imgData[1].split(sep='___')[1]}
        QMetaObject.invokeMethod(self.listEl, "appendImgTop", Q_ARG(QVariant, value))
def getImgSource(self, ip):
    ip = ip.replace(".", "_")
    #if not os.path.exists('results\\img\\' + ip + '.jpg'):
    #    return 'fileNotFound.png'
    #return '..\\results\\img\\' + ip + '.jpg'    # + str(random.randrange(2, 6)) +
'.jpg'
    return '..\\results\\img\\' + str(random.randrange(2, 6)) + '.jpg'

@pyqtSlot(str, str, int, float)
def setValue(self, testName, rangeIp, treads, timeout):
    self.testName = testName
    self.rangeIp = rangeIp
    self.treads = treads
    self.timeout = timeout

@pyqtSlot()
def pause(self):
    self.process.pause()

@pyqtSlot()
def cont(self):
    if not checkInternetConnection():
        QMetaObject.invokeMethod(self.listEl, "pauseF")
        messagebox.showerror("CameRAadar", 'Немає доступу до інтернету!')
        return
    self.process.cont()
    self.pauseSig = False

@pyqtSlot()
def stop(self):

```

```

        self.stopSig = True

    @pyqtSlot()
    def loadPage(self):
        self.loadSecond = True

class ResultWindowQThread(QThread, QObject):
    def __init__(self, res, qmlObj, parent=None):
        super().__init__()
        QObject.__init__(self)
        self.res = res
        self.qmlObj = qmlObj
        self.process = None
        self.stopSig = False
        self.testName = None
        self.treads = None
        self.timeout = None
        self.bruteOrImgUpdate = True #brute
        self.aliveOnly = True
        self.loadHome = False

    def run(self):
        if self.bruteOrImgUpdate:
            if not checkInternetConnection():
                QMetaObject.invokeMethod(self.listEl, "bruteEnd")
                messagebox.showerror("CameRAadar", 'Немає доступу до інтернету!')
                return

            self.process = UpdateAndBruteProcessManager()
            self.process.bruteRun(self.testName, self.treads, self.timeout,
self.aliveOnly)
            while True:
                time.sleep(0.3)
                self.res.brute_total = self.process.getBruteTotal()
                self.res.brute_available = self.process.getBruteAvailable()
                self.res.brute_block = self.process.getBruteBlock()
                self.res.brute_combinations_left =
self.process.getBruteCombinationsLeft()
                self.getImg()
                if not self.stopSig and not checkInternetConnection():
                    QMetaObject.invokeMethod(self.listEl, "bruteSartStop")
                    messagebox.showwarning("CameRAadar", 'Виявлено відсутність інтернету,
процес був зупинений!')
                    self.stopSig = True

            if self.process.bruteIsEnd():
                QMetaObject.invokeMethod(self.qmlObj, "bruteEnd")

```

```

        self.stopSig = False
        break
    else:
        if not checkInternetConnection():
            QMetaObject.invokeMethod(self.listEl, "imgUpdateEnd")
            messagebox.showerror("CameRAadar", 'Немає доступу до інтернету!')
            self.loadImg(self.testName)
            return

        self.process = UpdateAndBruteProcessManager()
        self.process.updateImgRun(self.testName, self.treads, self.aliveOnly)
        while True:
            time.sleep(0.3)
            self.getImg()
            if not self.stopSig and not checkInternetConnection():
                QMetaObject.invokeMethod(self.listEl, "imgUpdateSartStop")
                messagebox.showwarning("CameRAadar", 'Виявлено відсутність інтернету, процес був зупинений!')
                self.loadImg(self.testName)
                self.stopSig = True

            if self.process.updateImgIsEnd():
                self.getTestData(self.testName)
                QMetaObject.invokeMethod(self.qmlObj, "imgUpdateEnd")
                self.stopSig = False
                break

        db.backup()
        print('END RUN!')
        self.exit()
        self.process = None
        return

def getImg(self):
    while not self.process.isImg():
        imgData = self.process.getImgData()
        if imgData[2] == 2:
            log = imgData[1].split(sep='___')[0].split(sep=':')[0]
            password = imgData[1].split(sep='___')[0].split(sep=':')[1]
            value = {"ipValue": imgData[0], "loginValue": log, "passwordValue":
password, "sourceImg": 'noSignal.png', "urlValue":imgData[1].split(sep='___')[1]}
            QMetaObject.invokeMethod(self.qmlObj, "appendImg", Q_ARG(QVariant,
value))

        continue
        log = imgData[1].split(sep='___')[0].split(sep=':')[0]
        password = imgData[1].split(sep='___')[0].split(sep=':')[1]

```

```

        value = {"ipValue": imgData[0], "loginValue": log, "passwordValue": password,
"sourceImg": self.getImgSource(imgData[0]), "urlValue":imgData[1].split(sep='___')[1]}
        QMetaObject.invokeMethod(self.qmlObj, "appendImgTop", Q_ARG(QVariant, value))
    def getImgSource(self, ip):
        ip = ip.replace(".", "_")
        #if not os.path.exists('results\\img\\' + ip + '.jpg'):
        #    return 'fileNotFound.png'
        #return '..\\results\\img\\' + ip + '.jpg' # + str(random.randrange(2, 6)) +
'.jpg'
        return '..\\results\\img\\' + str(random.randrange(2, 6)) + '.jpg'

    def loadImg(self, testName):
        tempIpList, tempUrlList = db.getResultIpWithImg(testName)
        QMetaObject.invokeMethod(self.qmlObj, "clearImgList")
        for i in range(len(tempIpList)):
            log = tempUrlList[i].split(sep='___')[0].split(sep=':')[0]
            password = tempUrlList[i].split(sep='___')[0].split(sep=':')[1]
            value = {"ipValue": tempIpList[i], "loginValue": log, "passwordValue":
password,
                    "sourceImg": self.getImgSource(tempIpList[i]),
"urlValue":tempUrlList[i].split(sep='___')[1]}
            QMetaObject.invokeMethod(self.qmlObj, "appendImg", Q_ARG(QVariant, value))

        tempIpList, tempUrlList = db.getResultIpWithImg(testName, True, False)
        for i in range(len(tempIpList)):
            log = tempUrlList[i].split(sep='___')[0].split(sep=':')[0]
            password = tempUrlList[i].split(sep='___')[0].split(sep=':')[1]
            value = {"ipValue": tempIpList[i], "loginValue": log, "passwordValue":
password, "sourceImg": 'noSignal.png', "urlValue":tempUrlList[i].split(sep='___')[1]}
            QMetaObject.invokeMethod(self.qmlObj, "appendImg", Q_ARG(QVariant, value))

    @pyqtSlot()
    def loadTests(self):
        temp = db.getTestNameList()
        for i in temp:
            value = {"testName": i}
            QMetaObject.invokeMethod(self.qmlObj, "appendTest", Q_ARG(QVariant, value))

    @pyqtSlot(str)
    def getTestData(self, testName):
        self.testName = testName
        self.res.name = testName

        self.res.result_total = str(db.getResultQuantity(testName))
        self.res.result_alive = str(db.getResultAlive(testName, True, True))
        self.res.result_alive_with_img = str(db.getResultIpWithImg(testName, True, True,
True))

```

```

        self.res.result_img = str(db.getResultIpWithImg(testName, False, True, True,
True))

        self.res.brute_total = str(db.getBruteQuantity(testName))
        self.res.brute_available = str(db.getBruteBlockQuantity(testName, False))
        self.res.brute_block = str(db.getBruteBlockQuantity(testName))
        self.res.brute_combinations_left = str(len(db.getNoPassedUrlList(testName,
db.ReadUrls()))))

        self.loadImg(testName)

    @pyqtSlot(str, int, bool)
    def runUpdateImg(self, testName, treads, aliveOnly):
        self.testName = testName
        self.treads = treads
        self.aliveOnly = aliveOnly
        self.bruteOrImgUpdate = False

    @pyqtSlot(str, int, int, bool)
    def runBrute(self, testName, treads, timeout, aliveOnly):
        self.testName = testName
        self.treads = treads
        self.timeout = timeout
        self.aliveOnly = aliveOnly
        self.bruteOrImgUpdate = True

    @pyqtSlot()
    def stopBrute(self):
        self.process.bruteStop()

    @pyqtSlot()
    def stopUpdateImg(self):
        self.process.updateImgStop()

    @pyqtSlot()
    def laodPage(self):
        self.loadHome = True

    @pyqtSlot(str)
    def removeTestBlock(self, testName):
        answer = messagebox.askokcancel("CameRAadar", "Ви впевнені, що хочете видалити
результати тестування?")
        if not answer:
            return
        db.removeTestBlock(testName)
        QMetaObject.invokeMethod(self.qmlObj, "clearTestListModel")
        self.loadTests()

        if self.testName == testName:
            self.testName = '-----'

```

```

        self.res.name = '-----'

        self.res.result_total = '----'
        self.res.result_alive = '---'
        self.res.result_alive_with_img = '---'
        self.res.result_img = '---'

        self.res.brute_total = '----'
        self.res.brute_available = '---'
        self.res.brute_block = '---'
        self.res.brute_combinations_left = '---'
        QMetaObject.invokeMethod(self.qmlObj, "buttonsOff")
        QMetaObject.invokeMethod(self.qmlObj, "clearImgList")

class PlayVideoQThread(QThread, QObject):
    def __init__(self):
        super().__init__()
        QObject.__init__(self)
        self.ip = ''
        self.log = ''
        self.password = ''
        self.url = ''

    def run(self):
        import cv2
        ones = True
        width = None
        height = None
        cap = cv2.VideoCapture('rtsp://' + self.log + ':' + self.password + '@' + self.ip
+ ':8080' + self.url)
        cv2.namedWindow(self.ip, cv2.WINDOW_NORMAL)
        ones = True
        scale_percent = 70
        while(cap.isOpened()):
            ret, frame = cap.read()
            if not ret:
                cap.release()
                messagebox.showwarning("CameRAдар", 'Втрачено з\'єднання з камерою!')
                cv2.destroyAllWindows()
                return
            if ones:
                width = int(frame.shape[1] * scale_percent / 100)
                height = int(frame.shape[0] * scale_percent / 100)
                cv2.resizeWindow(self.ip, (width, height))
                ones = False
            cv2.imshow(self.ip, frame)
            if cv2.waitKey(20) & 0xFF == ord('q'):

```

```

        cap.release()
        cv2.destroyAllWindows()
        return
cap.release()
cv2.destroyAllWindows()
messagebox.showwarning("CameRAdar", 'Не вдалося підключитися до камери!')

