МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное

бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Телевидения и звукового вещания им. С.И. Катаева (ТиЗВ)»

Практическая работа №3

«**Развертывание Docker контейнера Kali Linux для идентификации IP-камер и перебора URI видеопотоков**»

Студент: Салищев Иван Александрович

Группа: БИК-2406

Проверил: Егоров Дмитрий Аркадьевич

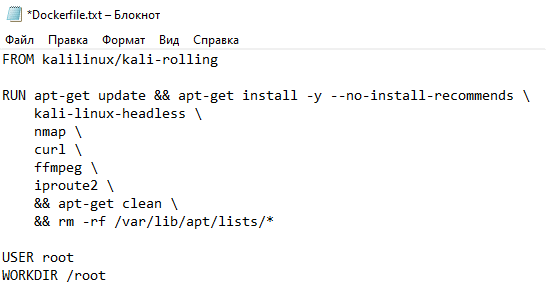
Москва 2025г.

Цель работы:

1. Получить практические навыки работы c Docker.
2. Получить практические навыки сканирования сети с помощью Nmap.
3. Разработать Bash скрипт для автоматического перебора URI видеопотоков.
4. Научиться использовать curl для анализа HTTP ответов.
5. Научиться использовать ffmpeg для захвата кадров из видеопотока.
6. Приобрести опыт в документировании проделанной работы.

Примечание: все время я работал с PowerShell.

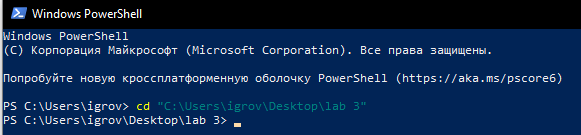
Ход Работы:

1.Создаем Dockerfile и сохраняем без расширения.

* FROM kalilinux/kali-rolling - берем официальный образ Kali Linux.
* RUN ... - обновляем пакеты и устанавливаем нужные утилиты:
  + kali-linux-headless - базовые инструменты Kali.
  + nmap - сканер портов.
  + curl - утилита для HTTP-запросов.
  + ffmpeg - для работы с видео.
  + iproute2 - для сетевых команд (например, ip addr).
* USER root - работаем под root .
* WORKDIR /root - рабочая директория.

2. Создаем папку, в которой будут храниться все наши файлы.

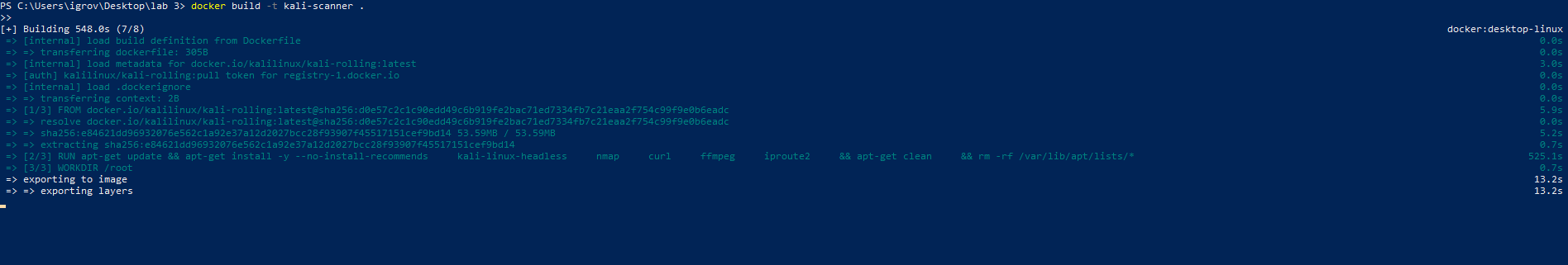
Открываем терминал, в моем случае, это PowerShell.

