МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра Інформаційної Безпеки

Зворотна розробка та аналіз шкідливого програмного забезпечення

Модульна контрольна робота №1

Аналіз зразків шкідливого програмного забезпечення

Перевірив:

_____

Виконав:
студент II курсу
групи ФБ-01
Сахній Н.Р.

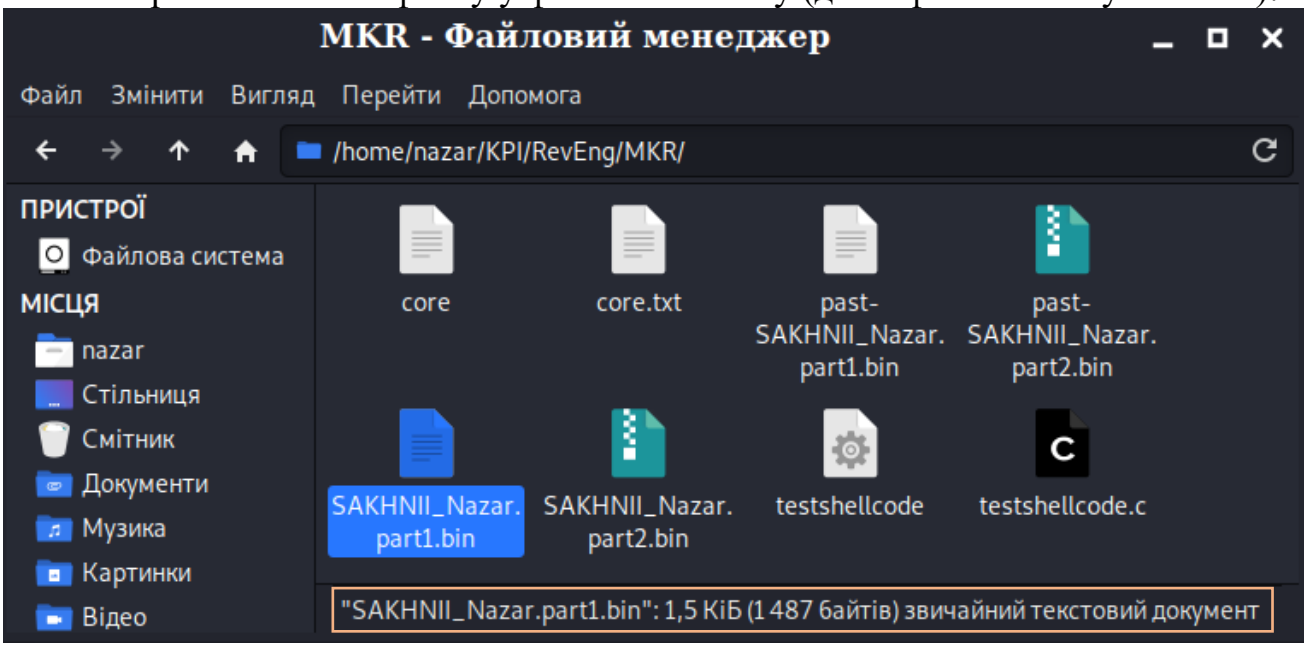
Київ 2022

Завдання 1: `hvost{y861mpUq}bobra`

- Отримано зразок ШПЗ: “SAKHNII_Nazar.part1.bin”.

```
┌──(nazar㉿snz24)-[/home/nazar/KPI/RevEng/MKR]
└─$ file SAKHNII_Nazar.part1.bin
SAKHNII_Nazar.part1.bin: ASCII text
```

