3 Seas Documentation

Team Members: Rebecca Hassett, Lise Ho, Andrew Jaffie

Project: Covert Command and Control Data Exfiltration Project

Required Languages:

Python 3.7

Victim Target Operating System:

Windows 10

Attacker Target Operating System:

Windows 10 and Ubuntu 18.04

3Seas Capabilities:

The covert C&C server application includes a victim application and attacker application both developed in Python. The victim python files are generated into an executable using pyinstaller with the installer.bat file. The victim executable communicates to the attacker application using Google Drive as our unsuspicious middleman. Whenever a machine starts running the victim application, it checks whether or not this machine is a new victim or not. If the machine is a new victim on Google Drive, then the folder ID of the new victim on Google Drive is stored in the Windows Registry as a key. The folder ID is stored in Software\SystemFiles as a key called "Config" which an actual victim would not suspect to look at nor want to change in fear of breaking their system. If the folderID is already stored at this location, then the victim can start polling the drive looking at files only in their folder.

When a new victim is generated in Google Drive, they receive a new folder with a uniquely generated ID that contains two subfolders called “ToAttacker” and “FromAttacker”. The victim continuously polls the FromAttacker folder in their folder to look for new files from the attacker with commands. The victim will poll Google Drive every 10 seconds. If the victim cannot find its folder when polling Google Drive, it generates a batch file that removes the victim executable as well as the run registry key and the victim ID registry key. This allows the attacker to delete a victim, and then all traces of the victim application to disappear on the victim machine.

Another functionality that the victim application has is that it creates a run registry key in HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run. The run registry key’s name is “AntivirusTools” and the value is the path to the victim executable. This is for running the victim application every time the victim’s machine is restarted.

The attacker application sends commands to the victim through Google Drive. This can be done in “Interactive Mode” where the attacker has a UI that alerts the attacker in real-time when new victim has been added to Google Drive or a current victim has uploaded a response to a command in Google Drive. This utilizes Google Drive push notifications for alerting the attacker of changes in Google Drive. The top screen of the UI in Interactive Mode also saves a list of commands that have been sent to the victim. The commands and their descriptions are listed in the section below.

To run interactive mode, in the command line run “**python attack.py -l -i**”

<u>Attacker Commands:</u>

get_machine_info [victims] or **minfo [victims]** Returns information about victim's machine such as operating system and processor of machine

get_file [path filename] [victims] or **--gf [path filename] [victims]** Retrieves file from victim's machine

send_file [filename] [victims] or **--sf [filename] [victims]** Sends file to victim's machine. In our version of our application, we support file uploads up to ~10 GB - this is due to the fact Google Drive can allow files of max 10GB size. We have a partially implemented multi-part uploading and sending of files written that would have increased the max cap of files transmitted to > 10GB however we opted not to submit it due to insufficient time to test.

execute_command [cmd] [victims] or **exec [cmd] [victims]** Executes command on victim's machine. For example, execute_command ipconfig will get network configuration information from victim's machine. Another example would be execute_command dir C:\ to list files located in the victim's root directory.

execute_file [filename] [cmd] [victims] or **execf [filename] [cmd] [victims]** Allows attacker to send file to victim and execute the file using the specified command. For example, execute_file app.py "python app.py" can be used to execute app.py on the victim's machine.

download_files [victims] or **download [victims] -** Download all files victim has uploaded to their ToAttacker folder

delete [victims] or **--d [victims]** Delete Victim Folder [includes warning if attacker tries deleting all victims]

** To send a command to a specific victim, add the victim's name at the end of the command such as "send_file foo.png Becca"

** To send a command to multiple victims, add all of the victim's names space separated at the end of the command such as "send_file foo.png Becca Andrew Lise"

** To send a command to all victims, do not add any victim names after the command such as "send_file foo.png"

rename_victim [new name] [current name] or **rename [new name] [current name]**

Renames specified victim folder's name

**Victim's current name must be listed next to command such as "rename_victim Becca 1234". Only one victim can be renamed to the specified name.

list_victims or **list** or **--lv** Prints a list of current victims in Google Drive

listen or **--l** Toggles server on and off for listening for push notifications

--interactive or **-i** Enables interactive mode window

exit Exits interactive mode and closes push notifications channel

help Prints list of commands

<u>Push Notifications Server Setup:</u>

In order to receive push notifications, there are a few steps you must follow. First, you must have a domain name that you control registered in the Google search console. Then, go to the Google Developer console → Google Drive API → Credentials → Domain Verification and add your domain. This may take a few minutes to take effect.