@pyqtSlot(str, str, str, str)
def setValue(self, ip, log, password, url):
    self.ip = ip
    self.log = log
    self.password = password
    self.url = url

class WindowManager():
    loadSecondPage = False
    def __init__(self):
        pass
    def loadFirstPage(self):
        app = QtGuiApplication(sys.argv)
        engine = QQmlApplicationEngine()
        values = MainWindowValueChange()

        playVideoQThread = PlayVideoQThread()

        engine.rootContext().setContextProperty("values", values)
        engine.rootContext().setContextProperty("playVideoQThread", playVideoQThread)
        app.setWindowIcon(QIcon("view\\icon.png"));
        engine.load("view\\main.qml")

        testQThread = MainWindowQThread(values, engine.rootObjects()[0])
        engine.rootContext().setContextProperty("testQThread", testQThread)
        engine.quit.connect(app.quit)
        app.exec_()

        if testQThread.loadSecond:
            self.loadSecondPage()

    def loadSecondPage(self):
        app = QtGuiApplication(sys.argv)
        engine = QQmlApplicationEngine()

        valuesPg2 = ResultWindowValueChange()
        playVideoQThread = PlayVideoQThread()
        engine.rootContext().setContextProperty("playVideoQThread", playVideoQThread)

        engine.rootContext().setContextProperty("valuesPg2", valuesPg2)

```

```

        engine.load('view\\Results.qml')
        resultWindowQThread = ResultWindowQThread(valuesPg2, engine.rootObjects()[0])
        engine.rootContext().setContextProperty("resultWindowQThread",
resultWindowQThread)
        resultWindowQThread.loadTests()
        engine.quit.connect(app.quit)
        app.exec_()

        if resultWindowQThread.loadHome:
            self.loadFirstPage()

windowManager = WindowManager()
sys.exit(windowManager.loadFirstPage())

# ProcessManager.py

class processManager:
    stopPauseSignal = None
    endOffAll = None

    counterPassedIps = []
    counterOpen = None
    counterAuthorized = None
    counterCaptured = None
    countTime = []
    resT = None
    timeProc = None
    sendIpProc = None

    ProcessList = []
    resultsCatcher = None
    queueRes = None
    queueResImg = None
    queueStop = None
    queueIp = None
    stopCatch = None

    ipsNum = 0
    def __init__(self, testName, nProcess, timeout, ips, urls):
        self.ipsNum = len(ips)
        self.stopPauseSignal = multiprocessing.Value('i', 0)# [-1] - Stop | [0] - Go |
[1] - Pause
        self.counterOpen = multiprocessing.Value('i', 0)
        self.counterAuthorized = multiprocessing.Value('i', 0)
        self.counterCaptured = multiprocessing.Value('i', 0)
        self.resT = multiprocessing.Value('i', 0)

```

```

self.endOffAll = multiprocessing.Value('i', 0)

self.queueRes = Queue()
self.queueResImg = Queue()
self.queueStop = Queue()
self.queueIp = Queue()

self.resultsCatcher = Process(target = resCatcher, args = (testName,
self.queueRes, self.queueResImg, self.counterOpen, self.counterAuthorized,
self.counterCaptured,))
self.resultsCatcher.start()
self.stopCatch = Process(target = stopCatcher, args = (testName,
self.queueStop,))
self.stopCatch.start()

idProc = 0
for i in range(nProcess):
    self.counterPassedIps.append(multiprocessing.Value('i', 0))
    self.countTime.append(multiprocessing.Value('i', 0))
    self.ProcessList.append(Process(target = scan, args = (urls, self.queueRes,
self.queueStop, self.queueIp, self.counterPassedIps[-1], self.stopPauseSignal,
self.countTime[-1], timeout,)))
    self.ProcessList[-1].start()
    idProc += 1

self.sendIpProc = Process(target = sendIp, args = (ips, self.queueIp, nProcess,))
self.sendIpProc.start()

self.timeProc = Process(target = timeLeftCal, args = (self.countTime, self.resT,
self.counterPassedIps, self.stopPauseSignal, len(ips),))
self.timeProc.start()

print('Num processes: ' + str(idProc))

def getTimeLeft(self):
    return str(self.resT.value) + ' хвил'
def getOpen(self):
    return str(self.counterOpen.value)
def getCaptured(self):
    return str(self.counterCaptured.value)
def getAuthorized(self):
    return str(self.counterAuthorized.value)
def getIpLeft(self):
    passed = 0
    for i in self.counterPassedIps:
        passed += i.value
    return str(self.ipsNum - passed)
def getProgress(self):

```

```

    passed = 0
    for i in self.counterPassedIps:
        passed += i.value
    return passed / self.ipsNum

def pause(self):
    self.stopPauseSignal.value = 1
def cont(self):
    self.stopPauseSignal.value = 0
def stop(self):
    self.stopPauseSignal.value = -1
    allIsStopped = False
    while not allIsStopped:
        allIsStopped = True
        time.sleep(1)
        for i in self.ProcessList:
            if i.is_alive():
                allIsStopped = False
                break
        if allIsStopped:
            break

    self.queueStop.put('STOP')
    self.queueRes.put([None, None, -1])
    r1 = self.queueRes.empty()
    r2 = self.queueStop.empty()
    while not r1 or not r2:
        time.sleep(1)
        r1 = self.queueRes.empty()
        r2 = self.queueStop.empty()

    self.queueRes.close()
    self.queueStop.close()
    self.queueRes.join_thread()
    self.queueStop.join_thread()
    self.resultsCatcher.join()
    self.timeProc.join()
    self.stopCatch.join()

    print('END!')
def closeQueueImg(self):
    self.queueResImg.close()
    self.queueResImg.join_thread()

def isEnd(self):
    if self.endOffAll.value == 1:
        self.timeProc.kill()

```

```

        return True
    else:
        return False
def isImg(self):
    r = self.queueResImg.empty()
    return r
def getImgData(self):
    r = self.queueResImg.get()
    return r
def clear(self):
    for i in range(len(self.counterPassedIps)):
        self.counterPassedIps[i].value = 0
        self.countTime[i].value = 0

    self.ipsNum = 0
    self.stopPauseSignal.value = 0# [-1] - Stop | [0] - Go | [1] - Pause
    self.counterOpen.value = 0
    self.counterAuthorized.value = 0
    self.counterCaptured.value = 0
    self.resT.value = 0
    self.endOffAll.value = 0

    self.stopPauseSignal = None
    self.endOffAll = None

    self.counterPassedIps = []
    self.counterOpen = None
    self.counterAuthorized = None
    self.counterCaptured = None
    self.countTime = []
    self.resT = None
    self.timeProc = None

    self.ProcessList = []
    self.resultsCatcher = None
    self.queueRes = None
    self.queueStop = None
    self.stopCatch = None
    self.queueResImg = None
def resCatcher(testName, queueRes, queueResImg, cOpen, cAuthorized, cCaptured):
    from DataManager import dataManager
    db = dataManager()
    db.addTestBlock(testName)
    while True:
        res = queueRes.get()
        if res[2] == 0:

```

```

        db.addToBrute(testName, res[0], res[1], True)
    elif res[2] == 1:
        db.addToBrute(testName, res[0], res[1])
    elif res[2] == 2:
        db.setResult(testName, res[0], res[1], False)
        cAuthorized.value += 1
        queueResImg.put(res)
    elif res[2] == 3:
        cAuthorized.value += 1
        cCaptured.value += 1
        db.setResult(testName, res[0], res[1])
        queueResImg.put(res)
    else:
        print('Res catcher and')
        return
    cOpen.value += 1

def stopCatcher(testName, queueStop):
    f = open('ranges\\continue\\' + testName + '.txt', 'w')
    rm = True
    while True:
        res = queueStop.get()
        if res != 'STOP':
            f.write(res + '\n')
            rm = False
        else:
            f.close()
            print('Stop catcher end')
            if rm:
                import os
                os.remove('ranges\\continue\\' + testName + '.txt')
            return

def timeLeftCal(t, resT, countIp, stopPauseSignal, totalIp):
    startTime = None
    while True:
        time.sleep(0.2)
        if stopPauseSignal.value == 0:
            passedForTime = 0
            if startTime == None:
                startTime = time.time()
            for i in t:
                passedForTime += i.value
            if passedForTime >= 1000:
                for i in t:
                    i.value = 0
                passedIp = 0

```

```

        for i in countIp:
            passedIp += i.value
            ti = time.time() - startTime
            ti = ((totalIp - passedIp) / passedForTime) * ti
            resT.value = int(ti / 60)
            startTime = time.time()
    elif stopPauseSignal.value == 1:
        while True:
            time.sleep(2)
            if stopPauseSignal.value == -1:
                print('Time process end')
                return
            elif stopPauseSignal.value != 1:
                break
    else:
        print('Time process end')
        return

def sendIp(ips, queueIp, processQuantity):
    for i in ips:
        queueIp.put(i)
    for i in range(processQuantity):
        queueIp.put(None)

def scan(urls, queueRes, queueStop, queueIp, counterPassedIps, stopPauseSignal,
t, timeout):
    while True:
        i = queueIp.get()
        if i == None:
            return
        if stopPauseSignal.value == 1:
            while True:
                time.sleep(2)
                if stopPauseSignal.value != 1:
                    break
            time.sleep(1)
        if stopPauseSignal.value == 0:
            countUrls = 0
            res = connect(i, urls[0].split(sep='___')[0],
urls[0].split(sep='___')[1], timeout, 0)
            if res == -1:
                counterPassedIps.value += 1
                t.value += 1
                continue
            if res == 0:
                queueRes.put([i, [], 0])
                counterPassedIps.value += 1

```

```

        t.value += 1
        continue
    if res == 2 or res == 3:
        queueRes.put([i, urls[0], res])
        counterPassedIps.value += 1
        t.value += 1
        continue
    countUrls += 1
    for u in urls[1:]:
        time.sleep(10)
        res = connect(i, u.split(sep='___')[0], u.split(sep='___')[1],
timeout, 1)

        if res == 2 or res == 3:
            queueRes.put([i, u, res])
            counterPassedIps.value += 1
            t.value += 1
            break
        elif res == 0:
            queueRes.put([i, urls[:countUrls], 0])
            counterPassedIps.value += 1
            t.value += 1
            break
        countUrls += 1
    if countUrls == len(urls):
        queueRes.put([i, urls, 1])
        counterPassedIps.value += 1
        t.value += 1

    if stopPauseSignal.value == -1:
        queueStop.put(i)