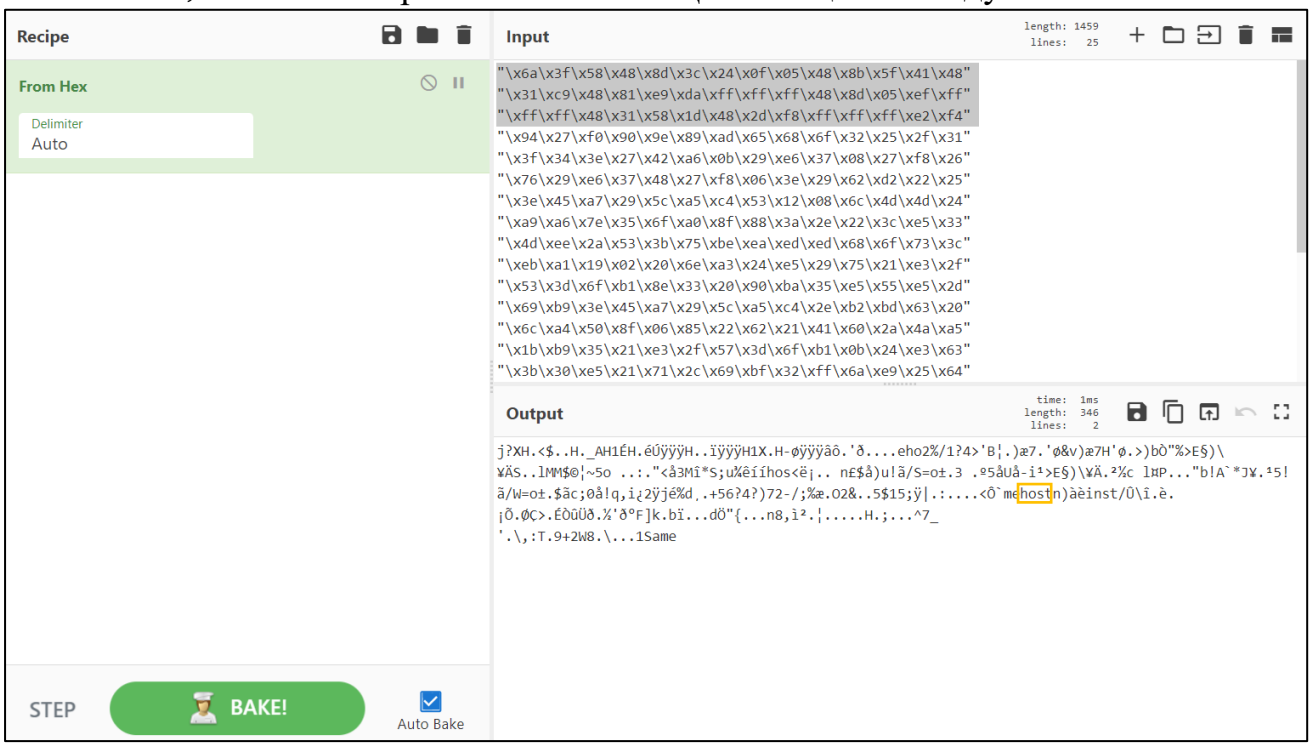
- Переглянемо тип файлу у файлообміннику (деякі файли з минулої МКР).



