CS501 Final Project: C2 Engineering Design Documentation

Team member: Jianqi Ma, Ning Wang, Wei-Heng (Wayne) Lin, Yuhao He, Xinyu Liu

Installation

After clone our project from GitHub, do the following to run the implant

- To compile, run `make implant` in the implant folder
- Double click `implant.exe` to run the implant in default mode and connect to remote server
- For development/testing purposes, you can run `implant.exe -local` to connect to your local server and view implant behavior in the console window.

For C2 server:

- We have deployed our server on Heroku, you can use this link to access our server
- Or, if you want you could run our server locally as well by following the steps below (Make sure you have python3 installed)
    i. cd into the C2 folder
    ii. Create a virtual environment with `python3 -m venv c2_env`
    iii. Activate the virtual environment with `source c2_env/bin/activate`, or `c2_env/Scripts/activate` if you are using powershell
    iv. You can exit the virtual environment with `deactivate`
    v. Install the dependencies with `pip3 install -r requirements.txt`
    vi. Create a new database called `c2_server` in **mysql** by using `CREATE DATABASE c2_server`
    vii. Create a new file in the C2 folder named `creds.py` with the line `db_cred = {'username'='', 'password', ''}`, and fill in your username and password for your mysql server
    viii. Create a database with `python3 make_db.py`
    ix. Run the server with `flask run`

Capabilities

- Powershell code execution: By selecting the **powershell code** command type on the client side enter arbitrary powershell code from client side and execute the given code in victim machine
- Steal: By selecting the **steal** command type on the client side, the implant will steal the chrome passwords stored on victim machine and return the results to the client
- shellcode execution: By selecting the **shellcode** command type on the client side, the implant will fetch shellcode created by the server from a precompiled executable file, and inject the shellcode into a remote process and execute it.
- change_config: By selecting the **change_config** command type and enter a numerical value **N** on the client side, the sleep time of implant will be set to **N** seconds
- cryptography: AESGCM encrypted communication between the server and the implant
- secure connection: HTTPs
- User Friendly Interface

Operational Security

For shellcode injection, we choose to inject to a remote process instead of the current process. Doing so can be a red flag to certain virus detection mechanisms, since if we do so, we would need to have the ability to spawn a new process inside the current process. However, our implant has other functionalities, like running arbitrary powershell commands, that requires the ability to spawn new processes. Thus, injecting into a remote process won't actually increase the suspicion of our implant. Moreover, Since we are injecting into a remote process, it is easier to handle unexpected errors in the injected shellcode.

Our implant supports sleep time modification on the fly. By default, our implant would constantly try to talk to the c2 server in order to inform the c2 server that the implant is currently running. However, doing so can produce suspicious network traffic between the c2 server and the victim machine. In order to mitigate this effect, we implemented the **change config** command to the c2 server, so that the adversary can actively change the sleep time for the implant.

The implant and c2 server talks with symmetric encryption on top of HTTPS. Doing so can prevent some MiTM attacks performed by injecting some malicious CA into the victim machine. However, currently the symmetric key is hardcoded into the implant. If someone finds that key, all of our implant's traffic can be intercepted and modified freely. For future works, we can try to implement some key exchange algorithm like the Diffie Hellman Key Exchange to reach a consensus before the real communication between the implant and the c2 server happens.

Our implant persists itself by adding itself in the user registry run keys. This way the implant will start up automatically when the computer boots. However, this behavior can set off windows defenders, which is not very practical in the real world. To achieve more covert persistence, we need to consider other techniques such as the ICQ entry.

Utility

Our implant can be a very effective loader. With some simple modifications on the c2 server side, we can have multiple shellcodes which effectively act as modularized functionalities of a malware. Once the implant gets initial access to the victim, the adversary can decide what they want for this victim machine, and then send corresponding commands to the implant. Implant will then dynamically load the functionalities it needs to perform that task from the c2 server, and inject the shellcodes into some processes and return the results back to the c2 server.

Collaboration Breakdown

- Implant: **Jianqi Ma, Ning Wang, Xinyu Liu**
- Stealer: **Xinyu Liu**
- C2 Server: **Wei-Heng (Wayne) Lin, Yuhao He**
- Client: **Wei-Heng (Wayne) Lin**
- Special feature - Client UI: **Jianqi Ma**