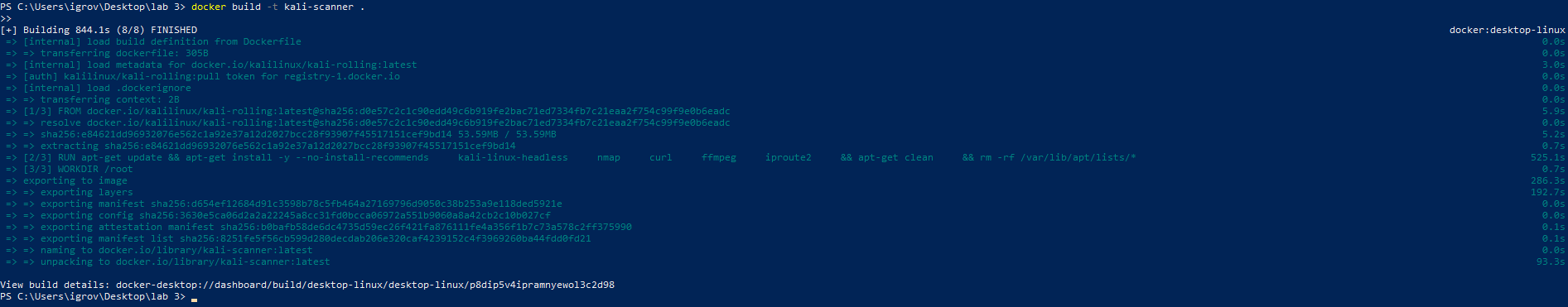
Переходим в нашу папку. 

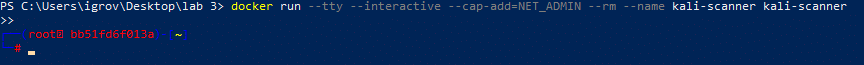
3. Собираем образ kali-scanner

* docker build - команда для сборки образа.
* -t kali-scanner – имя образа.
* . – моя папка(текущая).

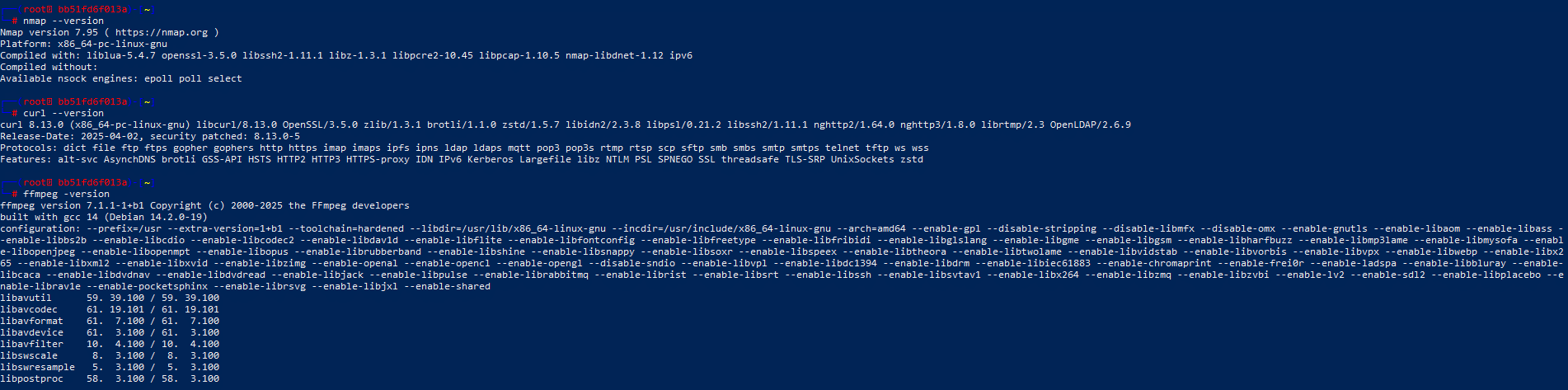
3.1 Промежуточный этап, на моменте сборки (на всякий случай)



3.2 Завершение установки: 

4. Как только сборка завершилась, запускаем контейнер.

* --tty --interactive (-it) - интерактивный режим (терминал, PowerShell).
* --cap-add=NET\_ADMIN - права на сетевые операции.
* --rm - удалить контейнер после выхода.
* --name kali-scanner - имя контейнера.
* kali-scanner - имя образа.

5. Проверяем, что все нужные инструменты(утилиты) установлены. 