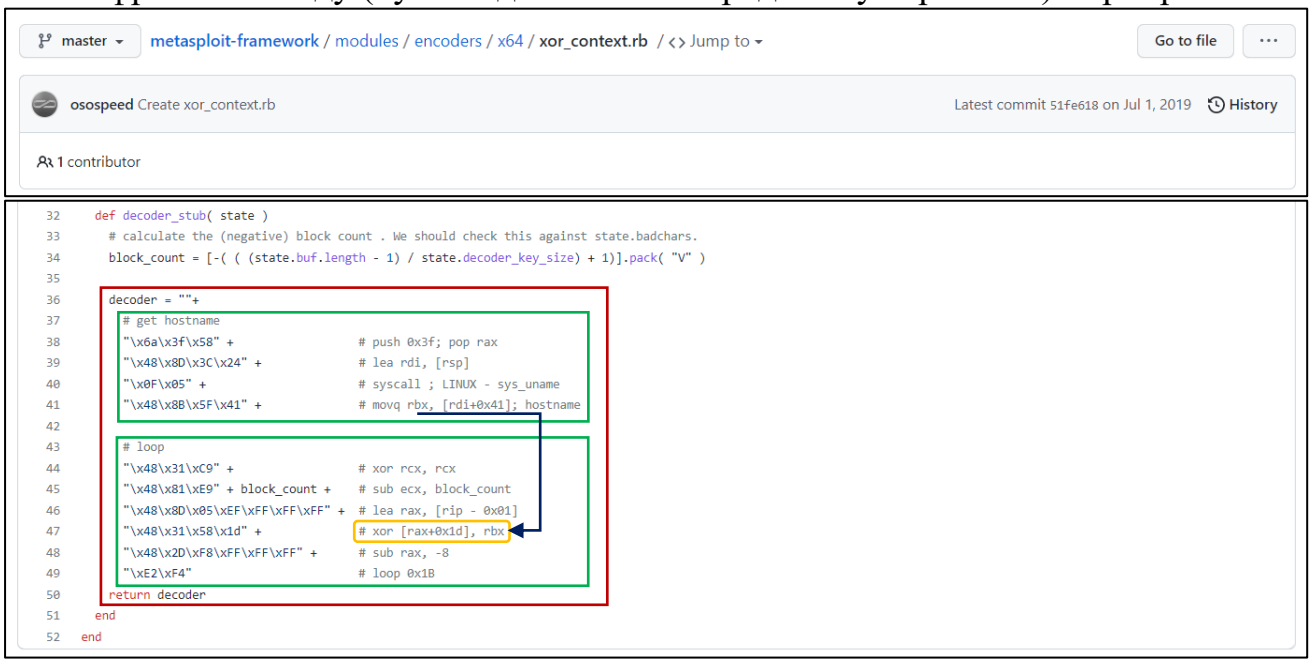
- Переглувши вміст цього файлу, помітимо, що у ньому знаходиться “сирий” фрагмент викононуваного коду. Тому далі спробуємо знайти щось цікаве.

```
┌──(nazar㉿snz24)-[/home/nazar/KPI/RevEng/MKR]
└─$ cat SAKHNII_Nazar.part1.bin
unsigned char buffer[] =
"\x6a\x3f\x58\x48\x8d\x3c\x24\x0f\x05\x48\x8b\x5f\x41\x48"
"\x31\xc9\x48\x81\xe9\xda\xff\xff\xff\x48\x8d\x05\xef\xff"
"\xff\xff\x48\x31\x58\x1d\x48\x2d\xf8\xff\xff\xff\xe2\xf4"
"\x94\x27\xf0\x90\x9e\x89\xad\x65\x68\x6f\x32\x25\x2f\x31"
"\x3f\x34\x3e\x27\x42\xa6\x0b\x29\xe6\x37\x08\x27\xf8\x26"
"\x76\x29\xe6\x37\x48\x27\xf8\x06\x3e\x29\x62\xd2\x22\x25"
"\x3e\x45\xa7\x29\x5c\xa5\xc4\x53\x12\x08\x6c\x4d\x4d\x24"
"\xa9\xa6\x7e\x35\x6f\xa0\x8f\x88\x3a\x2e\x22\x3c\xe5\x33"
"\x4d\xee\x2a\x53\x3b\x75\xbe\xea\xed\xed\x68\x6f\x73\x3c"
"\xeb\xa1\x19\x02\x20\x6e\xa3\x24\xe5\x29\x75\x21\xe3\x2f"
"\x53\x3d\x6f\xb1\x8e\x33\x20\x90\xba\x35\xe5\x55\xe5\x2d"
"\x69\xb9\x3e\x45\xa7\x29\x5c\xa5\xc4\x2e\xb2\xbd\x63\x20"
"\x6c\xa4\x50\x8f\x06\x85\x22\x62\x21\x41\x60\x2a\x4a\xa5"
"\x1b\xb9\x35\x21\xe3\x2f\x57\x3d\x6f\xb1\x0b\x24\xe3\x63"
"\x3b\x30\xe5\x21\x71\x2c\x69\xbf\x32\xff\x6a\xe9\x25\x64"
"\xb8\x2e\x2b\x35\x36\x3f\x34\x3f\x29\x37\x32\x2d\x2f\x3b"
"\x25\xe6\x84\x4f\x32\x26\x91\x81\x35\x24\x31\x35\x3b\xff"
"\x7c\x88\x3a\x9a\x97\x90\x2e\x3c\xd4\x60\x6d\x65\x68\x6f"
"\x73\x74\x6e\x29\xe0\xe8\x69\x6e\x73\x74\x2f\xdb\x5c\xee"
"\x07\xe8\x8c\xa1\xd5\x91\xd8\xc7\x3e\x2e\xc9\xd2\xfb\xdc"
"\xf0\x9a\xbd\x27\xf0\xb0\x46\x5d\x6b\x19\x62\xef\x88\x94"
"\x1b\x64\xd6\x22\x7b\x1d\x1c\x1e\x6e\x38\x2c\xec\xb2\x90"
"\xa6\x1f\x07\x15\x19\x1c\x48\x0e\x3b\x2e\x18\x02\x5e\x37"
"\x5f\x0a\x27\x13\x5c\x2c\x3a\x54\x1f\x39\x2b\x32\x57\x38"
"\x00\x5c\x01\x0c\x1e\x31\x53\x61\x6d\x65";
```