Next, you will need to set up a reverse proxy. The minimal setup would be a container with NGINX and an SSH server. NGINX should be configured to proxy_pass from https to 127.0.0.1:8080. Google requires you to get a *valid* SSL certificate, which I recommend using the certbot utility for since you can get a free cert from Let's Encrypt. From there, you have two choices: either you can run the attacker program on the container/VM, or you can run it locally and use an SSH tunnel that forwards requests from the container/VM to your machine. This way, you can easily use the interactive mode without needing to worry about getting the program onto the VM or x-server forwarding. An SSH tunnel is definitely the recommended method and forwarding the UI using x-server forwarding is untested.

Finally, put the URL you want the drive API to hit into the file config.py. Now all you need to do is run `python attack.py --listen --interactive`. Also, `listen` inside interactive mode toggles the notification server on and off. It is ***strongly*** recommended you use the `exit` command to quit while the server is running, as ungraceful exits can result in duplicate notifications or at least extra traffic over the next few hours.

Steganography Portion [Not Submitted Due to Transient Errors]:

Steganography is implemented in this project using the Stegano 0.9.3 library. When the victim receives a command from the server, the victim hides its message to the user in an image. The image inside victim must have the name “template.jpg”. The message is hidden in a the LSB of a set of pixels of the image. The set of pixels used for encoding the image is chosen using an Eratosthenes generator. The name that the image is set to is “picture” + number where the number is mapped to the type of command that the victim is responding to. For example, “picture1.jpg” maps to a “machine_info.txt”. The attacker receives the image from the victim, and then the attacker obtains the message that is hidden in the picture’s images and prints the image to a file whose name indicates the type of command that the victim is responding to. This feature was not submitted due to transient errors. We did not have sufficient time to determine why the steganography module would crash during random runs.

Project Setup:

Download 3-Seas Covert C&C from github repository. The requirements can be installed by running “pip install -r requirements.txt”.

Creating Google Drive Service Account:

1. Using a gmail account, navigate to the link: Service Account Link
    a. This link contains detailed instructions on setting up a Google Drive Service Account
2. Select the Google Cloud Platform (GCP) Console link
    a. Agree to the terms of service

b. Click “Select Project”, “New Project”, enter Project Name, and “Create”
c. Select “Enable APIs & Services” at the top of the page
d. Search for “Google Drive API”, select it, and click “Enable”
e. Select the three bar menu in the upper left hand corner, select IAM&Admin, and select “Service Accounts”
f. Select “Create Service Account”, enter a Service account name, and select “Create”
g. Select “Continue”
h. Select “Create Key” and “Create” with JSON selected. The JSON file will be downloaded automatically.

<u>Victim Setup:</u>

Before converting the victim files into an executable to be maliciously placed on the victim’s machine, the attacker must create a Google Drive service account and retrieve the json credentials for this account. Once the JSON credentials file is retrieved, rename it to “creds.json”. Move the creds.json file inside the Victim folder. Inside the Victim folder, run “installer.bat” from the command line. The batch file uses pyinstaller to package the victim python files into an executable file. The executable file will be inside the generated dist folder inside the Victim folder. The generated executable file is called “victim.exe”. This executable can now be maliciously placed on the victim’s machine to wreak havoc.

