Разработка инструмента оценки безопасности приложений
и информационных систем

Подготовительные процедуры и руководство по эксплуатации

команда «5HM3L», разработка инструмента
оценки защищенности средств контейнеризации «Jamel»

Оглавление

1. О разработанном инструменте Jamel

В данном документе представлена информация о назначении, использовании, ключевых возможностях и составе инструмента Jamel.

Программа Jamel (далее – Инструмент) предназначена для решения задач по анализу средств контейнеризации, которые внедрены в сеть организации или которые планируются к внедрению. Инструмент предоставляет пользователю отчет с детальной информацией о существующих в используемых образах уязвимостях, а также выводит список с версионностью применяемых файлов для оптимизации процесса управления уязвимостями и оперативным принятием мер, в случае выявления новых угроз. Инструмент предназначен не только для администраторов безопасности в организации, но и сотрудникам, отвечающим за развертывание тестовых и продуктовых сред, разработчикам программного обеспечения и ответственным за организацию процесса vulnerability management.

Благодаря проработанной архитектуре Инструмента, встроенным автоматическим тестам, а также соблюдению правил «чистого кода», данный проект может быть легко горизонтально масштабирован, в него могут быть добавлены другие проверяющие модули, а также возможна интеграция с существующими системами.

При помощи разработанного Инструмента Вы можете:

1. Сканировать на уязвимости образы из Docker Hub;
2. Сканировать на уязвимости образы, сохраненные локально;
3. Сканировать код на уязвимые библиотеки;
4. Сохранять результат анализа или просмотреть его через утилиту.

2. Основные понятия

Инструмент реализован в виде клиент-серверной архитектуры. Для удобства тестирования обе части могут быть запущены на одном хосте. Также, архитектура проекта предусматривает возможность эксплуатации продукта в виде распределенной системы, когда на хосте тестировщика будет только клиентская часть (утилит командой строки для Linux, Windows или MacOS систем), а на отдельном хосте (физическое устройство/виртуальная машина/контейнер) будет запущена серверная часть.

Клиентское приложение – это утилита jamel-admin для подключения к серверной части, передачи данных для проверки и вывода результатов. Загружается из релиза продукта.

Серверная часть.

Состоит из нескольких составных частей. Запускается при помощи преднастроенных образов. Более подробно описано в разделе 7.

3. Технические требования

Для работы Инструмента предъявляются следующие минимальные требования:

Серверная часть.

Минимальные аппаратные требования:

- Процессор с частотой 1 ГГц или выше.
- Оперативная память: 4 ГБ.
- Объем свободного места на диске: 10 ГБ.

При выборе свободного места необходимо учесть объем проверяемых образов.

Программные требования:

- установленный git, docker и docker compose

Операционная система:

- Astra Linux 1.7/Ubuntu 22.04/Ubuntu 24.04/Debian 12 и другие ОС семейства Linux.

Клиентская часть.

Минимальные аппаратные требования:

- Процессор с частотой 1 ГГц или выше.
- Оперативная память: 2 ГБ.
- Объем свободного места на диске: 10 ГБ.

При выборе свободного места необходимо учесть объем проверяемых образов.

Операционная система:

- Astra Linux 1.7/Ubuntu 22.04/Ubuntu 24.04/Debian 12 и другие ОС семейства Linux.
- Windows (amd64).
- MacOS (m1 и выше).
- MacOS (Intel).

5. Запуск проекта

Для запуска данного проекта необходимо выполнить следующие действия.

1. Выполнить установку необходимых пакетов для подготовки окружения:

```
sudo apt update --yes
sudo apt-get install git docker.io docker-compose-v2
--yes
sudo usermod -aG docker $USER
```

Примечание: если docker-compose-v2 не найден - установите docker-compose и далее во всех вызовах docker compose используйте docker-compose.

2. Скопировать проект и перейти в директорию с ним:

```
git clone https://git.codenrock.com/sovkombank-
securehack-1331/cnrprod1733496609-team-
81653/jamel?roistat_visit=1998549
cd jamel/
```

3. Осуществить вход также для утилиты docker для доступа к хранилищу образов:

```
docker login https://git.codenrock.com:5050
```

4. Запустить проект:

```
docker compose up -d
```

Примечание: если планируется использование ОС Astra Linux 1.7, то благодаря новому механизму проверки контейнеров, бывают ошибки вида:

ERROR: failed to register layer: directory '/var/lib/docker/overlay2/e2b2b6adc504672ef5a45a9bf9b0043f41e4ab1d1f52d05e54e35aa32df20130/diff' contains vulnerabilities! [{oval:astra:def:437401477323443746914756390662 7 true Astra Linux - уязвимость в db5.3 }]. Image: e2b2b6adc504672ef5a45a9bf9b0043f41e4ab1d1f52d05e54e35aa32df20130

Это означает, что встроенный начиная с версии 1.7.4 сканер уязвимостей Openscap, проверяющий уязвимости по базе oval-db, выявил угрозу в загружаемом образе. Для отключения данной проверки можно настроить исключения или отключить проверку, создав файл ***/etc/docker/daemon.json*** *со следующим содержимым:*

```
{ "astra-sec-level" : 6 }
```

После этого перезагрузить docker.

Также, после запуска системы происходит процесс обновления баз. Работа с системой возможна только после успешного обновления баз CVE. Сообщение об успешной загрузке баз - ***updated finished.***

Проверит статус работы/запуска можно при помощи команды:

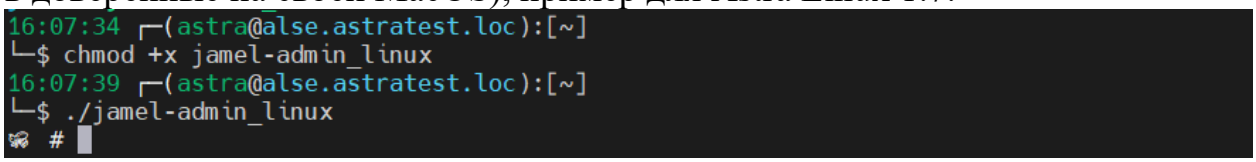
```
docker logs client

Пример успешного вывода:
2024/12/14 16:40:37 loop started
2024/12/14 16:40:37 start update task
2024/12/14 16:45:39 updated finished
```

5. Далее, как проект будет запущен, а базы обновлены, можно приступать к работе по функциональному предназначению. Загрузите клиентское приложение, размещенное на станице с релизами. На указанном ресурсе, размещены файлы для различных ОС, а именно:

- **jamel-admin_linux** – для Linux (amd64).
- **jamel-admin_windows.exe** – для Windows (amd64).
- **jamel-admin_darwin_arm** – для MacOS (m1 и выше).
- **jamel-admin_darwin_amd64** – для MacOS (Intel).

6. Выполните необходимые процедуры и запустите (сделайте файл управляемым командой **chmod +x jamel-admin_linux** или добавьте файл в доверенные на своей MacOS), пример для Astra Linux 1.7:



Взаимодействие клиентской части приложения (управляющего файла) с сервером осуществляется по сети. Стандартно производится подключение на 127.0.0.1:8443, однако возможно подключение к удаленно развернутому серверу или подключение из другой сети. Для этого необходимо указать адрес сервера в переменной окружения в формате SERVER=ip:port, например:

```
SERVER=192.168.10.10:8443 ./jamel-admin_linux

Или

export SERVER=192.168.10.10:8443
./jamel-admin_linux
```

6. Руководство пользователя

После описанных в предыдущих разделах действий имеется возможность приступать к работе с разработанным Инструментом.

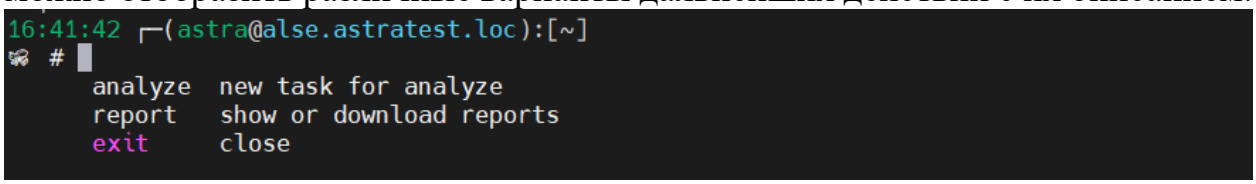
Как описывалось выше, запуск клиентской части инструмента возможен для пользователей, обладающих различными ролями в организации, а также с устройств, под управлением различных ОС.