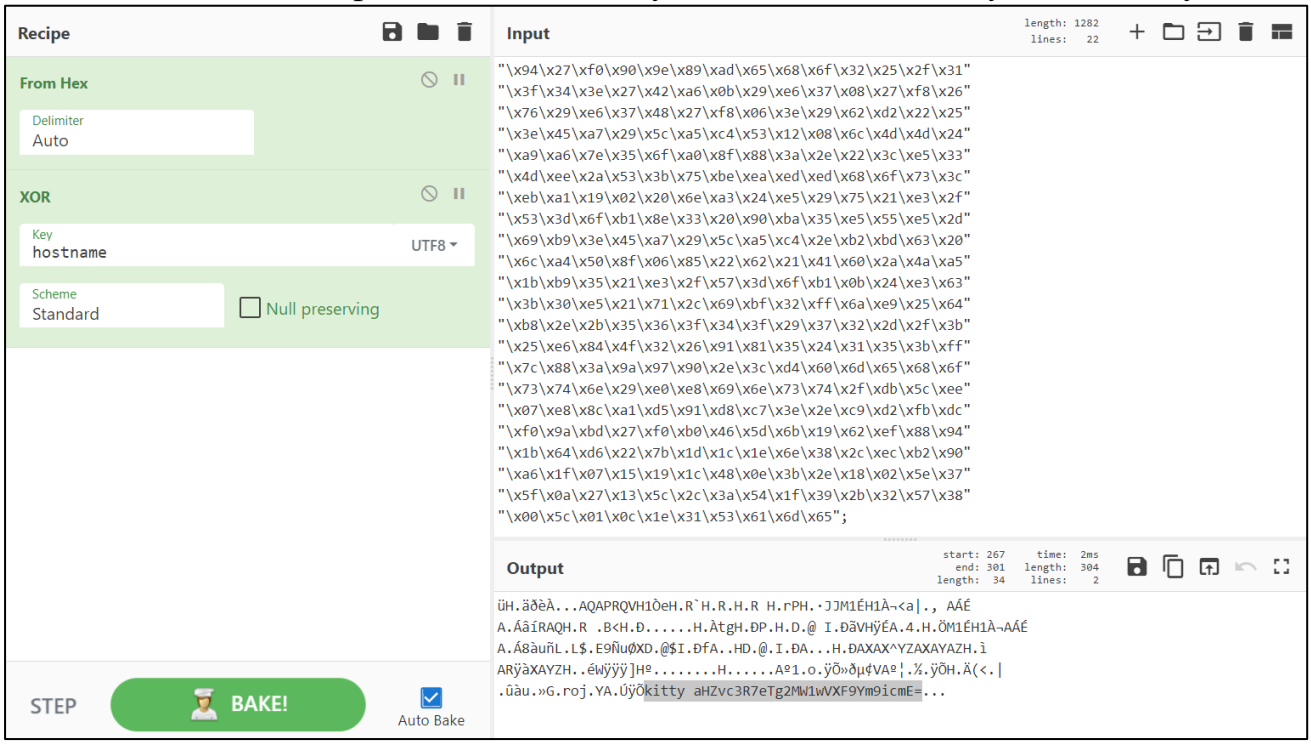
- Спробуємо розкодувати raw-дані у CyberChef. На виході бачимо абсолютно незрозумілий набір символів, однак там присутнє єдине слово англійської мови, яке можна прочитати – host. Це мені щось нагадує…



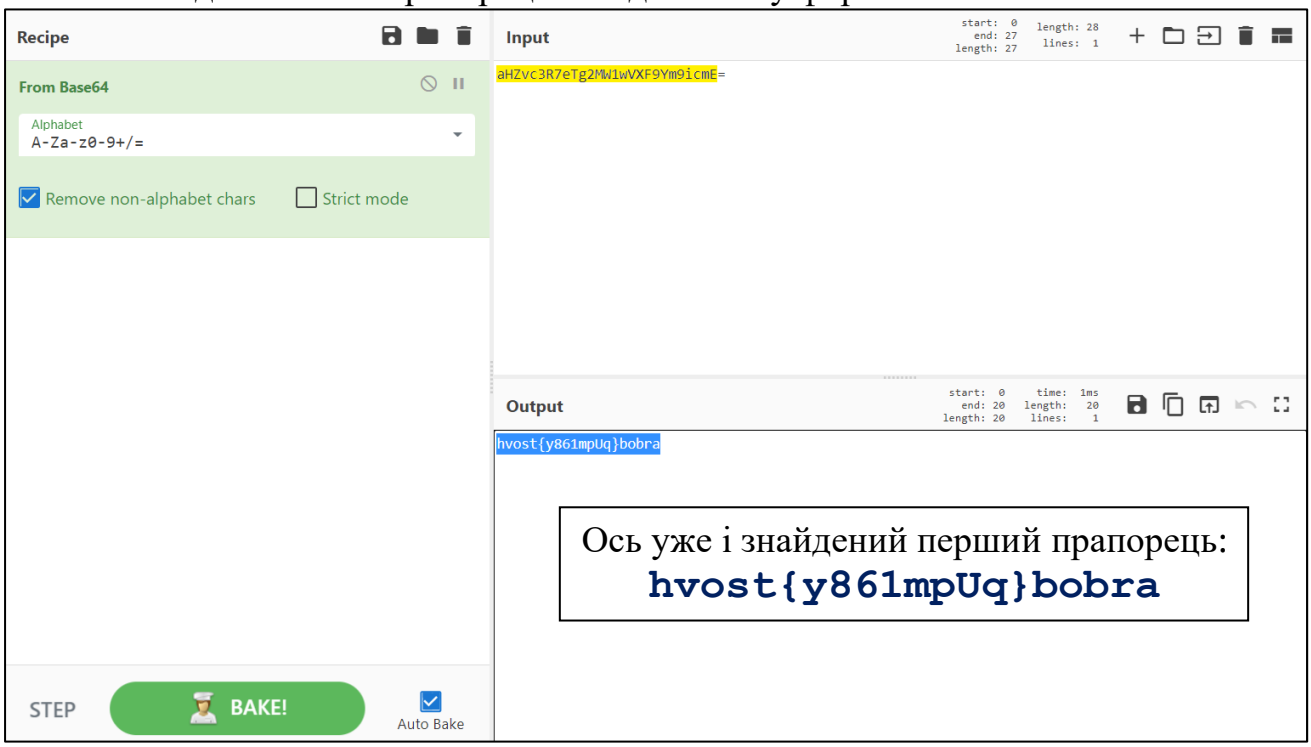
- Схоже, що декодер `x64/xor_context` використовує значення hostname при кодуванні корисного навантаження у metasploit-framework. Звідси, якщо переглянути детальніше процес кодування, можемо помітити, що деякі байти відповідають за декодер, тому можна спробувати їх знайти у фрагменті коду (були виділені на попередньому скріншоті) і прибрати.