# UpdateAndBruteProcessManager.py
class UpdateAndBruteProcessManager:
    queueImg = Queue()
    queueResImgUpdate = Queue()
    queueResBrute = Queue()

    queueImgUpdate = Queue()
    queueBrute = Queue()

    stopSignal = multiprocessing.Value('i', 0)
    endSignal = multiprocessing.Value('i', 0)

    imgUpdateProcessList = []
    imgUpdatePutGetProcess = None
    resultImgUpdateProgress = multiprocessing.Value('i', 0)
    resultImgUpdateTotal = 0

```

```

bruteProcessList = []
brutePutGetProcess = None
brute_available = multiprocessing.Value('i', 0)
brute_block = multiprocessing.Value('i', 0)
brute_combinations_left = multiprocessing.Value('i', 0)
brute_total = multiprocessing.Value('i', 0)

def __init__(self):
    pass

def updateImgRun(self, testName, numProcess, aliveOnly = True):
    from DataManager import dataManager
    db = dataManager()
    self.resultImgUpdateTotal = db.getResultAlive(testName, True, True)
    if not aliveOnly:
        self.resultImgUpdateTotal += db.getResultAlive(testName, False, True)
    for i in range(numProcess):
        self.imgUpdateProcessList.append(Process(target = processImgUpdate, args =
(self.queueImgUpdate, self.queueResImgUpdate, self.stopSignal,)))
        self.imgUpdateProcessList[-1].start()
        self.imgUpdatePutGetProcess = Process(target = processPutGetImgUpdate, args =
(testName, numProcess, self.queueImg, self.queueImgUpdate, self.queueResImgUpdate,
aliveOnly, self.resultImgUpdateProgress, self.endSignal,))
        self.imgUpdatePutGetProcess.start()

def bruteRun(self, testName, numProcess, timeout, availableOnly = True):
    from DataManager import dataManager
    db = dataManager()

    self.brute_available.value = db.getBruteBlockQuantity(testName, False)
    self.brute_block.value = db.getBruteBlockQuantity(testName)
    self.brute_combinations_left.value = len(db.getNoPassedUrlList(testName,
db.ReadUrls()))
    self.brute_total.value = db.getBruteQuantity(testName)

    for i in range(numProcess):
        self.bruteProcessList.append(Process(target = processBrute, args =
(self.queueBrute, self.queueResBrute,)))
        self.bruteProcessList[-1].start()
        self.brutePutGetProcess = Process(target = processPutGetBrute, args = (testName,
numProcess, timeout, self.queueImg, self.queueBrute, self.queueResBrute, availableOnly,
self.brute_available, self.brute_block, self.brute_combinations_left, self.brute_total,
self.stopSignal, self.endSignal,))
        self.brutePutGetProcess.start()

```

```

def updateImgStop(self):
    self.stopSignal.value = 1
def updateImgIsEnd(self):
    if self.endSignal.value == 1:
        time.sleep(1)
        allIsStopped = False
        while not allIsStopped:
            allIsStopped = True
            time.sleep(1)
            for i in self.imgUpdateProcessList:
                if i.is_alive():
                    allIsStopped = False
                    break
            if allIsStopped:
                break
        self.queueImg = None
        self.queueResImgUpdate = None
        self.queueImgUpdate = None
        for i in range(len(self.imgUpdateProcessList)):
            self.imgUpdateProcessList[i] = None
        self.imgUpdateProcessList.clear()
        self.imgUpdatePutGetProcess = None
        self.endSignal.value = 0
        self.stopSignal.value = 0
        return True
    else:
        False
def getProgressUpdateImg(self):
    return self.resultImgUpdateProgress / self.resultImgUpdateTotal

def bruteStop(self):
    self.stopSignal.value = 1
def bruteIsEnd(self):
    if self.endSignal.value == 1:
        time.sleep(1)
        allIsStopped = False
        while not allIsStopped:
            allIsStopped = True
            time.sleep(1)
            for i in self.bruteProcessList:
                if i.is_alive():
                    allIsStopped = False
                    break
            if allIsStopped:
                break
        self.queueImg = None
        self.queueResBrute = None

```

```

        self.queueBrute = None
        for i in range(len(self.bruteProcessList)):
            self.bruteProcessList[i] = None
        self.bruteProcessList.clear()
        self.brutePutGetProcess = None
        self.endSignal.value = 0
        self.stopSignal.value = 0
        return True
    else:
        False

def getBruteAvailable(self):
    return str(self.brute_available.value)
def getBruteBlock(self):
    return str(self.brute_block.value)
def getBruteCombinationsLeft(self):
    return str(self.brute_combinations_left.value)
def getBruteTotal(self):
    return str(self.brute_total.value)

def isImg(self):
    r = self.queueImg.empty()
    return r
def getImgData(self):
    r = self.queueImg.get()
    return r

def processPutGetImgUpdate(testName, numProcess, queueImg, queueImgUpdate,
queueResImgUpdate, aliveOnly, resultImgUpdateProgress, endSignal):
    from DataManager import dataManager
    db = dataManager()
    ipList, urlList = db.getResultAlive(testName)
    if not aliveOnly:
        bufIp, bufUrl = db.getResultAlive(testName, False)
        ipList += bufIp
        urlList += bufUrl
    for i in range(len(ipList)):
        queueImgUpdate.put([ipList[i], urlList[i]])
    for i in range(numProcess):
        queueImgUpdate.put(None)
    stopProcesseCounter = 0
    while True:
        res = queueResImgUpdate.get()
        if res == None:
            stopProcesseCounter += 1
            if stopProcesseCounter == numProcess:
                endSignal.value = 1

```

```

        break
    elif res[2] == -1:
        db.setNotAlive(testName, res[0])
        resultImgUpdateProgress.value += 1
    elif res[2] == 2:
        resultImgUpdateProgress.value += 1
        db.setAlive(testName, res[0])
        queueImg.put(res)
    elif res[2] == 3:
        queueImg.put(res)
        db.setYseImg(testName, res[0])
        db.setAlive(testName, res[0])
        resultImgUpdateProgress.value += 1
def processImgUpdate(queueImgUpdate, queueResImgUpdate, stopSignal):
    while True:
        val = queueImgUpdate.get()
        if stopSignal.value == 1 or val == None:
            queueResImgUpdate.put(None)
            return
        url = 'rtsp://' + val[1].split(sep='___')[0]
        url += '@' + val[0] + ':554' + val[1].split(sep='___')[1]
        res = capture(val[0], url)
        queueResImgUpdate.put([val[0], val[1], res])

def processPutGetBrute(testName, numProcess, timeout, queueImg, queueBrute,
queueResBrute, availableOnly, brute_available, brute_block, brute_combinations_left,
brute_total, stopSignal, endSignal):
    from DataManager import dataManager
    db = dataManager()
    if not availableOnly:
        ips, urls = db.getBruteBlockForCheck(testName)
        for i in range(len(ips)):
            queueBrute.put([ips[i], urls[i]])
        for i in range(len(ips)):
            res = queueResBrute.get()
            if res[2] == 1:
                db.addPassedUrl(testName, res[0], res[1])
                db.setBlock(testName, res[0], False)
                brute_available.value += 1
                brute_block.value -= 1
            elif res[2] == 2:
                db.addResult(testName, res[0], res[1], False)
                db.setBlock(testName, res[0], False)
                brute_block.value -= 1
                brute_total.value -= 1
                queueImg.put(res)
            elif res[2] == 3:

```

```

        db.setResult(testName, res[0], res[1])
        db.setBlock(testName, res[0], False)
        queueImg.put(res)
        brute_block.value -= 1
        brute_total.value -= 1
    time.sleep(timeout)

urlList = db.ReadUrls()
combinations = db.getNoPassedUrlList(testName, urlList)
brute_combinations_left.value = len(combinations)

for x in combinations:
    if stopSignal.value == 1:
        for i in range(numProcess):
            queueBrute.put(None)
        endSignal.value = 1
        return
    noPassedUrlList = db.getBruteWithNoPassedUrl(testName, x)
    if len(noPassedUrlList) == 0:
        brute_combinations_left.value -= 1
        continue

    for i in noPassedUrlList:
        queueBrute.put([i, x])
    for i in range(len(noPassedUrlList)):
        res = queueResBrute.get()
        #print(res[2])
        if res[2] == 0:
            db.setBlock(testName, res[0])
            brute_available.value -= 1
            brute_block.value += 1
        elif res[2] == 1:
            db.addPassedUrl(testName, res[0], res[1])
        elif res[2] == 2:
            db.setResult(testName, res[0], res[1], False)
            brute_available.value -= 1
            brute_total.value -= 1
            queueImg.put(res)
        elif res[2] == 3:
            db.setResult(testName, res[0], res[1])
            #queueImg.put([res[0], res[1]])
            queueImg.put(res)
            brute_available.value -= 1
            brute_total.value -= 1

    if db.getBruteBlockQuantity(testName, False) == 0:

```

```

        endSignal.value = 1
        return
    brute_combinations_left.value -= 1
    time.sleep(timeout)
for i in range(numProcess):
    queueBrute.put(None)
endSignal.value = 1
return
def processBrute(queueBrute, queueResBrute):
    while True:
        val = queueBrute.get()
        if val == None:
            return
        login = val[1].split(sep='___')[0]
        url = val[1].split(sep='___')[1]
        res = connect(val[0], login, url)
        queueResBrute.put([val[0], val[1], res])

# DataManager.py
import os
import pickle

SEPARATOR = '\\\
'
ROOT = ''
DATA = ROOT + 'data' + SEPARATOR
RANGES = ROOT + 'ranges' + SEPARATOR
RANGES_CONTINUE = RANGES + 'continue' + SEPARATOR
RESULTS = ROOT + 'results' + SEPARATOR
RESULTS_IMG = RESULTS + 'img' + SEPARATOR
DATA_BACKUP = DATA + 'backup' + SEPARATOR

class CamInBrute:
    def __init__(self):
        self.ip = ''
        self.block = False
        self.url = []
class Result:
    def __init__(self):
        self.ip = ''
        self.url = ''
        self.img = True
        self.alive = True

class TestBlock:
    def __init__(self):
        self.name = ''
        self.inBrute = []

```