Для осуществления работы необходимо:

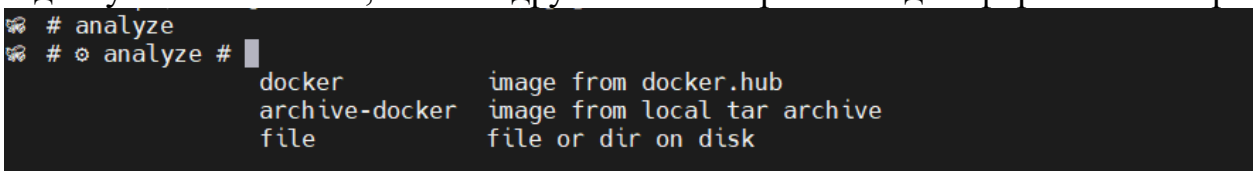
1. Запустить клиентское приложение, выполнив команду (пример для ОС Linux, когда и серверная и клиентские части расположены на одном хосте):

```
./jamel-admin_linux
```

2. После этого в интерфейсе командной строки при помощи табуляции можно отобразить различные варианты дальнейших действий с их описанием:



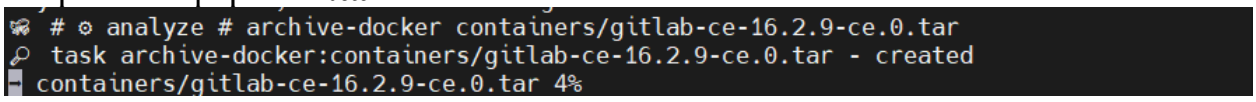
3. Введя команду analyze Вы будете перемещены в следующее меню. На выбор можно проанализировать образ из Docker Hub, загруженный и доступный локально, а также другой любой архив с кодом формата .tar/.zip.



4. Примеры для каждой команды. Запустим на проверку образ **ubuntu:16.04:**

Выведено сообщение о создании задачи проверки. Продолжим тесты.

5. Создадим задание на проверку локального файла с образом, который сохранен в формате **.tar**



Утилита выводит статус загрузки этого образа для анализа. Продолжим тесты.

6. Третий вариант проверки, опция «**file**». Необходима для проверки кода программ на наличие ссылок на вредоносные библиотеки. Создадим задание на проверку **.tar** архива с кодом тестового проекта.

```
# ⚙ analyze # file file/python3_script.tar
task file:file/python3_script.tar - created

 Id                                   Created      Filename                Type
 29783fc7-d56d-4f92-a649-03be8a6f4697 Dec 15 17:51 file/python3_script.tar FILE
```

7. Работа с результатами анализа.

После выполнения анализа загруженных данных, имеется возможность ознакомиться с результатами. Для этого необходимо при помощи exit вернуться в главное меню и перейти в подраздел report.

```
# ⚙ analyze # exit
# report
# 📋 report #
                list  show all results
                show  show report for task
                json  download report for task in json
```

8. Команда **list** покажет все отчеты, команда **show** предоставит возможность просмотреть конкретный отчет (табуляция также поддерживается). Также для удобства работы имеется возможность выгрузки результатов анализа в формате **json**, используя одноименную команду, например:

```
# 📋 report # show 1


 Id                                   Created      Filename     Type

 532327c2-1107-4b3d-9a5a-a471d20f4c24 Dec 15 17:26 ubuntu:16.04 DOCKER



 NAME              INSTALLED                    FIXED-IN     TYPE  VULNERABILITY   SE
VERITY
 bash              4.3-14ubuntu1.4              (won't fix)  deb   CVE-2022-3715   Me
dium
 bash              4.3-14ubuntu1.4                           deb   CVE-2019-18276  Lo
w
 bsdutils          1:2.27.1-6ubuntu3.10         (won't fix)  deb   CVE-2021-37600  Lo
w
 bsdutils          1:2.27.1-6ubuntu3.10         (won't fix)  deb   CVE-2020-21583  Lo
w
 bsdutils          1:2.27.1-6ubuntu3.10                      deb   CVE-2016-5011   Lo
w
 bsdutils          1:2.27.1-6ubuntu3.10                      deb   CVE-2016-2779   Lo
w
 coreutils         8.25-2ubuntu3~16.04                       deb   CVE-2017-18018  Lo
w
 coreutils         8.25-2ubuntu3~16.04                       deb   CVE-2016-2781   Lo
w
 dpkg              1.18.4ubuntu1.7                           deb   CVE-2017-8283   Ne
gligible
```

7. Руководство администратора

Общая структура проекта:

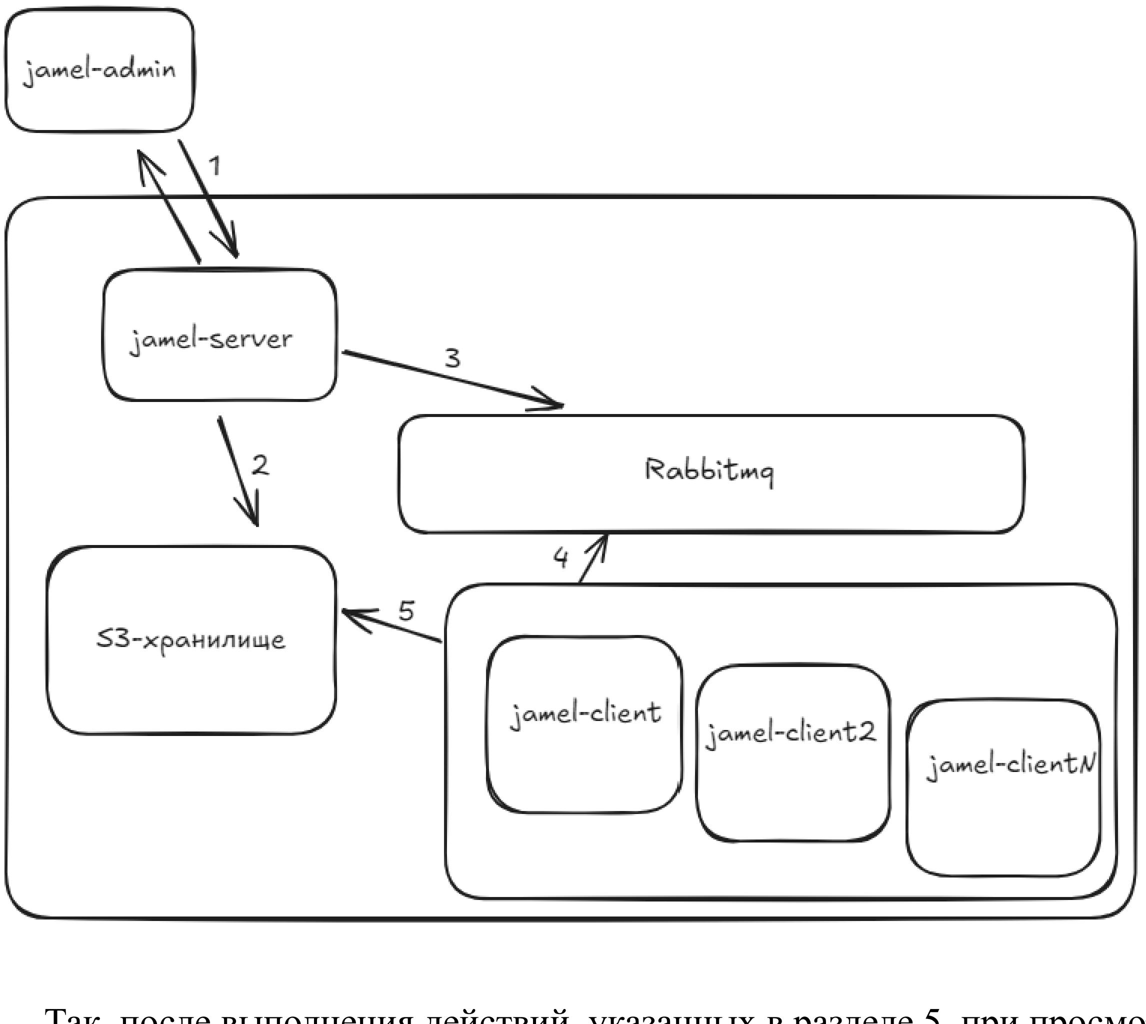
jamel-admin (процесс 1 - передает данные для проверки и выводит результат);

jamel-server (ожидает подключения с приложений-клиентов из раздела 6 данной инструкции и передает полученные файлы на проверку);

minio (процесс 2 - сохранение в S3-хранилище файлов);

rabbitmq (процесс 3 - брокер сообщений, который получает от **jamel-server** задание на проверку со ссылкой на файлы в **minio**);

jamel-client (процесс 4 и 5 - получает от **rabbitmq** задания на проверку, а от **minio** файлы и возвращает результат).