- Отже, прибравши їх як просто додатковий заголовок виконуваного коду, отримаємо послідовні байти закодованого корисного навантаження. Тоді ж можна спробувати розкодувати їх за допомогою операції XOR (про це дізнались в .rb файлі, де ключем буде слово hostname у UTF-8 кодуванні.



- Нарешті ми побачили знайоме слово `kitty`, а біля нього скоріш за все і знаходиться наш прапорець закодований у форматі base64.



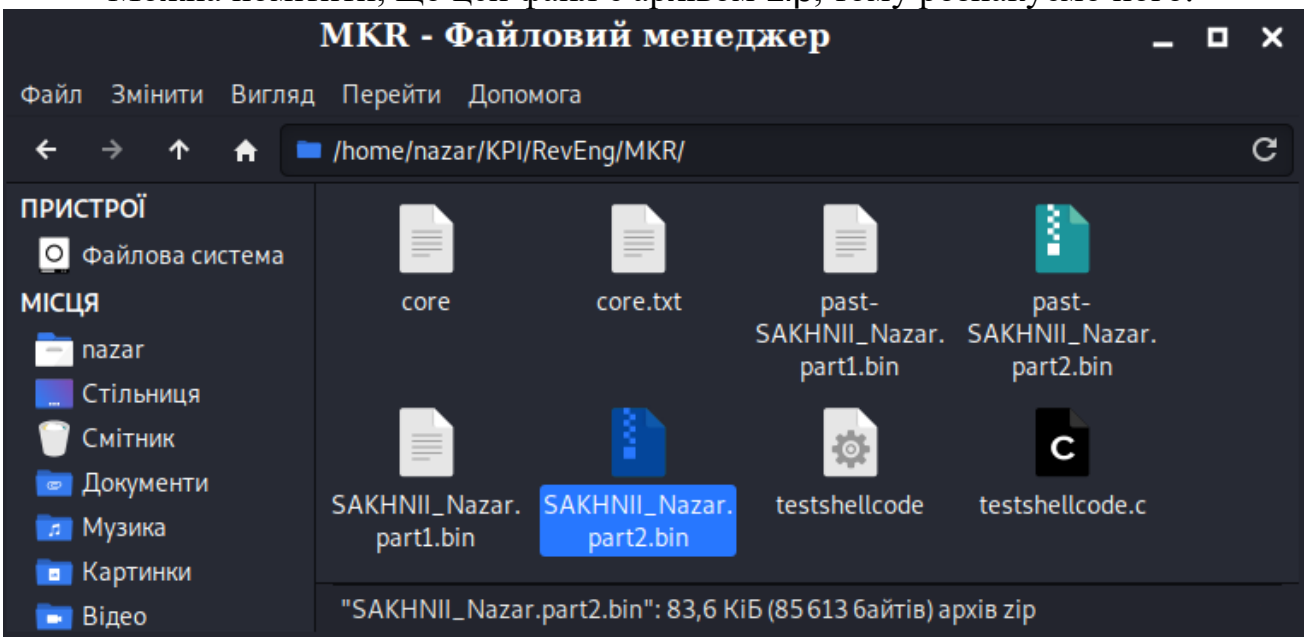
Ось уже і знайдений перший прапорець:
hvost{y861mpUq}bobra