```

self.results = []

class dataManager:
    fileNames = ['data', 'urls.txt']
    def SetUp(self):
        if not os.path.exists(DATA[:-1]):
            os.mkdir(DATA[:-1])
        if not os.path.exists(RANGES[:-1]):
            os.mkdir(RANGES[:-1])
        if not os.path.exists(RANGES_CONTINUE[:-1]):
            os.mkdir(RANGES_CONTINUE[:-1])
        if not os.path.exists(RESULTS[:-1]):
            os.mkdir(RESULTS[:-1])
        if not os.path.exists(RESULTS_IMG[:-1]):
            os.mkdir(RESULTS_IMG[:-1])
        if not os.path.exists(DATA_BACKUP[:-1]):
            os.mkdir(DATA_BACKUP[:-1])

        for n in self.fileNames:
            if not os.path.exists(DATA + n):
                f = open(DATA + n, 'w')
                f.close()

    def ReadUrls(self):
        f = open(DATA + 'urls.txt', "r")
        urlsList = []
        for x in f:
            x = x.rstrip()
            if len(x.split(sep='___')) != 2 or len(x.split(sep='___')[0].split(sep=':'))
!= 2:
                return None
            urlsList.append(x.rstrip())
        f.close()
        return urlsList

    def addTestBlock(self, name):
        f = open(DATA + 'data', 'rb')
        while True:
            try:
                temp = pickle.load(f)
                if temp.name == name:
                    f.close()
                    return False
            except EOFError:
                f.close()
                break
        f.close()

```

```

obj = TestBlock()
obj.name = name
f = open(DATA + 'data', 'ab')
pickle.dump(obj, f)
f.close()

def addToBrute(self, testName, ip, url, block = False):
    obj = CamInBrute()
    obj.ip = ip
    obj.url = url
    obj.block = block

    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                temp.inBrute.append(obj)
                tempL.append(temp)
                continue
            tempL.append(temp)
        except EOFError:
            break
    f.close()

    f = open(DATA + 'data', 'wb')
    for i in tempL:
        pickle.dump(i, f)
    f.close()

def addPassedUrl(self, testName, ip, url):
    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in range(len(temp.inBrute)):
                    if temp.inBrute[i].ip == ip:
                        temp.inBrute[i].url.append(url)
                        break
                tempL.append(temp)
                continue
            tempL.append(temp)
        except EOFError:
            break
    f.close()

```

```

f = open(DATA + 'data', 'wb')
for i in tempL:
    pickle.dump(i, f)
f.close()

def addResult(self, testName, ip, url, img = True, alive = True):
    obj = Result()
    obj.ip = ip
    obj.url = url
    obj.alive = alive
    obj.img = img

    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                temp.results.append(obj)
                tempBrute = []
                for i in temp.inBrute:
                    if i.ip == ip:
                        continue
                    tempBrute.append(i)
                temp.inBrute = tempBrute
                tempL.append(temp)
            continue
        except EOFError:
            break
    f.close()

    f = open(DATA + 'data', 'wb')
    for i in tempL:
        pickle.dump(i, f)
    f.close()

def setBlock(self, testName, ip, yesNo = True):
    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in range(len(temp.inBrute)):
                    if temp.inBrute[i].ip == ip:
                        temp.inBrute[i].block = yesNo
                break
        except EOFError:
            break

```

```

        tempL.append(temp)
        continue
    tempL.append(temp)
except EOFError:
    break
f.close()

f = open(DATA + 'data', 'wb')
for i in tempL:
    pickle.dump(i, f)
f.close()

def setNotAlive(self, testName, ip):
    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in range(len(temp.results)):
                    if temp.results[i].ip == ip:
                        temp.results[i].alive = False
                        break
                tempL.append(temp)
                continue
            tempL.append(temp)
        except EOFError:
            break
    f.close()

f = open(DATA + 'data', 'wb')
for i in tempL:
    pickle.dump(i, f)
f.close()

def setYseImg(self, testName, ip):
    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in range(len(temp.results)):
                    if temp.results[i].ip == ip:
                        temp.results[i].img = True
                        break
                tempL.append(temp)
                continue
            tempL.append(temp)

```

```

        except EOFError:
            break
    f.close()

    f = open(DATA + 'data', 'wb')
    for i in tempL:
        pickle.dump(i, f)
    f.close()
def setAlive(self, testName, ip):
    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in range(len(temp.results)):
                    if temp.results[i].ip == ip:
                        temp.results[i].alive = True
                        break
                tempL.append(temp)
                continue
            tempL.append(temp)
        except EOFError:
            break
    f.close()

    f = open(DATA + 'data', 'wb')
    for i in tempL:
        pickle.dump(i, f)
    f.close()

def getResultIpWithImg(self, testName, alive = True, yesImg = True, getQuantity =
False, totalImg = False):
    ipList = []
    urlList = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.results:
                    if i.img == yesImg and i.alive == alive or totalImg and i.img:
                        ipList.append(i.ip)
                        urlList.append(i.url)
        except EOFError:
            f.close()
            if not getQuantity:

```

```

        return ipList, urlList
    return len(ipList)
def getResultAlive(self, testName, yes = True, getQuantity = False):
    ipList = []
    urlList = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.results:
                    if i.alive == yes:
                        ipList.append(i.ip)
                        urlList.append(i.url)
        except EOFError:
            f.close()
            if not getQuantity:
                return ipList, urlList
            return len(ipList)
def getResultUrl(self, testName, ip):
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.results:
                    if i.ip == ip:
                        f.close()
                        return i.url
        except EOFError:
            f.close()
            return None
def getResultAll(self, testName):
    ips = []
    urls = []
    img = []
    alive = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.results:
                    ips.append(i.ip)
                    urls.append(i.url)
                    img.append(i.img)
                    alive.append(i.alive)

```

```

        except EOFError:
            f.close()
            return ips, urls, img, alive
def getResultQuantity(self, testName):
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                return len(temp.results)
        except EOFError:
            f.close()

def getBruteWithNoPassedUrl(self, testName, url, availableOnli = True):
    data = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.inBrute:
                    if availableOnli and not i.block or not availableOnli:
                        yesNo = True
                        for j in i.url:
                            if j == url:
                                yesNo = False
                                break
                        if yesNo:
                            data.append(i.ip)
        except EOFError:
            f.close()
            return data
def getBruteBlockQuantity(self, testName, yes = True):
    ipList = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.inBrute:
                    if i.block == yes:
                        ipList.append(i.ip)
        except EOFError:
            f.close()
            return len(ipList)
def getBruteQuantity(self, testName):
    f = open(DATA + 'data', 'rb')

```

```

while True:
    try:
        temp = pickle.load(f)
        if temp.name == testName:
            return len(temp.inBrute)
    except EOFError:
        f.close()

def getNoPassedUrlList(self, testName, urlList):
    NoPassedUrlList = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for url in urlList:
                    yesNo = True
                    for i in temp.inBrute:
                        yesNo = True
                        if not i.block:
                            for j in i.url:
                                if j == url:
                                    yesNo = False
                                    break
                            if yesNo:
                                NoPassedUrlList.append(url)
                                break
                    #if yesNo:
                    #    NoPassedUrlList.append(url)
        except EOFError:
            f.close()
            return NoPassedUrlList

def getBruteBlockForCheck(self, testName):
    urlList = []
    ipList = []
    urls = self.ReadUrls()
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.inBrute:
                    if i.block:
                        for url in urls:
                            yesNo = True
                            for j in i.url:
                                if j == url:
                                    yesNo = False

```

```

                break
            if yesNo:
                urlList.append(url)
                ipList.append(i.ip)
                break
        except EOFError:
            f.close()
            return ipList, urlList

def getTestNameList(self):
    data = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            data.append(temp.name)
        except EOFError:
            f.close()
            return data

def removeTestBlock(self, testName):
    tempL = []
    f = open(DATA + 'data', 'rb')
    while True:
        try:
            temp = pickle.load(f)
            if temp.name == testName:
                for i in temp.results:
                    ip = i.ip.replace(".", "_")
                    if i.img and os.path.exists(RESULTS_IMG + ip + '.jpg'):
                        os.remove(RESULTS_IMG + ip + '.jpg')
                continue
            tempL.append(temp)
        except EOFError:
            break
    f.close()

    f = open(DATA + 'data', 'wb')
    for i in tempL:
        pickle.dump(i, f)
    f.close()

def backup(self):
    import shutil
    from os import listdir
    from os.path import isfile, join
    if os.path.exists(DATA_BACKUP + 'data'):
        os.remove(DATA_BACKUP + 'data')

```

```

        shutil.copy2(DATA + 'data', DATA_BACKUP)

\\ main.qml

ApplicationWindow {
    visible: true
    id: wind
    width: 1000
    height: 600
    color: "#1d1d34"
    title: qsTr("CameRAadar")

    property bool running: false;
    function append(newElement) {
        listModel.append(newElement)
    }
    function appendImgTop(newElement) {
        listModel.insert(0, newElement)
    }
    function rangeError(){
        rangesRectangle.border.color = 'red'
        end()
    }
    function rangeNotError(){
        rangesRectangle.border.color = '#3e3eb8'
    }
    function nameError(){
        testNameBackground.border.color = 'red'
        end()
    }
    function nameNotError(){
        testNameBackground.border.color = '#3e3eb8'
    }
    function loadResultsWindow(){
        if(!running){
            testQThread.laodPage()
            wind.close()
        }
    }

    function end(){
        cont.isEnabled = false
        stop.isEnabled = false
        fastStop.isEnabled = false
        start.isEnabled = true
        running = false
    }
}

```

```

function startF(){
    testQThread.setValue(testName.text, ranges.text, threads.val, timeout.val)
    testQThread.start()
    stop.isEnabled = true
    fastStop.isEnabled = true
    start.isEnabled = false
    running = true
}
function stopF(){
    testQThread.stop()
    stop.isEnabled = false
    fastStop.isEnabled = false
    cont.isEnabled = false
}
function contF(){
    cont.isEnabled = false
    stop.isEnabled = true
    testQThread.cont()
}
function pauseF(){
    testQThread.pause()
    cont.isEnabled = true
    stop.isEnabled = false
}

```

```

Item {
    id: 1
    anchors.top: parent.top
    anchors.left: wind.left

    width: wind.width / 3
    height: wind.height// - 50
    Rectangle {
        id: rangesRectangle
        width: parent.width
        height: parent.height
        border.color: "#3e3eb8"
        border.width: 2
        color: "#11111f"
        radius: 25

        ScrollView {
            id: view
            anchors.fill: parent

            TextArea {

```

```

        id: ranges
        placeholderText: qsTr("12.40.71.32-
12.40.71.39\n24.120.200.0\n24.120.201.255\n98.160.217.0-98.160.217.127")
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 40
        text: ""
        selectByMouse: true
        objectName: "foo_object"
    }
}
}
Item {
    id: m
    anchors.top: parent.top
    anchors.left: l.right

    width: wind.width / 3
    height: wind.height// - 50
    Rectangle {
        id: rec
        width: parent.width
        height: parent.height
        //border.color: "red"
        //border.width: 2
        color: "transparent"

