Armox Botnet

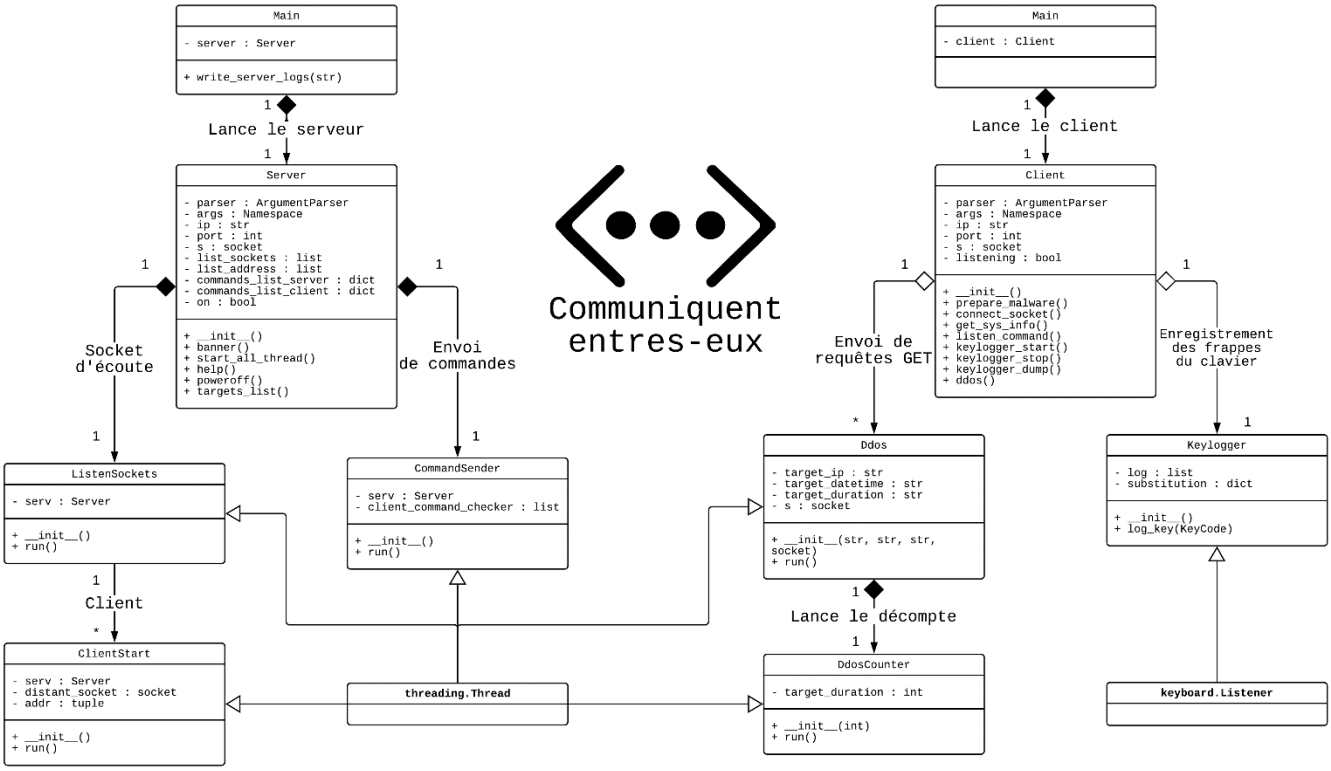
Projet : Développement Python

Laenen Maximilien
Michaux Clément

23/12/2019

Table des matières

I. UML



II. Bibliothèques externes utilisées

Pynput :

- Pynput permet de contrôler et écouter les entrées d'un ordinateur comme la souris et le clavier.

Dans le cadre de notre projet, celui-ci fût utilisé afin d'enregistrer les frappes de clavier des machines infectées de notre botnet.

Requests :

- Requests permet d'envoyer des requêtes HTTP de manière simplifiées.

Ici, nous n'utilisons que la requête HTTP GET car nous souhaitons faire du déni de service. Mais nous pourrions très bien utiliser d'autres requêtes comme « PUT / HEAD / … »
« La méthode HTTP GET demande une représentation de la ressource spécifiée. Les requêtes GET doivent uniquement être utilisées afin de récupérer des données. »

III. Bibliothèques internes utilisées

Socket :

- Cette bibliothèque permet de faire communiquer deux machines ou plus entre elles. Une machine va écouter les demandes de connexion sur une de ses adresses IP et un de ses ports (appelé serveur) et accepter les machines se connectant à celui-ci.
  La machine cliente, elle, va essayer de se connecter au serveur sur l'IP et le port sur lesquelles le serveur écoute. Une fois que le serveur et le/les client/s sont interconnectés, ils peuvent s'envoyer et recevoir des messages/commandes que la connexion soit faite en half-duplex ou en full-duplex.