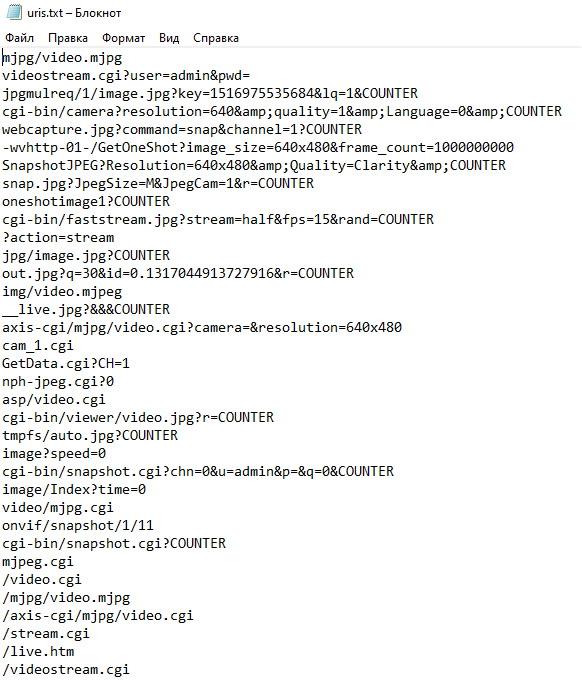
6. Моя подсеть: **78.36.19.0/24,** берем самые популярные порты и запускаем Nmap.

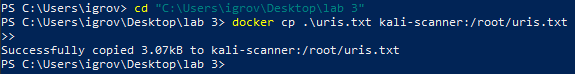


Данный файл сохранится в папке /root, внутри контейнера. Потом его можно скопировать на хост через docker cp.

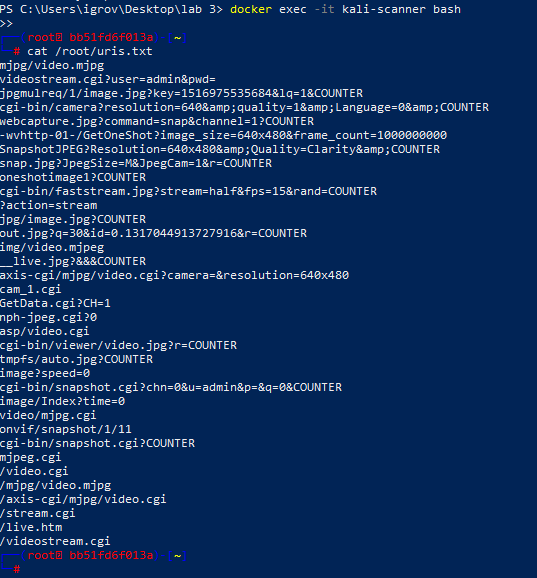
* -Pn - не проверять ICMP (ping), сразу сканировать порты.
* -p ... - список портов (основные для IP-камер).
* 78.36.19.0/24 – мой диапазон.
* -oN nmap\_results.txt - сохраняем вывод в файл.

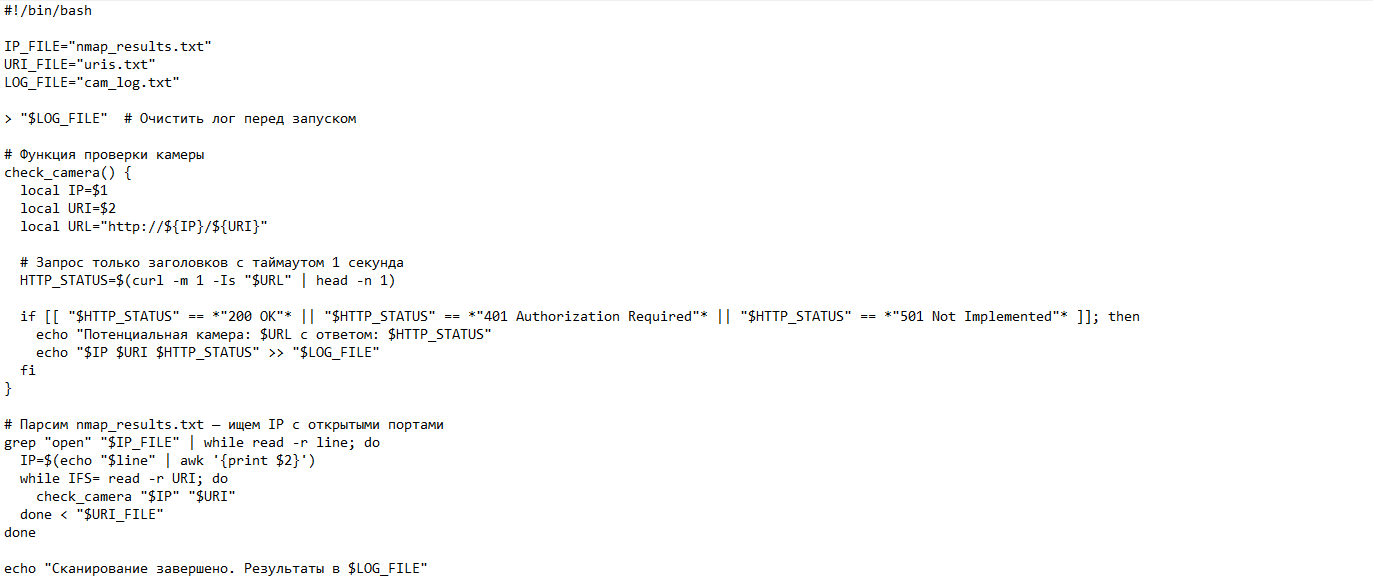
7. Cоздаем файл uris.txt с самыми популярными URL.