Завдання 2: `hvost{GTmzxWgl}bobra`

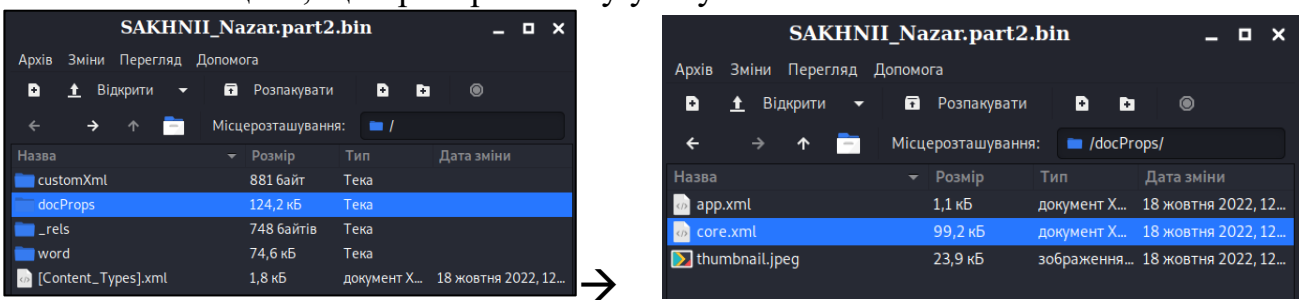
- Отримано зразок ШПЗ: “SAKHNII_Nazar.part2.bin”.

```
┌──(nazar㉿snz24)-[/home/nazar/KPI/RevEng/MKR]
└─$ file SAKHNII_Nazar.part2.bin
SAKHNII_Nazar.part2.bin: Microsoft Word 2007+
```

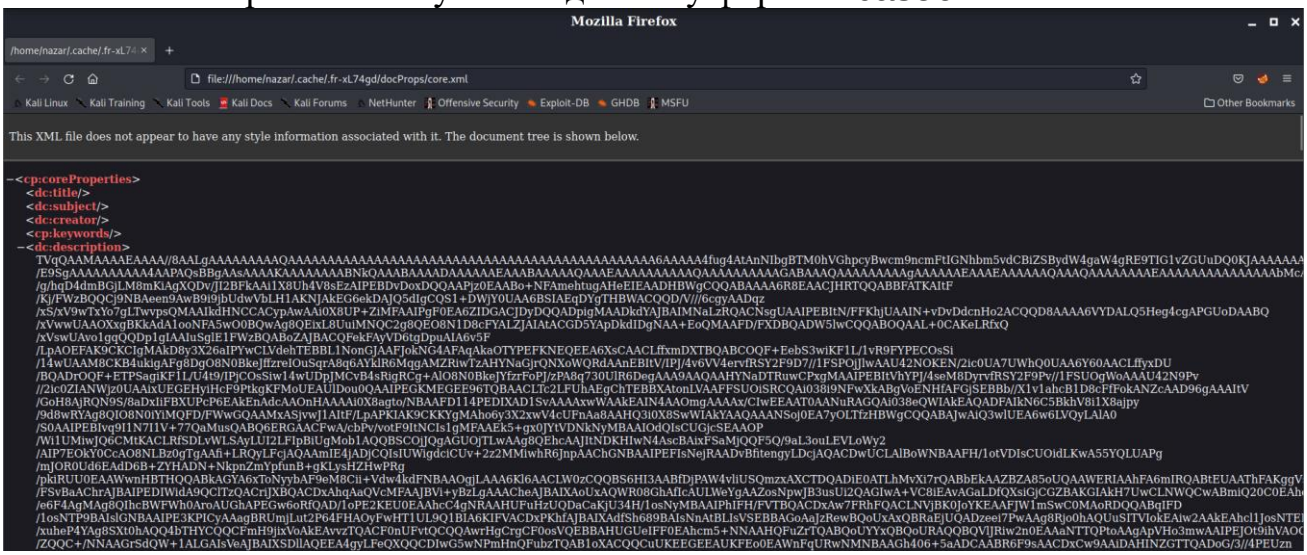
- Можна помітити, що цей файл є архівом zip, тому розпакуємо його.