Notre projet intègre les sockets afin de démarrer un serveur qui va écouter les demandes de connexion de toutes les machines infectées. Par la suite, on peut envoyer des commandes aux « robots » afin de démarrer différentes actions sur le client comme un keylogger/un ddos/…

Threading:

- Ce module, lui aussi, intégré permet de démarrer de nouveaux « threads » qui vont fonctionner en parallèle de votre programme principal. Ceci est très pratique, car il peut faire tourner plusieurs fonctions par exemple les unes à côté des autres.

La principale utilité de ce module dans notre projet est de pouvoir lancer un keylogger et en même temps un ddos. Ce qui est très utile, cela ne nous oblige pas à arrêter le premier pour pouvoir lancer le second.

Ctypes :

- Ctypes est une library interne qui permet de faire appel à des fonctions présentes dans une bibliothèque compilée. Cela nous permet de faire appel à des .DLL (Windows) ou des .SO (Linux)

Nous utilisons très peu celle-ci seulement pour cacher la console lors du lancement du client afin que la victime n'ait aucunement l'idée qu'elle ait été infectée. Et une seconde fois pour afficher un message d'erreur lors du lancement du programme afin de rendre le programme ressemblant à plein d'autres.

« Une Dynamic Link Library (en français, bibliothèque de liens dynamiques) est une bibliothèque logicielle dont les fonctions sont chargées en mémoire par un programme, au besoin, lors de son exécution, par opposition aux bibliothèques logicielles statiques ou partagées dont les fonctions sont chargées en mémoire avant le début de l'exécution du programme. »

« Les DLL sont une des fondations des systèmes d'exploitation Windows. Les fonctions internes de Windows sont mises à disposition des programmes par des interfaces de programmation mises en œuvre par des fichiers DLL10. Les trois principales bibliothèques sont User32.dll (manipulation de l'interface utilisateur), GDI32.dll (manipulation des dispositifs d'impression et d'affichage), et Kernel32.dll (utilisation des fonctions du noyau de Windows concernant les fichiers et les processus). »

Datetime :

- Ce module fournit des classes qui vous permettent de manipuler les dates/le temps.

Dans le cadre de notre projet, la seule fonction que nous utilisons est datetime.today() afin de récupérer la date précise du moment ou cette fonction est appelée. Nous utilisons ensuite celle-ci afin de planifier ou non notre DDOS.

Os :

- La bibliothèque « os » nous permet de manipuler des fonctionnalités Windows, de jouer avec les chemins, etc.

Quelques fonctions utilisées sont par exemple, os.walk() qui nous permet de nous déplacer dans l'arborescence de fichier, os.path.basename(__file__) renvois le nom de fichier du programme lancé… Principalement utilisé pour compléter la copie du malware dans un autre dossier.

Random:

- Permets de générer des nombres pseudos-randoms.

Dans notre projet, ceci nous permet de choisir une bannière de manière aléatoire.

« Un générateur de nombres pseudo-aléatoires, pseudorandom number generator (PRNG) en anglais, est un algorithme qui génère une séquence de nombres présentant certaines propriétés du hasard. Par exemple, les nombres sont supposés être suffisamment indépendants les uns des autres, et il est potentiellement difficile de repérer des groupes de nombres qui suivent une certaine règle (comportements de groupe).
Cependant, les sorties d'un tel générateur ne sont pas entièrement aléatoires ; elles s'approchent seulement des propriétés idéales des sources complètement aléatoires, comme le faisait remarquer ironiquement John von Neumann : "Quiconque considère des méthodes arithmétiques pour produire des nombres aléatoires" disait-il "est, bien sûr, en train de commettre un péché". De vrais nombres aléatoires peuvent être produits avec du matériel qui tire parti de certaines propriétés physiques stochastiques (bruit électronique d'une résistance par exemple). »

Argparse :

- Cette library nous donne la possibilité de lancer le programme avec des arguments que nous définissons dans le programme.

L'exemple type dans notre programme est de pouvoir lancer le serveur en affichant ou non la bannière via -q.

Re :

- Permets de vérifier une chaine de caractères via une expression régulière.