        Image {
            id: icon
            //visible: false
            source: 'iconList.png'
            sourceSize.width: (m.width / 7)
            //anchors.horizontalCenter: r.horizontalCenter
            anchors.top: parent.top
            anchors.left: parent.left
            anchors.topMargin: 5
            anchors.leftMargin: 5
            MouseArea {
                anchors.fill: parent
                onClicked: {
                    loadResultsWindow()
                }
            }
        }
    }

    Text {
        id: text1

```

```

        text: "Залишилось"
        color: "#cfcfcf"
        anchors.top: parent.top
        anchors.horizontalCenter: parent.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 25
    }

    Text {
        id: text2
        text: values.time
        color: "#cfcfcf"
        anchors.top: text1.bottom
        anchors.horizontalCenter: parent.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 25
    }

    CircularProgressBar {
        id: progress2
        lineWidth: 8
        value: values.progress
        size: ((m.width + m.height) / 2) / 2.5
        secondaryColor: "#e0e0e0"
        primaryColor: "#ab47bc"
        anchors.top: text2.bottom
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.topMargin: (m.height - (((cont.height * 3) + threadsText.height +
(rec.height / 60) * 4) + testName.height + (text1.height * 2))) / 5
        Text {
            text: parseInt(progress2.value * 100) + "%"
            anchors.centerIn: parent
            font.pointSize: 20
            color: progress2.primaryColor
        }
    }

    Text {
        id: text3
        text: "Залишилось"
        color: "#cfcfcf"
        anchors.horizontalCenter: progress2.left
        anchors.verticalCenter: progress2.top
        anchors.verticalCenterOffset: - 30
        anchors.horizontalCenterOffset: 20
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    Text {
        id: text4
        text: values.ips

```

```

        color: "#cfcfcf"
        anchors.top: text3.bottom
        anchors.horizontalCenter: text3.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    Text {
        id: text5
        text: "Відкритий порт"
        color: "#cfcfcf"
        anchors.horizontalCenter: progress2.right
        anchors.verticalCenter: progress2.top
        anchors.verticalCenterOffset: - 30
        anchors.horizontalCenterOffset: - 20
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    Text {
        id: text6
        text: values.open
        color: "#cfcfcf"
        anchors.top: text5.bottom
        anchors.horizontalCenter: text5.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    Text {
        id: text7
        text: "Вразливих"
        color: "#cfcfcf"
        anchors.horizontalCenter: progress2.right
        anchors.verticalCenter: progress2.bottom
        anchors.verticalCenterOffset: 10
        anchors.horizontalCenterOffset: - 20
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    Text {
        id: text8
        text: values.authorized
        color: "#cfcfcf"
        anchors.top: text7.bottom
        anchors.horizontalCenter: text7.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    Text {

```

```

        id: text9
        text: "Зображень"
        color: "#cfcfcf"
        anchors.horizontalCenter: progress2.left
        anchors.verticalCenter: progress2.bottom
        anchors.verticalCenterOffset: 10
        anchors.horizontalCenterOffset: 20
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    Text {
        id: text10
        text: values.captured
        color: "#cfcfcf"
        anchors.top: text9.bottom
        anchors.horizontalCenter: text9.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35
    }

    TextField {
        id: testName
        placeholderText: qsTr("Назва")
        width: rec.width - (rec.width / 15) * 2
        height: fastStop.height
        anchors.bottom: rec.bottom
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottomMargin: (cont.height * 3) + threadsText.height +
(rec.height / 60) * 4
        anchors.leftMargin: rec.width / 15
        horizontalAlignment: TextInput.AlignHCenter
        maximumLength: 27
        background: Rectangle {
            id: testNameBackground
            color: "#2f2f54"
            border.width: 2
            border.color: "#3e3eb8"
            radius: 90
        }
        //contentItem: Text{
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        //}
    }

    CustomButton {
        id: fastStop
        heightVal: rec.height / 15
        widthVal: rec.width / 2.5
    }

```

```

        runFunc: stopF

        textVal: 'Зупинити'
        anchors.bottom: rec.bottom
        anchors.left: rec.left
        anchors.leftMargin: rec.width / 15
        anchors.bottomMargin: rec.height / 60
    }

    Text {
        id: threadsText
        text: "Потоки"
        color: "#cfcfcf"
        anchors.horizontalCenter: threads.horizontalCenter
        anchors.bottom: threads.top
        font.pointSize: ((m.width + m.height) / 2) / 45
        anchors.bottomMargin: 5
    }

    CustomSpinBox{
        id: threads
        widthVal: rec.width / 2.5
        heightVal: rec.height / 15

        val: 400
        valMin: 50
        valMax: 1000
        step: 50

        anchors.bottom: rec.bottom
        anchors.left: rec.left
        anchors.bottomMargin: cont.height + stop.height + (rec.height / 60) * 3
        anchors.leftMargin: rec.width / 15
    }

    CustomButton {
        id: start
        heightVal: rec.height / 15
        widthVal: rec.width / 2.5
        runFunc: startF
        isEnabled: true

        textVal: 'Старт'
        anchors.bottom: rec.bottom
        anchors.left: rec.left
        anchors.bottomMargin: stop.height + (rec.height / 60) * 2
        anchors.leftMargin: rec.width / 15
    }

```

```

Text {
    id: timeoutText
    text: "Очікування відповіді"
    color: "#cfcfcf"
    anchors.horizontalCenter: timeout.horizontalCenter
    anchors.bottom: timeout.top
    font.pointSize: ((m.width + m.height) / 2) / 45
    anchors.bottomMargin: 5
}

CustomSpinBox{
    id: timeout
    widthVal: rec.width / 2.5
    heightVal: rec.height / 15

    val: 1
    valMin: 0.1
    valMax: 5
    step: 0.1

    anchors.bottom: rec.bottom
    anchors.right: rec.right
    anchors.bottomMargin: cont.height + stop.height + (rec.height / 60) * 3
    anchors.rightMargin: rec.width / 15
}

CustomButton {
    id: cont
    heightVal: rec.height / 15
    widthVal: rec.width / 2.5
    runFunc: contF

    textVal: 'Продовжити'
    anchors.bottom: rec.bottom
    anchors.right: rec.right
    anchors.bottomMargin: stop.height + (rec.height / 60) * 2
    anchors.rightMargin: rec.width / 15
}

CustomButton {
    id: stop
    heightVal: rec.height / 15
    widthVal: rec.width / 2.5
    runFunc: pauseF

    textVal: 'Пауза'

```

```

        anchors.bottom: rec.bottom
        anchors.right: rec.right
        anchors.rightMargin: rec.width / 15
        anchors.bottomMargin: rec.height / 60
    }
}
}
Item {
    id: r
    anchors.top: parent.top
    anchors.left: m.right

    width: wind.width / 3
    height: wind.height// - 50
    Rectangle {
        width: parent.width
        height: parent.height
        border.color: "#3e3eb8"
        border.width: 2
        color: "#11111f"
        radius: 25

        ListView {
            id: listview
            width: parent.width - 8
            height: parent.height
            spacing: 10
            anchors {
                horizontalCenter: parent.horizontalCenter
                bottom: parent.bottom
            }
            delegate: Item
            {
                Image {
                    id: getImgSize
                    visible: false
                    source: sourceImg
                    sourceSize.width: (wind.width / 3) - 8
                }
                width: (wind.width / 3) - 5
                height: getImgSize.height + r.height / 8
                Text {
                    id: url
                    visible: false
                    text: urlValue
                }
                Image {

```

```

source: sourceImg
sourceSize.width: (wind.width / 3) - 8
anchors {
    left: parent.left
    top: parent.top
}
Rectangle {
    anchors.top: parent.bottom
    width: (wind.width / 3) - 8
    height: r.height / 8
    color: "#2f2f54"
    radius: 20
    Rectangle {
        width: 20
        height: 20
        color: "#2f2f54"
        anchors {
            left: parent.left
            top: parent.top
        }
    }
}
Rectangle {
    width: 20
    height: 20
    color: "#2f2f54"
    anchors {
        right: parent.right
        top: parent.top
    }
}
Rectangle {
    width: (wind.width / 3) - 8
    height: 2
    anchors.top: parent.top
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.topMargin: parent.height / 2
    color: "#3e3eb8"
}
Rectangle {
    width: 2
    height: parent.height / 2
    anchors.top: parent.top
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.topMargin: parent.height / 2
    color: "#3e3eb8"
}
Item {

```

```

anchors {
    right: parent.right
    top: parent.top
}
width: parent.width
height: parent.height / 2
Text {
    id: ip
    text: ipValue
    color: "#cfcfcf"
    anchors.top: parent.top
    anchors.horizontalCenter: parent.horizontalCenter
    font.pointSize: ((m.width + m.height) / 2) / 35
    anchors.topMargin: (parent.height / 4) - (height / 6)
}
}
Item {
    anchors {
        left: parent.left
        bottom: parent.bottom
    }
    width: parent.width / 2
    height: parent.height / 2
    Text {
        id: login
        text: loginValue
        color: "#cfcfcf"
        anchors.top: parent.top
        anchors.horizontalCenter: parent.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.topMargin: (parent.height / 4) - (height / 6)
    }
}
Item {
    anchors {
        right: parent.right
        bottom: parent.bottom
    }
    width: parent.width / 2
    height: parent.height / 2
    Text {
        id: password
        text: passwordValue
        color: "#cfcfcf"
        anchors.top: parent.top
        anchors.horizontalCenter: parent.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35

```



```

}
property bool aliveOnly: true;
property bool availableOnly: true;
property bool bruteRunning: false;
property bool imgUpdateRunning: false;

function bruteEnd() {
    startBrite.textVal = "Старт"
    bruteRunning = false
    startImgUpdate.isEnabled = true
    startBrite.isEnabled = true
}

function imgUpdateEnd() {
    startImgUpdate.textVal = "Старт"
    imgUpdateRunning = false
    startImgUpdate.isEnabled = true
    startBrite.isEnabled = true
}

function bruteStartStop() {
    if (bruteRunning === false) {
        startBrite.textVal = "Зупинити"
        startImgUpdate.isEnabled = false
        bruteRunning = true
        resultWindowQThread.runBrute(titleText.text, threadsBrite.val, timeoutBrite.val,
availableOnlyId.val)
        resultWindowQThread.start()
    } else {
        resultWindowQThread.stopBrute()
        startBrite.isEnabled = false
    }
}

function imgUpdateStartStop() {
    if (imgUpdateRunning === false) {
        startImgUpdate.textVal = "Зупинити"
        startBrite.isEnabled = false
        imgUpdateRunning = true
        resultWindowQThread.runUpdateImg(titleText.text, threads.val, aliveOnlyId.val)
        imgListModel.clear()
        resultWindowQThread.start()
    } else {
        resultWindowQThread.stopUpdateImg()
        startImgUpdate.isEnabled = false
    }
}

function loadTestData(testName){
    if(!bruteRunning && !imgUpdateRunning){

```