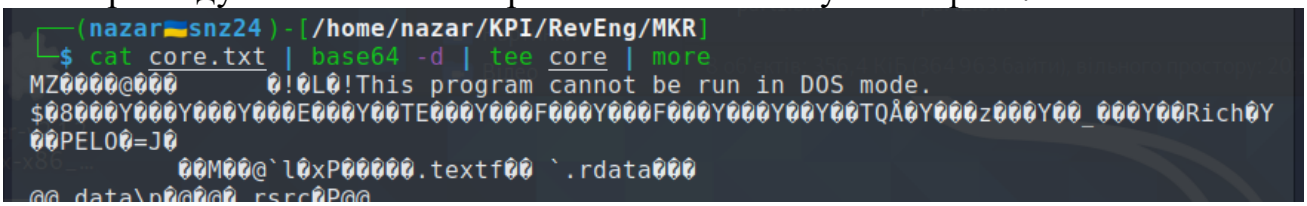
- Спробуємо повноцінно переглянути вміст цього архіву, можливо зможемо знайти щось, що приверне нашу увагу.



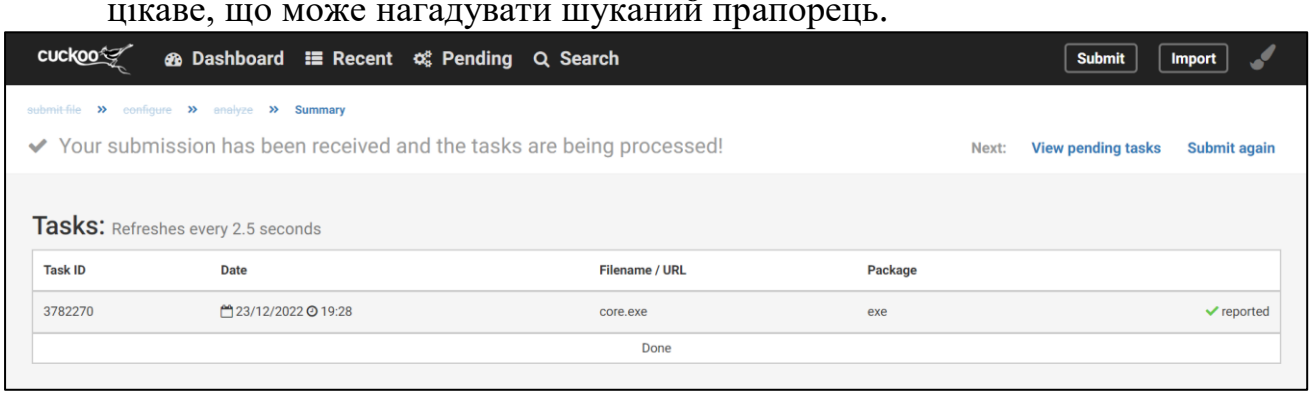
- У файлі `core.xml` із у папки `docProps`, бачимо, що між тегами `</dc:description>...</dc:description>` знаходиться деякий текст, який скоріш за все було закодовано у форматі base64.



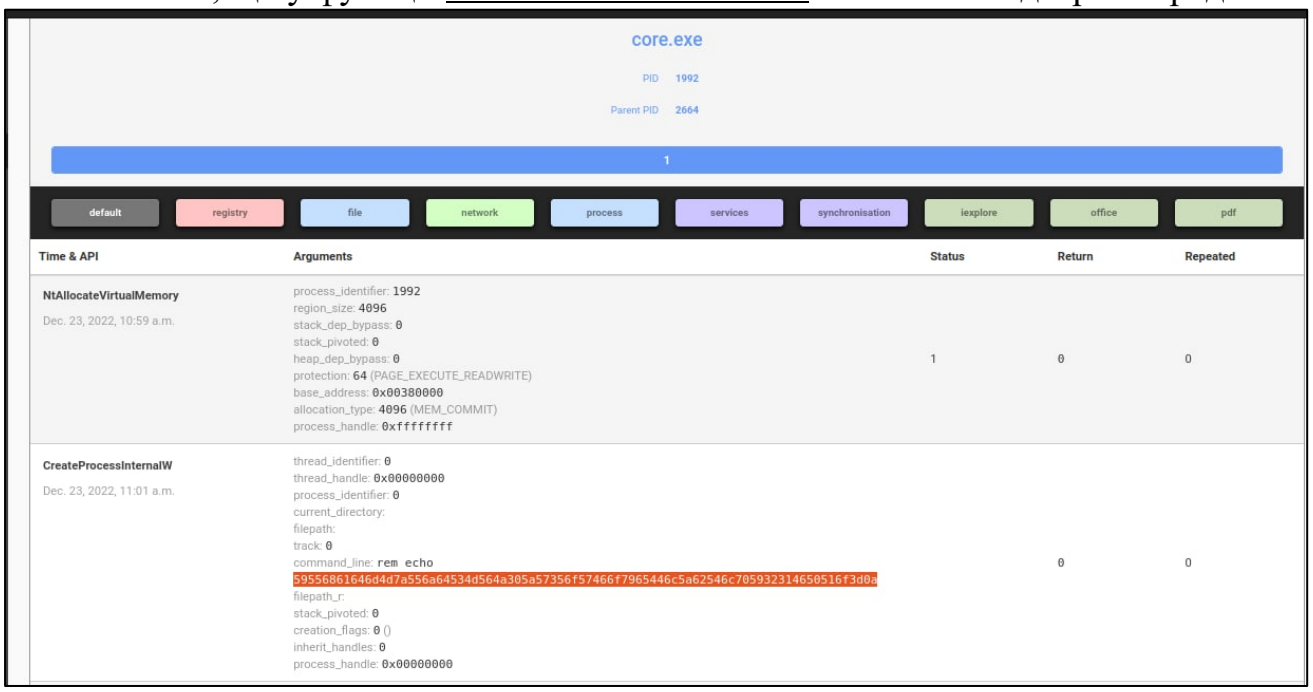
- Збережемо цей закодований фрагмент у текстовому файлі та спробуємо розкодувати його та створити із нього виконуваний файл.