Exemple : (kite|wind)surf vérifiera : kitesurf OU windsurf
Nous utilisons cette library pour vérifier qu'une date entrée est valide par exemple. Ou bien pour vérifier un numéro de port/une adresse IP. Cela nous offre la possibilité d'écrire quelque chose de très compact qui normalement prendrait plusieurs lignes.

Time :

- Ce module nous fournit des fonctions en relation avec le temps.

Principalement utilisées dans notre projet afin de faire des time.sleep() pour laisser le temps à l'utilisateur de lire des messages d'erreurs ou bien de faire une attente pour le temps de notre ddos.

Platform:

- Cette bibliothèque permet de récupérer le plus d'informations possible sur l'ordinateur.

La seule fonction utilisée dans ce projet est platforme.node() qui permet de récupérer le nom de la machine infectée. Très utile pour pouvoir identifier quelle machine renvoie quel message.

Shutil :

- Shutil est une library qui nous offre la possibilité de procéder à des opérations de haut niveau. Comme copier des fichiers, en supprimer….

Ici, nous n'implémentons que la copie du fichier dans un autre dossier afin que l'utilisateur ne retrouve que difficilement le malware.

Subprocess:

- Ce module nous permet de lancer de nouveaux sous-processus. Cela nous permet par exemple de lancer une invite de commande et de récupérer son output.

C'est exactement ce que nous avons fait dans notre programme. Nous démarrons une invite de commande en tant que sous-processus afin de récupérer les informations système que renvoie la commande : systeminfo

Winreg :

- Winreg nous donne accès au registre de Windows.

Principalement utilisé pour créer une clé de registre afin que notre malware démarre au lancement de Windows. Cela permet de rendre un malware persistant sans trop de difficulté.

IV. Code server :

« Main » :

write_server_logs()	Cette fonction va écrire tous les évènements se passant sur le serveur dans un fichier afin de savoir ce qu'il aurait pu se passer si un crash survenait ou que l'on voulait tout simplement inspecter l'activité du serveur. Ces logs sont datés.

Server:

__init__()	Fonction d'initialisation des différentes variables que l'on va utiliser dans les autres fonctions. ➢ Variables argparse ➢ IP/port ➢ Le socket d'écoute ➢ La liste des sockets client ➢ La liste des adresses des clients ➢ Les deux dictionnaires de commandes
banner()	Ce bout de code choisit une bannière de manière pseudo-aléatoire lors du lancement du programme. Il n'affiche pas de bannière si le paramètre -q est présent.
start_all_thread()	C'est la fonction principale, car c'est celle-ci qui va lancer les deux threads principaux : (ListenSockets / CommandServer) Il vérifie si l'adresse IP et le port spécifiés sont corrects via un regex.
help()	Affiche simplement l'aide. Cette aide comprend la liste de toutes les commandes disponibles avec une petite description.
poweroff()	Lors de l'appel de la commande « poweroff », cette fonction démarre. Elle ferme la connexion de chaque client une par une avant d'arrêter le serveur.
targets_list()	Fonction qui affiche une liste concise avec chaque machine infectée par le malware.

ListenSockets :

- Cette classe qui hérite de threading.Thread va écouter les connexions entrantes en boucle tant qu'on ne lui dit pas de s'arrêter.
- Lorsqu'un client se connecte, le serveur accepte la connexion et lance un nouveau thread spécifique à ce client.

ClientStart :

- Comme ClientStart hérite également de threading.Thread et que ListenSockets lance cette classe pour chaque client, il y aura autant de thread ClientStart que de machines clientes connectées.
- On écoute sur chaque socket client ce qu'il pourrait nous envoyer. Cela peut être une réponse à une commande que l'on envoie par exemple.
- Cette classe gère les erreurs de connexion si un client perd la connexion ou autre.

CommandSender :

- CommandSender est aussi lancée en tant que thread. C'est une classe qui va vérifier et exécuter chaque action correspondante à chaque commande.
- Keylogger_start() enverra au client de démarrer l'enregistreur de frappes.
- Et ainsi de suite avec les différentes commandes clients/serveurs.
- Si une commande contient des arguments, on vérifie la commande sans les arguments et si celle-ci est valide on l'envoie et c'est côté client que les arguments sont vérifiés.
- Si la commande introduite est « poweroff », on arrête le serveur.