```

        resultWindowQThread.getTestData(testName)
        startBrite.isEnabled = true
        startImgUpdate.isEnabled = true
    }
}

function loadMainWindow(){
    if(!bruteRunning && !imgUpdateRunning){
        resultWindowQThread.laodPage()
        wind.close()
    }
}

function removeTest(testName){
    if(!bruteRunning && !imgUpdateRunning){
        resultWindowQThread.removeTestBlock(testName)
    }
}

function buttonsOff(){
    startBrite.isEnabled = false
    startImgUpdate.isEnabled = false
}

Item {
    id: 1
    anchors.top: parent.top
    anchors.left: wind.left

    width: wind.width / 3
    height: wind.height
    Rectangle {
        width: parent.width
        height: parent.height
        border.color: "#3e3eb8"
        border.width: 2
        color: "#11111f"
        radius: 25
        ListView {
            id: listview
            width: parent.width - 8
            height: parent.height
            spacing: 5
            anchors {
                horizontalCenter: parent.horizontalCenter
                bottom: parent.bottom
            }
            delegate: Item
            {
                id: item

```

```

anchors.left: parent.left
anchors.right: parent.right
height: 50
Rectangle {
    width: parent.width - 4
    height: 50
    radius: 25
    color: "#2f2f54"
    border.width: 2
    border.color: "#3e3eb8"
    anchors.horizontalCenter: parent.horizontalCenter

    Image {
        id: iconRemove
        source: 'iconRemove.png'
        sourceSize.width: parent.height / 1.4
        anchors.left: parent.left
        anchors.verticalCenter: parent.verticalCenter
        anchors.leftMargin: 5
        MouseArea {
            anchors.fill: parent
            onClicked: {
                removeTest(testName)
            }
        }
    }
}

Text {
    id: testNameId
    text: testName
    color: "#cfcfcf"
    font.pointSize: parent.height / 3
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.verticalCenter: parent.verticalCenter
}

MouseArea {
    //anchors.fill: parent
    width: parent.width - iconRemove.width
    height: parent.height
    anchors.right: parent.right
    onClicked: {
        loadTestData(testNameId.text)
    }
}
}

model: ListModel {

```

```

        id: testListModel
        objectName: "testListObject"
    }
}
}
}
Item {
    id: m
    anchors.top: parent.top
    anchors.left: l.right

    width: wind.width / 3
    height: wind.height// - 50
    Rectangle {
        id: rec
        width: parent.width
        height: parent.height
        color: "transparent"

        Image {
            id: icon
            source: 'iconReturn.png'
            sourceSize.width: (m.width / 7)
            anchors.top: parent.top
            anchors.left: parent.left
            anchors.topMargin: 19
            anchors.leftMargin: 5
            MouseArea {
                anchors.fill: parent
                onClicked: {
                    loadMainWindow()
                }
            }
        }
    }

    Rectangle {
        id: title
        width: parent.width / 1.3
        height: titleText.height * 2
        radius: 25
        color: "#2f2f54"
        border.width: 2
        border.color: "#3e3eb8"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.top
        Text {
            id: titleText

```

```

        text: valuesPg2.name
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 25
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottom: parent.bottom
    }
}
Rectangle {
    id: authorized
    width: m.width - 15
    height: m.height / 2.9
    radius: 25
    color: "#1d1d34"
    border.width: 2
    border.color: "#3e3eb8"
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.top: title.bottom
    anchors.topMargin: authorizedTextFrame.height
    Rectangle {
        id: authorizedTextFrame
        width: parent.width / 1.3
        height: 40
        radius: 25
        color: "#1d1d34"
        border.width: 2
        border.color: "#3e3eb8"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.top
        Text {
            id: authorizedText
            text: "Вразливі"
            color: "#cfcfcf"
            font.pointSize: ((m.width + m.height) / 2) / 29
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.verticalCenter: parent.verticalCenter
        }
        Text {
            id: totalAuthorized
            text: valuesPg2.result_total
            color: "#cfcfcf"
            font.pointSize: ((m.width + m.height) / 2) / 35
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.top: authorizedText.bottom
            anchors.topMargin: 5
        }
    }
}
Text {

```

```

        id: authorizedTextImg
        text: "Зображень"
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.top: parent.top
        anchors.topMargin: 75
        anchors.left: parent.left
        anchors.leftMargin: rec.width / 50
        Text {
            id: authorizedCountImg
            text: valuesPg2.result_img
            color: "#cfcfcf"
            font.pointSize: ((m.width + m.height) / 2) / 35
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.top: parent.bottom
        }
    }
    Text {
        id: authorizedTextAliveImg
        text: "Вразливі із зображенням"
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 44
        Text {
            id: authorizedCountAliveImg
            text: valuesPg2.result_alive_with_img
            color: "#cfcfcf"
            font.pointSize: ((m.width + m.height) / 2) / 35
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.top: parent.bottom
        }
    }
    Text {
        id: authorizedTextAlive
        text: "Вразливі"
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.top: parent.top
        anchors.topMargin: 75
        anchors.right: parent.right
        anchors.rightMargin: rec.width / 50
        Text {
            id: authorizedCountAlive
            text: valuesPg2.result_alive
            color: "#cfcfcf"

```

```

        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.bottom
    }
}
Text {
    id: threadsText
    text: "Потоки"
    color: "#cfcfcf"
    anchors.horizontalCenter: threads.horizontalCenter
    anchors.bottom: threads.top
    font.pointSize: ((m.width + m.height) / 2) / 45
    anchors.bottomMargin: 5
}

CustomSpinBox{
    id: threads
    widthVal: rec.width / 2.5
    heightVal: rec.height / 15

    val: 50
    valMin: 10
    valMax: 100
    step: 10

    anchors.bottom: parent.bottom
    anchors.left: parent.left
    anchors.bottomMargin: 10
    anchors.leftMargin: rec.width / 50
}

CustomCheckBox{
    id: aliveOnlyId
    textVal: "Тільки вразливі"
    width: startBrite.width
    height: startBrite.height / 2
    val: true

    anchors.bottom: startImgUpdate.top
    anchors.horizontalCenter: startImgUpdate.horizontalCenter
}

CustomButton {
    id: startImgUpdate
    heightVal: rec.height / 15
    widthVal: rec.width / 2.5
    runFunc: imgUpdateSartStop

```

```

        isEnabled: false

        textVal: 'Cтapт'
        anchors.bottom: parent.bottom
        anchors.right: parent.right
        anchors.bottomMargin: 10
        anchors.rightMargin: rec.width / 50
    }

}

Rectangle {
    id: bruteForce
    width: m.width - 15
    height: m.height / 2.2
    radius: 25
    color: "#1d1d34"
    border.width: 2
    border.color: "#3e3eb8"
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.top: authorized.bottom
    anchors.topMargin: authorizedTextFrame.height
    Rectangle {
        id: bruteForceTextFrame
        width: parent.width / 1.3
        height: 40
        radius: 25
        color: "#1d1d34"
        border.width: 2
        border.color: "#3e3eb8"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.top
        Text {
            id: bruteForceText
            text: "Бпытфopc"
            color: "#cfcfcf"
            font.pointSize: ((m.width + m.height) / 2) / 29
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.verticalCenter: parent.verticalCenter
        }
    }
    Text {
        id: totalBruteForce
        text: valuesPg2.brute_total
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: bruteForceTextFrame.bottom
        anchors.topMargin: 5
    }
}

```

```

    }
}
Text {
    id: bruteForceTextAvailable
    text: "Доступні"
    color: "#cfcfcf"
    font.pointSize: ((m.width + m.height) / 2) / 35
    anchors.top: parent.top
    anchors.topMargin: 75
    anchors.left: parent.left
    anchors.leftMargin: rec.width / 50
    Text {
        id: bruteForceCountAvailable
        text: valuesPg2.brute_available
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.bottom
    }
}
Text {
    id: bruteForceTextImg
    text: "Доступно комбінацій"
    color: "#cfcfcf"
    font.pointSize: ((m.width + m.height) / 2) / 35
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.top: parent.top
    anchors.topMargin: 44
    Text {
        id: bruteForceCountImg
        text: valuesPg2.brute_combinations_left
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.bottom
    }
}
Text {
    id: bruteForceTextBlock
    text: "Недоступні"
    color: "#cfcfcf"
    font.pointSize: ((m.width + m.height) / 2) / 35
    anchors.top: parent.top
    anchors.topMargin: 75
    anchors.right: parent.right
    anchors.rightMargin: rec.width / 50
    Text {

```

```

        id: bruteForceCountBlock
        text: valuesPg2.brute_block
        color: "#cfcfcf"
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.bottom
    }
}

Text {
    text: "Тайм-аут"
    color: "#cfcfcf"
    anchors.horizontalCenter: threadsBrite.horizontalCenter
    anchors.bottom: timeoutBrite.top
    font.pointSize: ((m.width + m.height) / 2) / 45
    anchors.bottomMargin: 5
}

CustomSpinBox{
    id: timeoutBrite
    widthVal: rec.width / 2.5
    heightVal: rec.height / 15

    val: 10
    valMin: 2
    valMax: 120
    step: 2

    anchors.bottom: threadsTextBrute.top
    anchors.left: parent.left
    anchors.bottomMargin: 5
    anchors.leftMargin: rec.width / 50
}

Text {
    id: threadsTextBrute
    text: "ПотокиВ"
    color: "#cfcfcf"
    anchors.horizontalCenter: threadsBrite.horizontalCenter
    anchors.bottom: threadsBrite.top
    font.pointSize: ((m.width + m.height) / 2) / 45
    anchors.bottomMargin: 5
}

CustomSpinBox{
    id: threadsBrite
    widthVal: rec.width / 2.5
    heightVal: rec.height / 15

```

```

        val: 200
        valMin: 50
        valMax: 500
        step: 50

        anchors.bottom: parent.bottom
        anchors.left: parent.left
        anchors.bottomMargin: 10
        anchors.leftMargin: rec.width / 50
    }

    CustomCheckBox{
        id: availableOnlyId
        textVal: "Тільки доступні"
        width: startBrite.width
        height: startBrite.height / 2
        val: true

        anchors.bottom: startBrite.top
        anchors.horizontalCenter: startBrite.horizontalCenter
    }