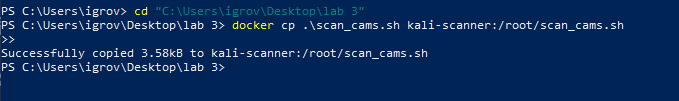
Далее добавляем его в контейнер 

Проверяю его наличие в контейнере:

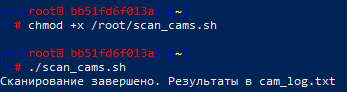


8. Создаем скрипт scan\_cams.sh  


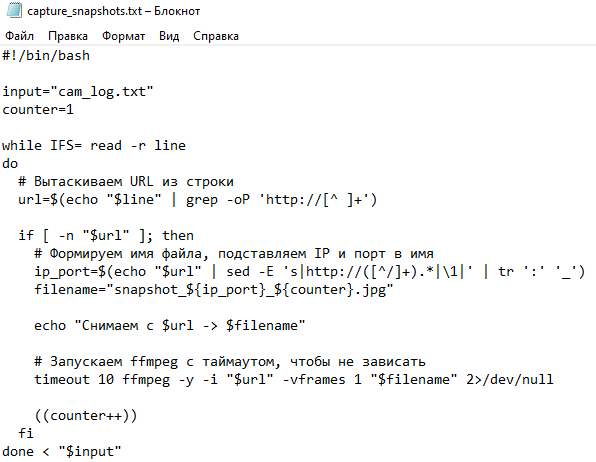
Добавлю файл с расширением sh в контейнер:



* check\_camera - функция, которая делает curl-запрос к каждому IP:PORT+URI и проверяет HTTP-статус.
* В цикле читаем nmap\_results.txt:
  + Если строка соответствует открытому порту, извлекаем порт.
  + IP ищем в предыдущей строке (Nmap так выводит).
  + Для каждого URI из файла вызываем check\_camera.
* Результаты пишутся в cam\_log.txt.

Делаем скрипт исполняемым: chmod +x scan\_cams.sh  
  
Запускаем: ./scan\_cams.sh  


Я решил автоматизировать перебор, для этого написал скрипт, который по списку потенциальных камер в cam\_log.txt будет запускать ffmpeg для каждой записи.



Сохранил его в формате sh, добавил в контейнер:

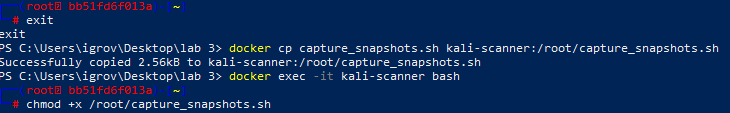
docker cp capture\_snapshots.sh kali-scanner:/root/capture\_snapshots.sh