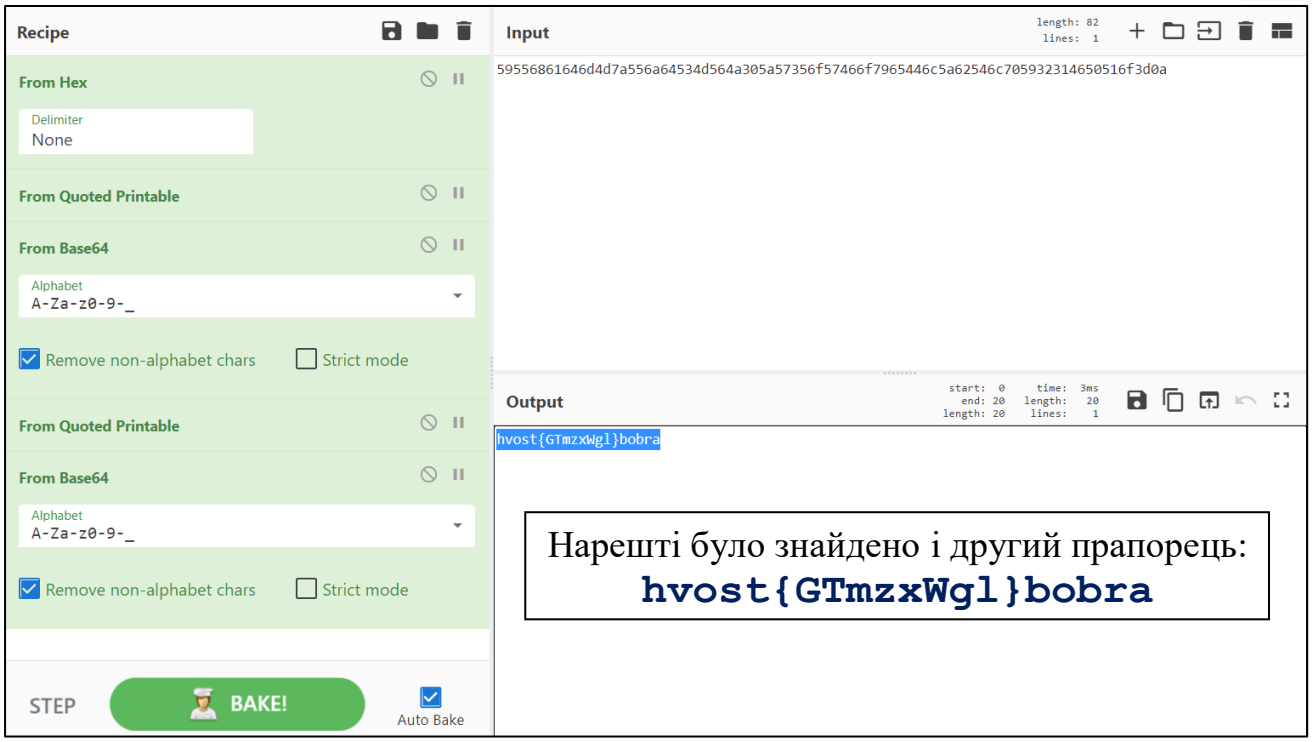
- Далі завантажуємо файл у середовище “Cuckoo Sandbox”, щоб знайти щось цікаве, що може нагадувати шуканий прапорець.



- Схоже, що у функції CreateProcessInternalW захований підозрілий рядок.



- За допомогою “магічної палички” розкодуємо отриманий фрагмент.



Нарешті було знайдено і другий прапорець:
hvost{GTmzxWgl}bobra