V. Code client :

Client :

__init__()	Fonction qui initialise les différentes variables que nous allons utiliser tout au long du programme ➢ Variables argparse ➢ IP/port ➢ Le socket ➢ Le nom de la machine
prepare_malware()	C'est la fonction qui va préparer le malware juste avant son utilisation. Celle-ci va : 1. Se copier lui-même dans %appdata% 2. Créer une clé de registre afin que le malware démarre au lancement de windows 3. Lancer en tâche de fond la version qu'il a copiée dans %appdata% 4. Afficher un message d'erreur de soi-disant compatibilité 5. Fermer le programme lancé de base afin de ne laisser place qu'à celui lancé en fond
connect_socket()	Ce bout de code va essayer de se connecter au serveur avec l'IP et le port spécifié via argparse
get_sys_info()	Cette fonction va récupérer les informations système du client afin de les envoyer au serveur
listen_command()	Lors de l'appel de cette fonction, l'écoute de commande provenant du serveur sur lequel le client s'est connecté démarre. Pour chaque commande reçue, le client exécutera une action en particulier.
keylogger_start()	Fonction qui lance la class Keylogger afin de démarrer l'enregistrement de frappes de clavier.
keylogger_stop()	Fonction qui arrête le Keylogger et écrit toutes les frappes de clavier dans un fichier afin de préparer la future commande : keylogger_dump(). Si la commande keylogger_start() n'a pas encore été lancée, la fonction affichera un message d'erreur correspondant.
keylogger_dump()	Récupère les informations contenues dans le fichier keylogs.txt afin de les envoyer au serveur.
ddos()	Lorsque le client reçoit la commande « ddos », il exécute la fonction ddos() qui va vérifier la justesse des arguments. Si c'est le cas, elle lance la class Ddos.

KeyLogger :

- Elle hérite de threading.thread donc elle peut tourner en parallèle du programme principal
- À chaque pression sur une touche du clavier, le keylogger l'ajoute dans une liste
- Celle-ci sera traitée ensuite par la fonction keylogger_stop()

Ddos :

- Ddos, lancée en tant que thread, vérifie si chaque paramètre de la commande est correct.
- Cela comprend l'IP, la date et la durée, vérifié par divers moyens comme les regex, des simples if…
- Si tous les arguments sont corrects, on lance les requêtes GET vers l'IP de destination à la date insérée. La durée introduite est gérée par DdosCounter.

DdosCounter :

- Cette classe démarre, bloque pendant x secondes et lorsque la durée est passée, se finit. Dès lors, le DDOS prend fin.

VI. Bibliographie :

Cours de développement – Henallux – BA2-Q1 – Sécurité des systèmes
https://pypi.org/project/pynput/
https://github.com/ncorbuk/Python-Keylogger/blob/master/Python%20advanced%20keylogger.py
https://pypi.org/project/requests/
https://developer.mozilla.org/fr/docs/Web/HTTP/Méthode/GET
https://docs.python.org/3/library/socket.html
https://docs.python.org/3/library/threading.html
https://docs.python.org/3/library/ctypes.html
https://www.devdungeon.com/content/dialog-boxes-python
https://stackoverflow.com/questions/4485610/python-message-box-without-huge-library-dependency
https://fr.wikipedia.org/wiki/Dynamic_Link_Library
https://docs.python.org/3/library/datetime.html
https://docs.python.org/3/library/os.html
https://stackoverflow.com/questions/23608547/environmental-variables-os-path-expandvars
https://stackoverflow.com/questions/595305/how-do-i-get-the-path-of-the-python-script-i-am-running-in
https://docs.python.org/3/library/random.html
https://fr.wikipedia.org/wiki/Générateur_de_nombres_pseudo-aléatoires
https://docs.python.org/3/library/argparse.html
https://docs.python.org/3/library/re.html
https://www.youtube.com/watch?v=dinW2QTSNI4
https://blog.usejournal.com/understanding-regex-101-204853651755
https://regexr.com
https://regex101.com
https://pythex.org/
https://stackoverflow.com/questions/11264005/using-a-regex-to-match-ip-addresses-in-python/11264056
https://stackoverflow.com/questions/12968093/regex-to-validate-port-number/12968117
https://stackoverflow.com/questions/4709652/python-regex-to-match-dates
https://docs.python.org/3/library/time.html
https://docs.python.org/3/library/platform.html
https://docs.python.org/3/library/shutil.html
https://www.geeksforgeeks.org/python-shutil-copyfile-method/
https://kite.com/python/examples/4260/shutil-copy-a-directory-tree,-ignoring-paths-that-match-a-pattern
https://docs.python.org/3/library/subprocess.html
https://stackoverflow.com/questions/325463/launch-a-shell-command-with-in-a-python-script-wait-for-the-termination-and-ret
https://docs.python.org/3/library/winreg.html
https://stackoverflow.com/questions/26762511/nameerror-name-openkey-is-not-defined-using-winreg