    CustomButton {
        id: startBrite
        heightVal: rec.height / 15
        widthVal: rec.width / 2.5
        runFunc: bruteSartStop
        isEnabled: false

        textVal: 'Старт'
        anchors.bottom: parent.bottom
        anchors.right: parent.right
        anchors.bottomMargin: 10
        anchors.rightMargin: rec.width / 50
    }
}

}

}

Item {
    id: r
    anchors.top: parent.top
    anchors.left: m.right

    width: wind.width / 3

```

```

height: wind.height// - 50
Rectangle {
    width: parent.width
    height: parent.height
    border.color: "#3e3eb8"
    border.width: 2
    color: "#11111f"
    radius: 25
    ListView {
        id: listviewImg
        width: parent.width - 8
        height: parent.height
        spacing: 10
        anchors {
            horizontalCenter: parent.horizontalCenter
            bottom: parent.bottom
        }
        delegate: Item
        {
            Image {
                id: getImgSize
                visible: false
                source: sourceImg
                sourceSize.width: (wind.width / 3) - 8
            }
            width: (wind.width / 3) - 5
            height: getImgSize.height + r.height / 8
            Text {
                id: url
                visible: false
                text: urlValue
            }
            Image {
                source: sourceImg
                sourceSize.width: (wind.width / 3) - 8
                anchors {
                    left: parent.left
                    top: parent.top
                }
            }
            Rectangle {
                anchors.top: parent.bottom
                width: (wind.width / 3) - 8
                height: r.height / 8
                color: "#2f2f54"
                radius: 20
                Rectangle {
                    width: 20

```

```

        height: 20
        color: "#2f2f54"
        anchors {
            left: parent.left
            top: parent.top
        }
    }
    Rectangle {
        width: 20
        height: 20
        color: "#2f2f54"
        anchors {
            right: parent.right
            top: parent.top
        }
    }
    Rectangle {
        width: (wind.width / 3) - 8
        height: 2
        anchors.top: parent.top
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.topMargin: parent.height / 2
        color: "#3e3eb8"
    }
    Rectangle {
        width: 2
        height: parent.height / 2
        anchors.top: parent.top
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.topMargin: parent.height / 2
        color: "#3e3eb8"
    }
    Item {
        anchors {
            right: parent.right
            top: parent.top
        }
        width: parent.width
        height: parent.height / 2
        Text {
            id: ip
            text: ipValue
            color: "#cfcfcf"
            anchors.top: parent.top
            anchors.horizontalCenter: parent.horizontalCenter
            font.pointSize: ((m.width + m.height) / 2) / 35
            anchors.topMargin: (parent.height / 4) - (height / 6)

```

```

    }
}
Item {
    anchors {
        left: parent.left
        bottom: parent.bottom
    }
    width: parent.width / 2
    height: parent.height / 2
    Text {
        id: login
        text: loginValue
        color: "#cfcfcf"
        anchors.top: parent.top
        anchors.horizontalCenter: parent.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.topMargin: (parent.height / 4) - (height / 6)
    }
}
Item {
    anchors {
        right: parent.right
        bottom: parent.bottom
    }
    width: parent.width / 2
    height: parent.height / 2
    Text {
        id: password
        text: passwordValue
        color: "#cfcfcf"
        anchors.top: parent.top
        anchors.horizontalCenter: parent.horizontalCenter
        font.pointSize: ((m.width + m.height) / 2) / 35
        anchors.topMargin: (parent.height / 4) - (height / 6)
    }
}
}
MouseArea {
    anchors.fill: parent
    onClicked: {
        playVideoQThread.setValue(ip.text, login.text, password.text, url.text)
        playVideoQThread.start()
    }
}
}
}
model: ListModel {

```

```

        id: imgListModel
        objectName: "imgListObject"
    }
}
}
}
}
}

```

\\ CustomSpinBox.qml

```

Item{
    id: root
    property real heightVal
    property real widthVal

    property var val
    property var valMin
    property var valMax
    property var step

    width: widthVal
    height: heightVal

    function plusF() {
        val = parseFloat(val)
        if (val != valMax) {
            val = val + step
            if(!Number.isInteger(val)){
                val = val.toFixed(1)
            }
            textNum.text = val
        }
    }

    function minusF() {
        val = parseFloat(val)
        if (val != valMin) {
            val = val - step
            if(!Number.isInteger(val)){
                val = val.toFixed(1)
            }
            textNum.text = val
        }
    }

    Rectangle {
        id: spinBoxId
        width: widthVal
        height: heightVal
    }
}

```

```

color: "#2f2f54"
border.color: "#3e3eb8"
border.width: 2
radius: 90
Rectangle {
    id: minus
    width: parent.height
    height: parent.height
    anchors.left: parent.left
    color: "transparent"
    radius: 90
    Rectangle {
        width: 25
        height: 4
        radius: 90
        color: "#3e3eb8"
        anchors.verticalCenter: minus.verticalCenter
        anchors.horizontalCenter: minus.horizontalCenter
    }
    MouseArea {
        anchors.fill: parent
        onClicked: {
            minusF()
        }
    }
}
Rectangle {
    width: 2
    height: parent.height
    color: "#3e3eb8"
    anchors.left: minus.right
}
Text {
    id: textNum
    text: val
    font.pointSize: ((widthVal + heightVal) / 2) / 7
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.verticalCenter: parent.verticalCenter
    color: "#cfcfcf"
}
Rectangle {
    id: plus
    width: parent.height
    height: parent.height
    anchors.right: parent.right
    color: "transparent"
    radius: 90

```

```

        Rectangle {
            width: 25
            height: 4
            radius: 90
            color: "#3e3eb8"
            anchors.verticalCenter: plus.verticalCenter
            anchors.horizontalCenter: plus.horizontalCenter
        }
        Rectangle {
            width: 4
            height: 25
            radius: 90
            color: "#3e3eb8"
            anchors.verticalCenter: plus.verticalCenter
            anchors.horizontalCenter: plus.horizontalCenter
        }
        MouseArea {
            anchors.fill: parent
            onClicked: {
                plusF()
            }
        }
    }
    Rectangle {
        width: 2
        height: parent.height
        color: "#3e3eb8"
        anchors.right: plus.left
    }
}

```

\\ CustomCheckBox.qml

```

Rectangle{
    property bool val
    property string textVal
    color: "transparent"

    function valSwitch() {
        if (val === true) {
            val = false
        } else {
            val = true
        }
    }

    Text {

```

```

        text: textVal
        font.pointSize: ((m.width + m.height) / 2) / 45
        color: "#cfcfcf"
        anchors.horizontalCenter: parent.horizontalCenter
        Rectangle {
            height: parent.height
            width: parent.height
            color: "transparent"
            border.width: 2
            border.color: "#3e3eb8"
            radius: 5
            anchors.left: parent.right
            anchors.leftMargin: 3
            anchors.verticalCenter: parent.verticalCenter
            Rectangle {
                visible: val
                height: parent.height / 1.8
                width: parent.height / 1.8
                anchors.centerIn: parent
                color: "#3e3eb8"
            }
        }
    }
}

MouseArea {
    anchors.fill: parent
    onClicked: {
        valSwitch()
    }
}
}

```

\\ CustomButton.qml

```

import QtQuick 2.9
import QtQuick.Controls 2.15

Item{
    id: root
    property real heightVal
    property real widthVal

    property string textVal
    property var runFunc
    property bool isEnabled

    width: widthVal
    height: heightVal

```

```

Button {
    id: buttonId
    text: qsTr(textVal)
    width: widthVal
    height: heightVal
    enabled: isEnabled

    font.pointSize: ((widthVal + heightVal) / 2) / 7
    contentItem: Text {
        text: buttonId.text
        font: buttonId.font
        opacity: enabled ? 1.0 : 0.3
        color: buttonId.down ? "#2f2f54" : "#cfcfcf"
        horizontalAlignment: Text.AlignHCenter
        verticalAlignment: Text.AlignVCenter
        elide: Text.ElideRight
    }