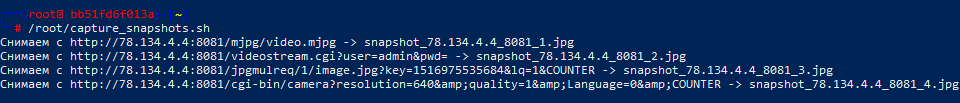
Сделал скрипт исполняемым:

chmod +x /root/capture\_snapshots.sh

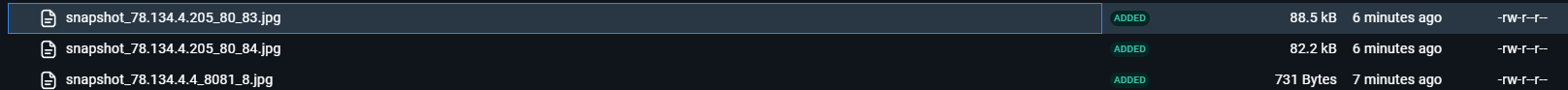
Запустил скрипт:

/root/capture\_snapshots.sh



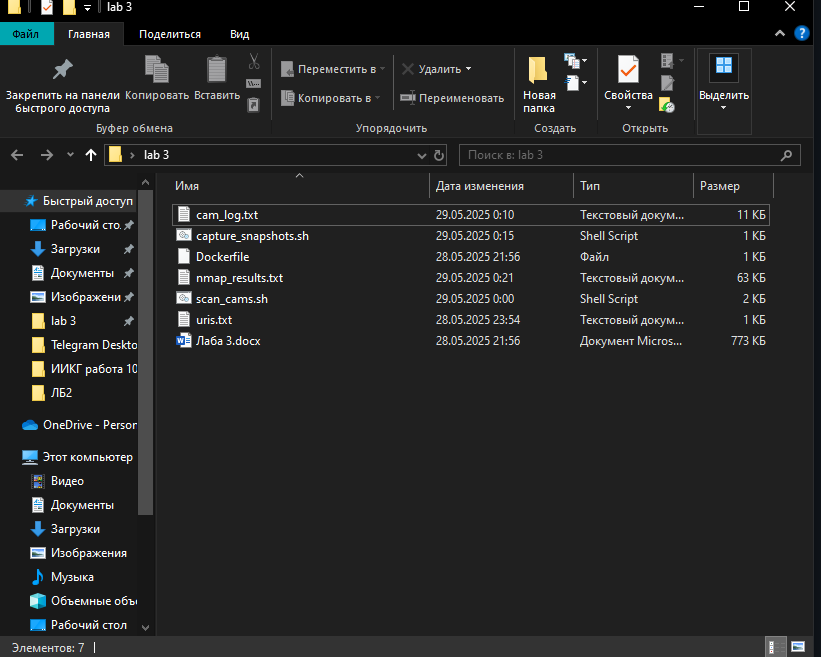


Так же тут видна небольшая часть логов, пока шел автоматизированный процесс

Всего получилось 3 изображения: 

Первые два одинаковые, третье изображение к сожалению не открывается

Тут как раз то что получилось на первых двух

Потом скопировал все в папку: 

Вывод и ответы на контрольные вопросы:

**I. Общие вопросы по установке и настройке**

**Dockerfile и Docker образ:**

* Dockerfile - это текстовый файл с инструкциями для автоматического создания Docker-образа. В нём прописывается, какую базу использовать, какие пакеты установить, какие файлы копировать и какие команды запускать.
* Преимущество Docker - быстрое и воспроизводимое развертывание среды, изоляция приложения, портативность и удобство масштабирования.
* docker build - команда для сборки образа из Dockerfile, docker run - для запуска контейнера на основе образа.
* Опции --tty и --interactive (-it) нужны для интерактивной работы с контейнером через терминал (чтобы можно было вводить команды).
* --cap-add=NET\_ADMIN даёт контейнеру дополнительные права на управление сетью (например, для запуска nmap, работы с сетевыми интерфейсами). Другие полезные capabilities: SYS\_ADMIN (администрирование), NET\_RAW (низкоуровневые сетевые операции).
* Проверить установку nmap, curl, ffmpeg можно командами nmap --version, curl --version, ffmpeg -version внутри контейнера.
* Для уменьшения размера образа можно использовать минимальные базовые образы (например, Alpine), объединять команды RUN, удалять кеши и временные файлы после установки.

