**Name:** Nathan Abdelmalek

Email: nca9067@rit.edu

**Team:** Bravo

**Introduction:** This tool is a compilation of several different things that may be useful during this competition. At the most basic level, this ansible script first builds an infrastructure (which would not be used during the competition, but helped with development), then installs vulnerable services and opens a webshell as well as a reverse shell. Finally, the ansible program creates a series of bash scripts designed to annoy the blue team and schedules them to go off during competition hours. The final script, if it is not stopped by the last 15 minutes of the competition, will systematically delete everything on the drives.

**Instructions for usage/testing:**

1. Open three terminal windows on the development/deployment box. This box has a static IP address of 100.65.3.230.
2. In one terminal window, navigate to the /home/ubuntu/tool directory. This is the directory that contains the deployment script.
3. If the infrastructure (UbuntuWebserver and UbuntuAttacker) has not been created, run the ansible file with the command **ansible-playbook -i inventory.ini attack.yml --tags build.** When you are grading this assignment, the infrastructure should already be in place up to this point.
4. Wait at least 1 minute for the systems to fully load.
5. In a separate terminal window, enter the command **ssh 192.168.0.20**. This box represents an attacker. You may have to remove an old entry from the hosts file.
6. In that same terminal window, enter the command **nc -l -p 1337** to open a listening port for the incoming reverse shell connection.
7. Return to the oriinal terminal session on the development box. Run the **command ansible-playbook -i inventory.ini attack.yml --tags modify.** This will install Apache and PHP on the target, add a malicious PHP file as a backup, create a reverse shell connection back to the listener, add www-data to the sudoers file, and create a python program on the development machine that allows you to easily interact with the webshell. You may have to remove entries from the hosts file and/or accept new fingerprints as necessary.
8. Wait for the script to execute. When the script hangs on the "execute python file" task, you will see that a root shell has been opened with the netcat listener in the other terminal window.
9. Explore as you wish, then exit the netcat session. The ansible script will end.
10. In the third terminal window, connect to the attacker box and start a netcat listener as described in steps 5 and 6.
11. In the terminal window connected to the attacker box, you will find a file in the /home/ubuntu directory called "webshell_requests.py". This simple python program takes user input, URL encodes it, and sends it to the malicious PHP file installed on the webserver. You can enter any command, so long as it does not involve redirects(>) or pipes (|). It is generally easier to re-open the netcat session.
12. Run the program with **python3 webshell_requests.py**. When prompted, enter the command **python3 /usr/share/nano/...** to run the implanted python backdoor hidden on the webserver. A reverse shell will once again be opened in the netcat session.

13. Exit out of any running sessions.
14. On the development box, enter the command **ansible-playbook -i inventory.ini attack.yml --tags funny**
15. Using any terminal window, enter the command **ssh 192.168.0.30**. This will connect you with the webserver target, where we will manually check the results of the script.
16. On the target, navigate to the /etc/haha directory. List the files here. Each file tells exactly one joke to the users logged into the system, then deletes itself.
17. Execute any file you wish using the command **sudo ./<file>.** **DO NOT EXECUTE HAHA12 – IT WILL DELETE ALL DATA FROM THE MACHINE.**
18. Run the command **sudo at -l**. This lists a series of "at" jobs that are scheduled to run every 15 minutes during the competition. The jokes are told in order, continually reminding the user that they should locate the source of the jokes and delete the files. While this is very funny, it is not practical outside of this competition. The purpose of warning the user is to give them every chance to remove the files before haha12 is deleted. These files are not difficult to detect, nor are they intended to be. This is to give the blue team something to keep them occupied.

**Lab Questions**

- What is the goal of this tool? What purpose does it bring to the competitions?

There are several different goals of this tool. Firstly, it is to give the red team persistent access into the blue team's network. Having both the malicious PHP file/webshell and the reverse shell python script provides redundancy – if either one is shut down it can be restored via the other method of access. Additionally, it gives the red team a solid foothold to be able to perform further attacks/date exfiltration. Additionally, this tool is designed to give the blue team practice in managing scheduled jobs, albeit in a rather comical way. This tool provides ample warning of what will happen if it is not removed, and it should encourage the blue team to try to find these hidden files. Not only does this provide them with practice, but it also diverts their attention, allowing the red team to perform additional actions while their focus is directed elsewhere.

- Did other tools influence your tool? If so, what are they? If not, what was your inspiration for the tool?

Yes, several other tools provided inspiration for my tool. In fact, many other components were pulled from other sources (listed below), and I strung them together with my own unique twist (the jokes). I am also solely responsible for creating the ansible to unify this attack so it can be efficiently deployed on real life targets. The inspiration for the jokes came from the movie "Spaceballs", of all places. As you likely already know, there is a scene in this film where the computer makes wisecracks as it counts down to the self-destruction of their own ship. In reality, this is almost exactly what this program does.

- What is the feasibility of another team member quickly learning to use or contribute to your tool? What makes it easy or difficult to learn?

I would say that this tool would be fairly easy to learn or contribute to. Most of this project is in either Ansible or Python, both extremely easy formats to read and understand. I have provided ample documentation for usage (see above) so any team member should be able to follow these steps to deploy an attack. Additionally, there is a lot of room for improvement, so it is certainly possible that others could easily make meaningful additions to the project.

## Sources

Malicious PHP Script – https://www.grobinson.me/single-line-php-script-to-gain-shell/

Malicious BASH Script – https://github.com/greyhat-academy/malbash/blob/main/wiper.sh

Ansible Documentation - https://docs.ansible.com/