    background: Rectangle {
        opacity: enabled ? 1 : 0.3
        color: enabled ? "#2f2f54" : "#252542"
        border.color: buttonId.down ? "#2f2f54" : "#3e3eb8"
        border.width: 2
        radius: 90
    }
    onClicked: {
        runFunc()
    }
}
}

```

ДОДАТОК Б
Слайди презентації

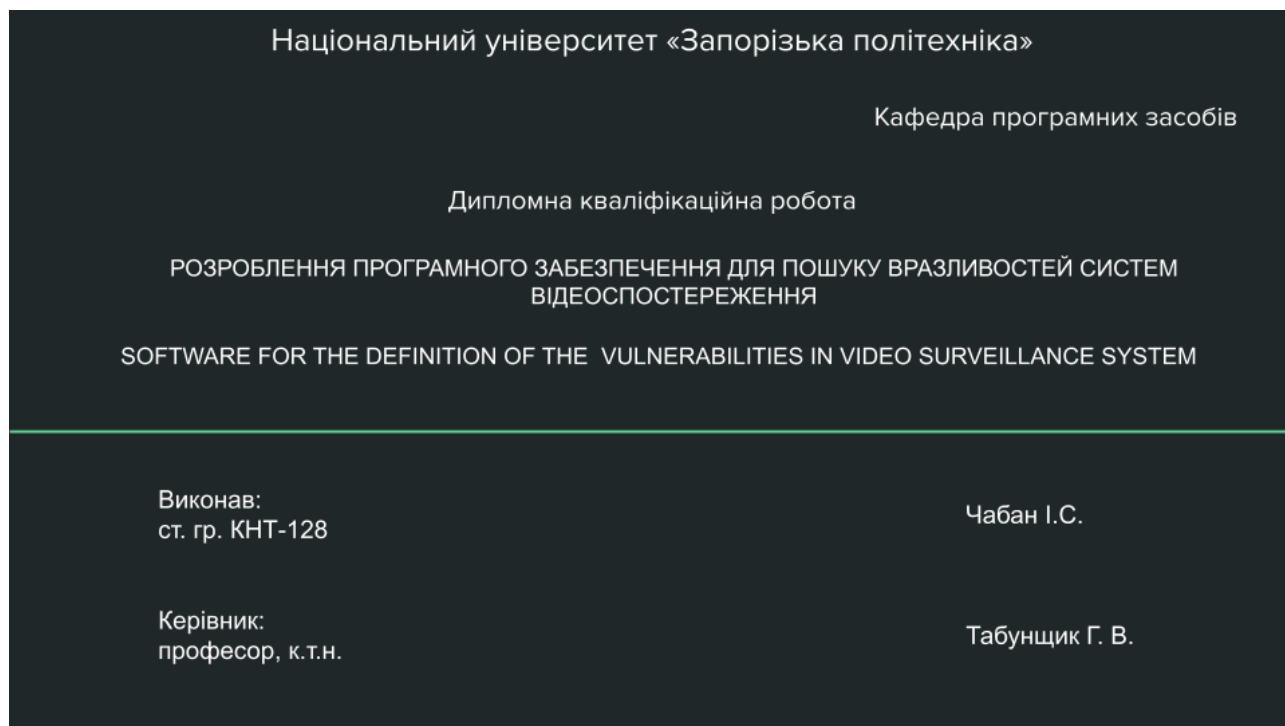


Рисунок Б.1 – Слайд №1

2

Об'єкт, предмет дослідження та мета роботи

- Об'єкт дослідження - вразливості систем відеоспостереження.
- Предмет дослідження - програмне забезпечення для пошуку вразливостей систем відеоспостереження.
- Мета роботи - розробка програмного забезпечення для пошуку вразливостей систем відеоспостереження.

Рисунок Б.2 – Слайд №2

Актуальні вразливості систем відеоспостереження 3

- Вразливість прошивки виробника Hikvision, додавання рядка “?auth=YWRtaW46MTEK” до http-запиту.
- Вразливість стандартних налаштувань виробників Dahua та Hikvision.
- Вразливість стандартних налаштувань систем відеоспостереження які підтримують RTSP протокол.

Рисунок Б.3 – Слайд №3

Вибір вразливості для пошуку 4

Пошук вразливостей буде здійснюватися за протоколом RTSP, скануванням 554 порту та перебором стандартних імені користувача, пароля та URN різних виробників.



Кількість підключених до інтернету пристроїв з відкритим 554 портом



Кількість підключених до інтернету пристроїв з відкритим 554 портом, та кодом відповіді 200

Рисунок Б.4 – Слайд №4

Рейтинг комбінацій

Ім'я користувача	Пароль	URN
admin	12345	/Streaming/Channels/101
admin	admin	/Streaming/Channels/101
root	pass	/axis-media/media.amp
admin	admin	/cam/realmonitor?channel=1&subtype=00&authbasic=[AUTH]
admin	1234	/Streaming/Channels/101
ubnt	ubnt	/live/ch00_0
admin	123456	/stream1
admin	admin	/primarystream
admin	12345	/ONVIF/MediaInput
admin	-	/videoMain

Рисунок Б.1 – Слайд №5

Вибір запиту для пошуку вразливості

Пошук вразливості буде виконуватись за допомогою запиту DESCRIBE

```
DESCRIBE rtsp://192.168.50.73/h264_pcm.sdp RTSP/1.0
CSeq: 1
Authorization: Basic YWRtaW46MTIzNQ==
User-Agent: RTSP Client
Accept: application/sdp

RTSP/1.0 401 Unauthorized
CSeq: 1
Content-Length: 0
WWW-Authenticate: Digest realm="IP Webcam", nonce="1654984107", qop="auth"
```

Запит DESCRIBE з неправильним паролем

```
DESCRIBE rtsp://192.168.50.73/h264_pcm.sd RTSP/1.0
CSeq: 1
Authorization: Basic YWRtaW46MTIzNDU=
User-Agent: RTSP Client
Accept: application/sdp

RTSP/1.0 404 Not Found
Server: IP Webcam Server 0.4
CSeq: 1
```

Запит DESCRIBE з неправильним URN

```
DESCRIBE rtsp://192.168.50.73/h264_pcm.sdp RTSP/1.0
CSeq: 1
Authorization: Basic YWRtaW46MTIzNDU=
User-Agent: RTSP Client
Accept: application/sdp

RTSP/1.0 200 OK
Server: IP Webcam Server 0.4
CSeq: 1
```

Запит DESCRIBE з правильним паролем та URN

Рисунок Б.2 – Слайд №6

Вибір мови програмування

Критерії	Мова програмування			
	Java	C++	C#	Python
Висока швидкість розробки	-	-	+-	+
Читабельність коду	-	-	-	+
Швидкість виконання коду	+-	+	+	-
Велика кількість готових бібліотек	-	-	-	+

Рисунок Б.3 – Слайд №7

Вибір бібліотеки для створення графічного інтерфейсу

Критерії	Бібліотеки			
	wxPython	Tkinter	Kivy	PyQt5 QML
Гнучкість кастомізації	-	-	+-	+
Сучасний дизайн	-	-	+-	+
Стандартна	-	+	-	-
Підтримка javascript	-	-	-	+

Рисунок Б.4 – Слайд №8

Вибір бібліотеки для прискорення процесу пошуку вразливостей

9

Критерії	Бібліотеки	
	threading	multiprocessing
Окреме місце в пам'яті	-	+
Використовує переваги кількох процесорів і ядер	-	+
Уникає обмеження GIL (Глобальне блокування інтерпретатора)	-	+
Займає невеликий об'єм пам'яті	+	-
Швидкість запуску	+	-

Рисунок Б.1 – Слайд №9

10

Архітектура бази даних

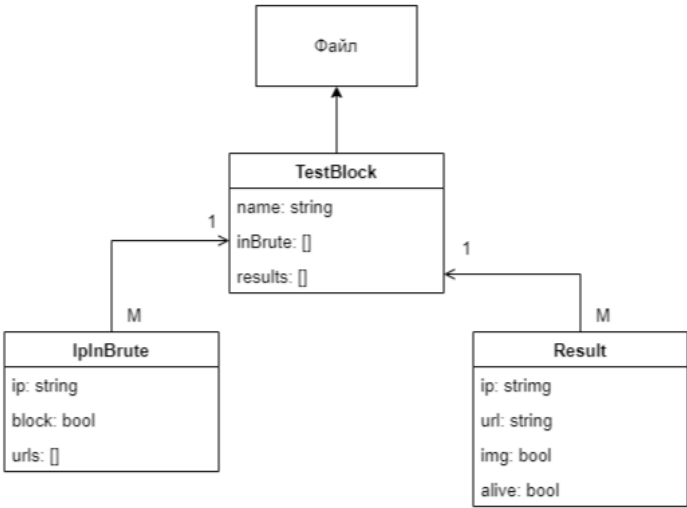


Рисунок Б.2 – Слайд №10

Інтерфейс програми

13

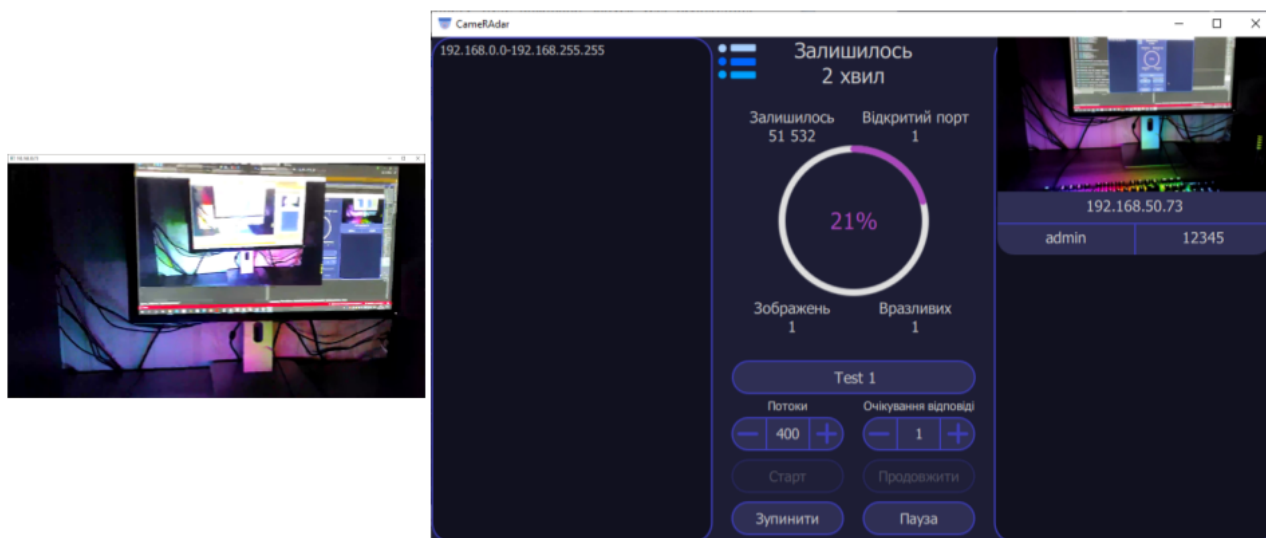


Рисунок Б.1 – Слайд №13

Інтерфейс програми

14

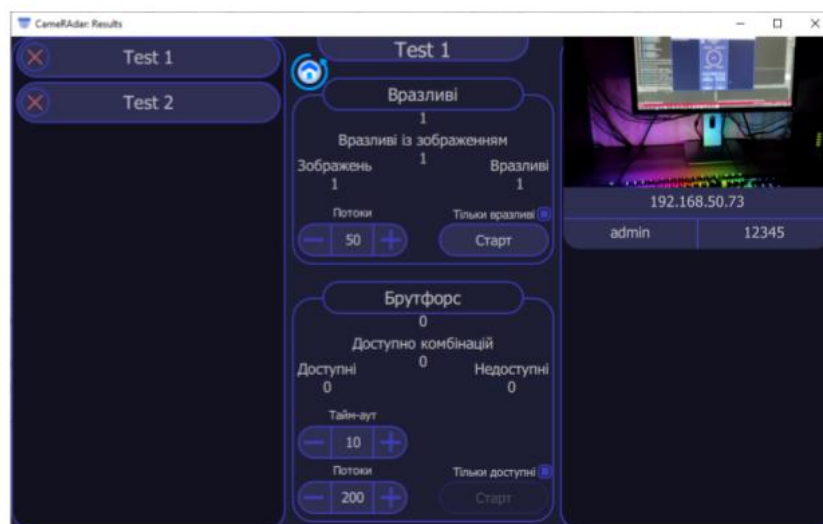


Рисунок Б.2 – Слайд №14

Висновки

- Розглянуто актуальні вразливості та виділено найпоширенішу
- Розглянуто спосіб пошуку вразливостей за допомогою протоколу RTSP
- Розроблено програмне забезпечення для пошуку вразливостей систем відеоспостереження мовою програмування Python, з графічним інтерфейсом який був створений за допомогою декларативної мови програмування QML.

Рисунок Б.3 – Слайд №15