**Настройка сети в Docker:**

* Docker создаёт виртуальные сети для контейнеров. По умолчанию используется bridge-сеть - контейнеры получают виртуальные интерфейсы, а хост пробрасывает порты.
* Другие режимы: host (контейнер использует сеть хоста напрямую), none (без сети), overlay (для мультихостовых сетей).
* Для отладки сети внутри контейнера можно использовать ping, traceroute, tcpdump, netstat.
* Проброс портов делается через -p <хост\_порт>:<контейнер\_порт>.
* Bridge-сеть изолирует контейнеры, а собственная сеть позволяет лучше управлять связью и настройками между контейнерами.

**II. Вопросы по сканированию сети с помощью Nmap**

**Nmap:**

* Nmap - инструмент для сканирования сети и обнаружения активных хостов, открытых портов и сервисов.
* Типы сканирования: TCP SYN (по умолчанию), TCP connect, UDP, ACK, FIN и др. В лабораторной использовался TCP SYN для скорости и скрытности. Полезны также UDP-сканы для поиска сервисов.
* Параметры -p 80,81,554,8080,8081,8888 - сканировать только указанные порты; -oN nmap\_results.txt - сохранить вывод в файл в нормальном формате.
* CIDR нотация 192.168.1.0/24 обозначает подсеть с 256 адресами (от 192.168.1.0 до 192.168.1.255). Nmap сканирует все IP из указанного диапазона.
* По результатам сканирования IP-камеры можно определить по открытым портам (80, 554, 8080 и др.), по заголовкам HTTP, по характерным сервисам и ответам.
* Для улучшения точности можно использовать скрипты NSE, проверять HTTP-заголовки, анализировать отпечатки сервисов.
* Этические соображения: сканирование чужих сетей без разрешения - нарушение закона и этики. Нужно иметь разрешение или работать в своей сети.

**III. Вопросы по Bash скрипту**

**Общие вопросы:**

* Скрипт перебирает IP и URI, посылает HTTP-запросы, проверяет ответы и пытается сделать скриншот.
* Переменные в Bash - контейнеры для хранения значений, используются для упрощения кода и повторного использования.
* Циклы while и for позволяют повторять действия по списку или пока условие истинно.
* Условные операторы if проверяют условия и выполняют код в зависимости от результата.
* Перенаправление > создаёт или перезаписывает файл, >> добавляет в конец, < читает из файла.
* chmod +x scan\_cams.sh делает скрипт исполняемым.

**Функции скрипта:**

* check\_camera() проверяет доступность видеопотока по URL, анализирует HTTP-ответ, пытается сделать скриншот.
* Можно улучшить проверку, анализируя Content-Type (например, multipart/x-mixed-replace для MJPEG), статус коды, время отклика.
* Ошибки обрабатываются через проверку кода возврата команд, можно добавить повторные попытки и логирование.

**Анализ HTTP ответов:**

* HTTP-ответ содержит статус (например, 200 OK - всё хорошо, 401 - нужна авторизация), заголовки и тело.
* Статусы 200 и 401 считаются признаками камеры, так как 200 - поток доступен, 401 - защищён, но камера есть.
* curl -m 1 -Is - -m 1 - таймаут 1 секунда, -I - только заголовки, -s - тихий режим. Помогает быстро проверить ответ сервера.

**Улучшения скрипта:**

* Добавить многопоточность (параллельные проверки).
* Добавить опции для настройки IP, портов и URI через параметры командной строки.
* Анализировать Content-Type, чтобы избежать ложных срабатываний.

**IV. Вопросы по захвату скриншотов ffmpeg**

**FFmpeg:**

* FFmpeg - мощный инструмент для работы с видео и аудио, в том числе захвата кадров из потоков.
* Команда ffmpeg -i "http://<IP>:<PORT>/<URI>" -vframes 1 snapshot.jpg захватывает один кадр из видеопотока и сохраняет в файл.

**Анализ видеопотока:**

* Проверить успешность можно по коду возврата ffmpeg и наличию файла.
* Часто IP-камеры используют MJPEG, H.264, RTSP.
* FFmpeg можно использовать для записи видео в файл, конвертации и трансляции.

**Выводы**

* Я научился автоматизировать поиск IP-камер в сети, проверять доступность потоков и делать скриншоты.
* Понял, что камеры часто плохо защищены, и их легко найти по открытым портам и URI.
* Можно улучшить скрипты, добавив авторизацию, логирование, многопоточность и гибкие настройки.
* Знания пригодятся для аудита безопасности, пентестов и мониторинга видеонаблюдения.