<u>Attacker Setup:</u>

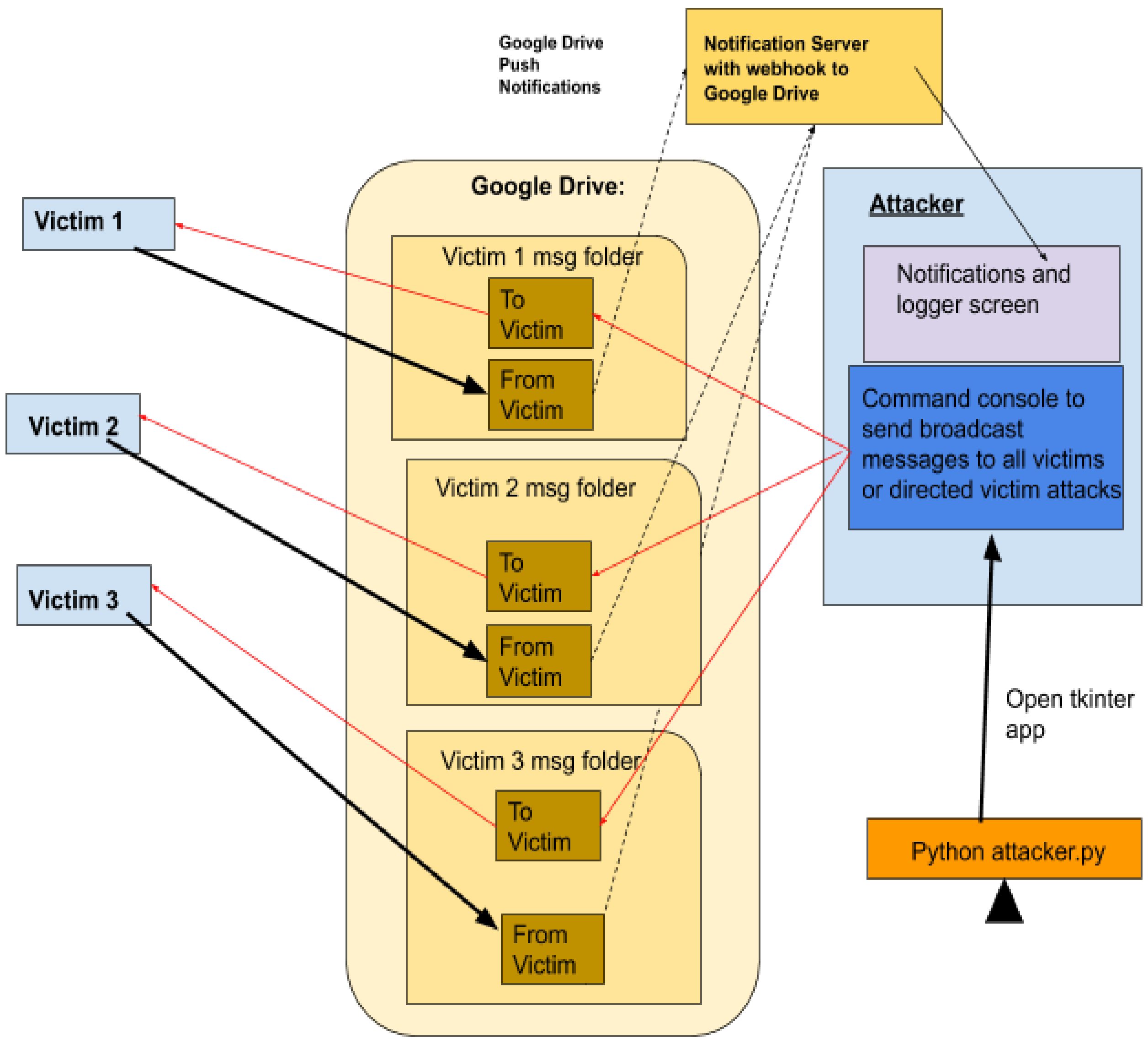
The Attacker setup is slightly different from the victim setup. The path to the service account/JSON credentials file is set in the config.py file. The same Google Drive service account is needed as with the victim, and with the same JSON credentials files. We recommend copying the creds.json file to the Attacker directory for easy path access. Then, you MUST set the SERVICE_ACCOUNT_FILE to the path to this JSON credentials file. You can also change the notification server url `NOTIF_SERVER_URL` if you wish to use another server for the notifications in this config url.

Last but not least, make sure to run the attacker.py file either in interactive mode or not so that you can launch your commands to attack the victims!

Logging:

We have implemented logging on the Attacker side. We opted to not have logging on the victim side as having more data than necessary on the victim side would only lead to a higher likelihood of the victim being noticed. Once the Attacker application is started, a logger object is started and info and error logs are logged on the top window of the UI. In addition, these logs are set to also be written into log files saved locally on the attackers computer under the logs directory in the Attacker directory. In order to receive messages from the notification server which runs in another process, a process- and thread-safe queue object is passed through when it is created and a separate thread in the main program is tasked with retrieving messages from the queue and forwarding it to the rest of the logging module.

Application Architecture Diagram:


Google Drive Push Notifications
Notification Server with webhook to Google Drive
Google Drive:
Victim 1 msg folder
To Victim
From Victim
Victim 2 msg folder
To Victim
From Victim
Victim 3 msg folder
To Victim
From Victim
Victim 1
Victim 2
Victim 3
Attacker
Notifications and logger screen
Command console to send broadcast messages to all victims or directed victim attacks
Open tkinter app
Python attacker.py