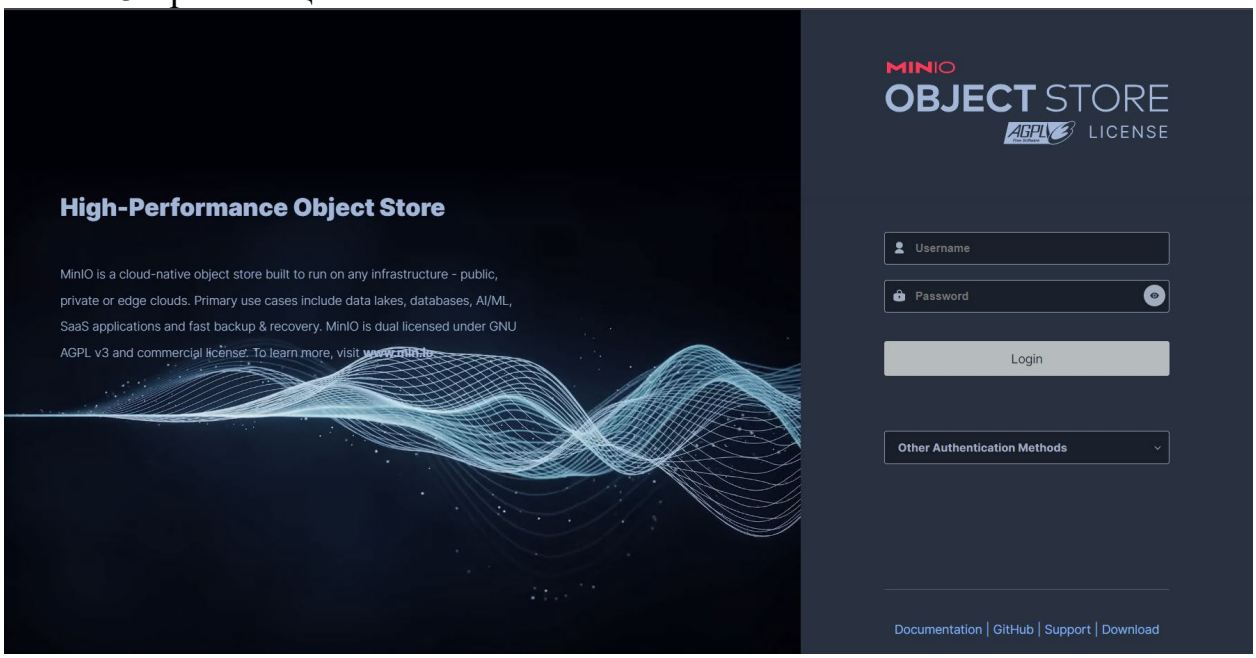
Так, после выполнения действий, указанных в разделе 5, при просмотре на устройстве, где была развернута серверная часть, запущенных образов, вывод команды **docker ps** должен сообщить о 4 запущенных контейнерах:

```
└─$ docker ps
CONTAINER ID   IMAGE                                                                                  COMMAND                  CREATED       STATUS
  PORTS                                                                                                                               NAMES
f0b4b37ae041   git.codenrock.com:5050/sovkombank-securehack-1331/cnrprod1733496609-team-81653/jamel/jamel-server:latest     "./jamel-server"         3 hours ago   Up 3 hours
  0.0.0.0:8443->8443/tcp, :::8443->8443/tcp                                                                                             server
bf2bc87464aa   git.codenrock.com:5050/sovkombank-securehack-1331/cnrprod1733496609-team-81653/jamel/jamel-client:latest     "./jamel-client"         3 hours ago   Up 3 hours
                                                                                                                                      client
136363b98bb0   git.codenrock.com:5050/sovkombank-securehack-1331/cnrprod1733496609-team-81653/jamel/minio:latest            "/usr/bin/docker-ent…"   3 hours ago   Up 3 hours (healthy)
  0.0.0.0:9000-9001->9000-9001/tcp, :::9000-9001->9000-9001/tcp                                                                         minio
af34c77de355   git.codenrock.com:5050/sovkombank-securehack-1331/cnrprod1733496609-team-81653/jamel/rabbitmq:3-management   "docker-entrypoint.s…"   3 hours ago   Up 3 hours (healthy)
  4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp, :::15672->15672/tcp   rabbitmq
```

Для контроля функционирования можно использовать различные варианты, например:

1. Просмотр логов контейнеризации при помощи команды **`docker logs client;`**
2. Подключение к контейнеру для выявления внутренних ошибок, перезапуска служб и пр. при помощи **docker run** или **docker exec;**
3. Изучение данных для подключения к сервисам (порты, адреса и учетные данные в файле **.server.sample.env** в директории с проектом) и авторизация в данных сервисах непосредственно для анализа.

S3-хранилище:



Брокер сообщений: