



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.

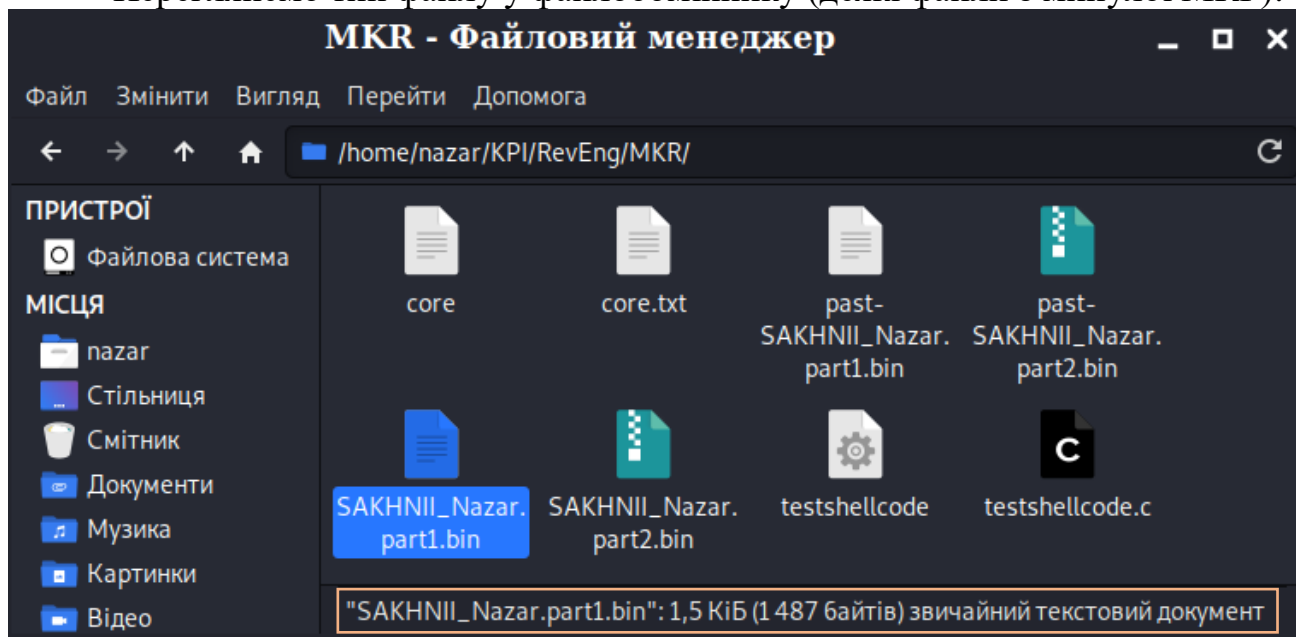
Київ 2022

Завдання 1: `hvest{y861mpUq}bobra`

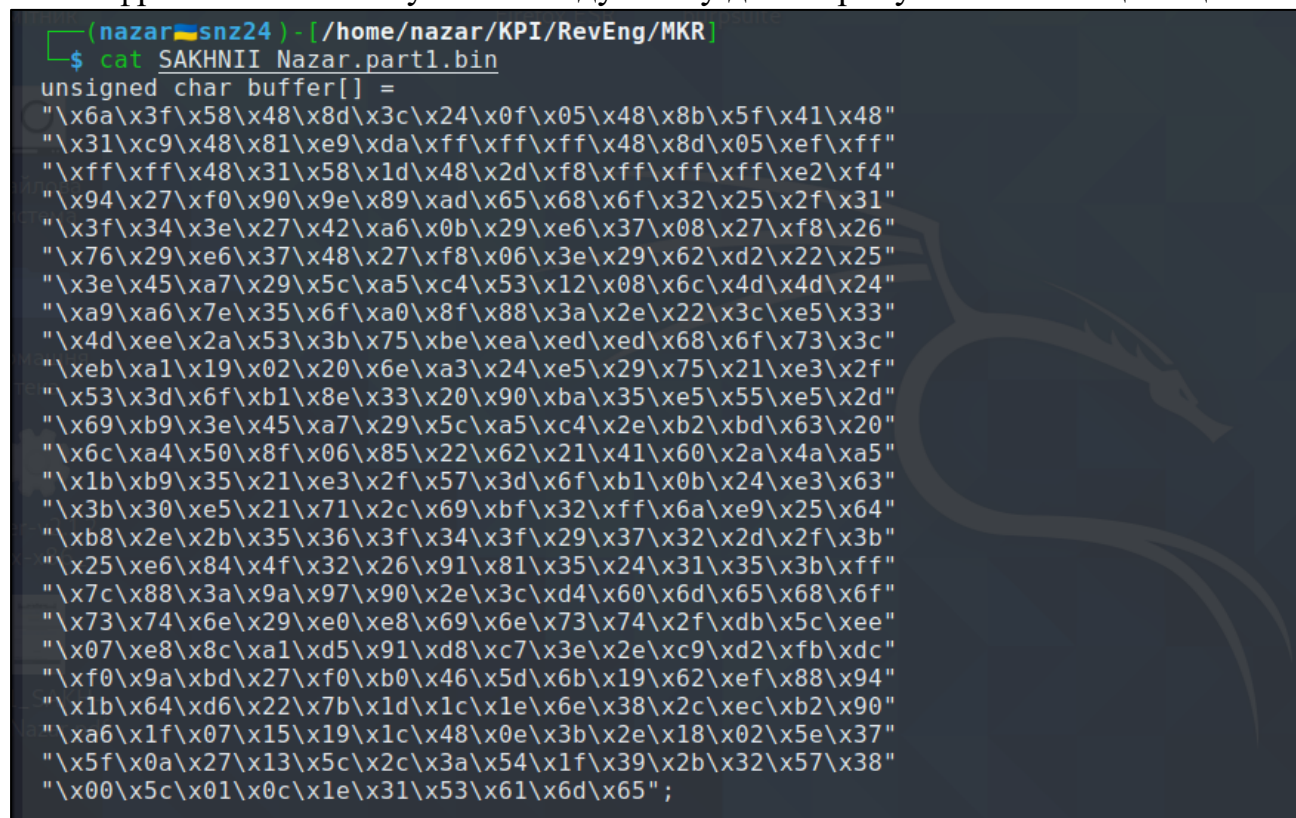
- Отримано зразок ШПЗ: "SAKHNIИ_Nazar.part1.bin".

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

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



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



- 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...

The screenshot shows the CyberChef interface. On the left, the 'Recipe' panel is set to 'From Hex' with 'Delimiter' set to 'Auto'. The 'Input' panel contains a large block of hex-encoded data. The 'Output' panel shows the decoded result, which is a mix of random characters and the word 'host' highlighted in yellow.

- 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.

The screenshot shows the source code of the `x64/xor_context.rb` module in the Metasploit Framework. The code is shown in a web browser. A red box highlights the 'decoder' section, and a green box highlights the 'loop' section. A blue arrow points from the 'loop' section to the 'decoder' section, indicating the flow of execution.

- 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.

Recipe

From Hex

Delimiter
Auto

XOR

Key
hostname UTF8

Scheme
Standard ☐ Null preserving

STEP Auto Bake

Input

length: 1282
lines: 22

```
"\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"
"\xb1\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\x2d\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";
```

Output

start: 267 time: 2ms
end: 301 length: 304
length: 34 lines: 2

```
ÜH.äðĖ...AQAPRVHİöEH.R`H.R.H.R H.rPH.·JJM1ÉH1Ä~<a|. , AĖ
A.ĀāīRAQH.R .B<H.Đ. ....H.Ātgh.ĐP.H.D.@ I.ĐāVhŸĖĀ.4.H.ŌM1ÉH1Ä~AĖ
A.ĀšauñL.L$.E9ñu0XD.@$I.Đfa..HD.@.I.ĐA...H.ĐAXAX^YZAXAYAZH.i
ARÿäXAYZH..ėwŸŸŸ]Hə.....H.....Aº1.o.ŸÖ»ðµŸVæ|.%.ŸÖH.Ā(<.|
.ūau.»G.roj.YA.ŸŸkitty aHZvc3R7eTg2Mw1wVXF9Ym9icmE=...
```

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

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars ☐ Strict mode

STEP Auto Bake

Input

start: 0 length: 28
end: 27 lines: 1
length: 27

```
aHZvc3R7eTg2Mw1wVXF9Ym9icmE=
```

Output

start: 0 time: 1ms
end: 20 length: 20
length: 20 lines: 1

```
hvest{y861mpUq}bobra
```

Ось уже і знайдений перший прапорець:

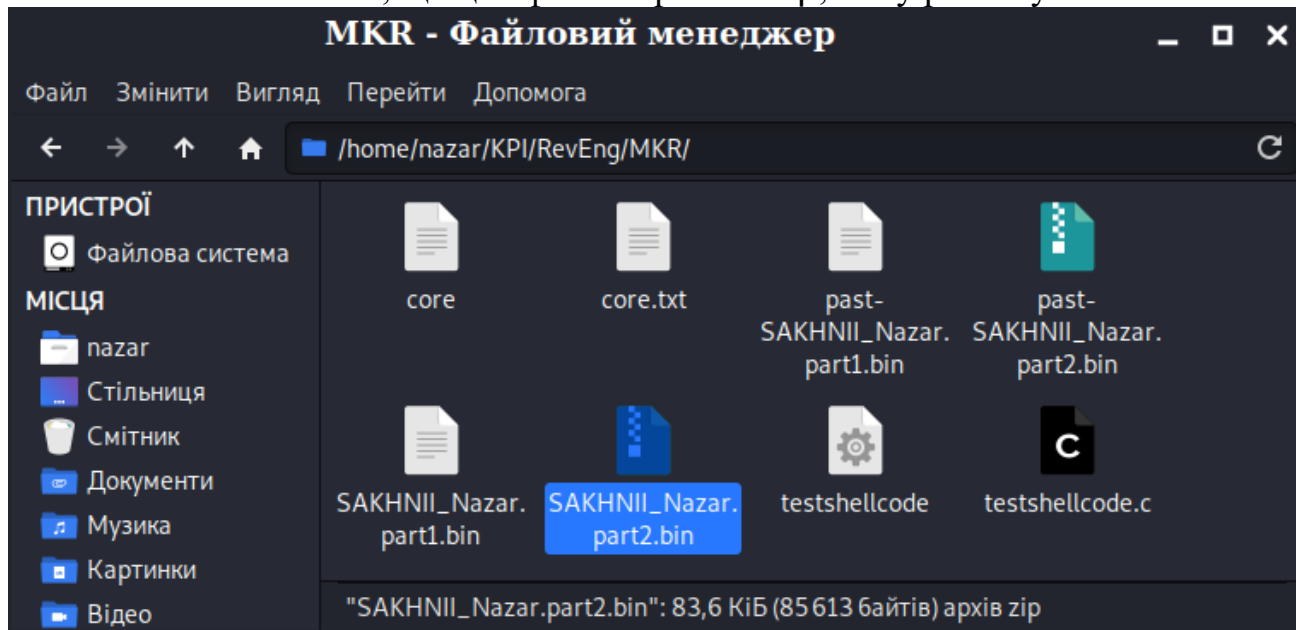
hvest{y861mpUq}bobra

Завдання 2: `hvest{GTmzxWgl}bobra`

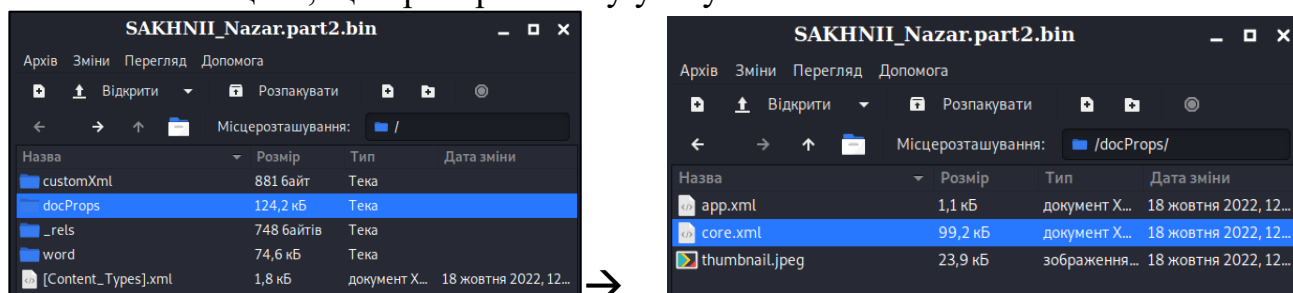
- Отримано зразок ШПЗ: “SAKHNIИ_Nazar.part2.bin”.

```
(nazar@snz24) - [/home/nazar/KPI/RevEng/MKR]  
$ file SAKHNIИ_Nazar.part2.bin  
SAKHNIИ_Nazar.part2.bin: Microsoft Word 2007+
```

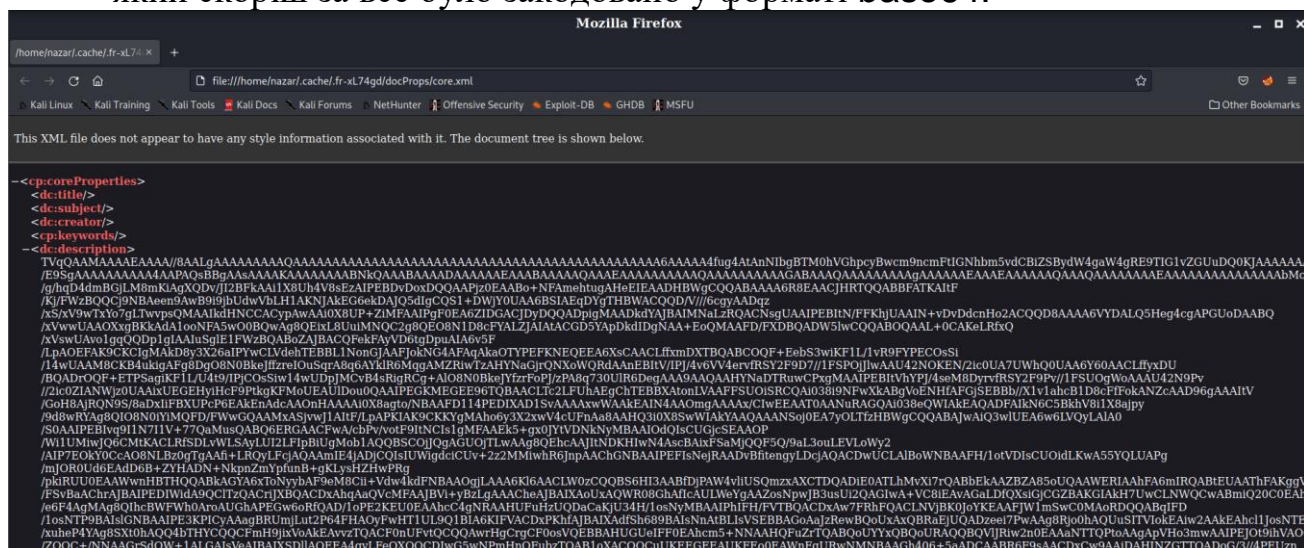
- Можна помітити, що цей файл є архівом zip, тому розпакуємо його.



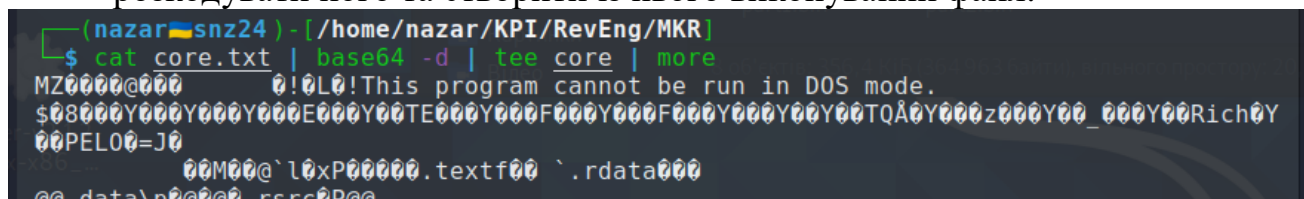
- Спробуємо повноцінно переглянути вміст цього архіву, можливо зможемо знайти щось, що приверне нашу увагу.



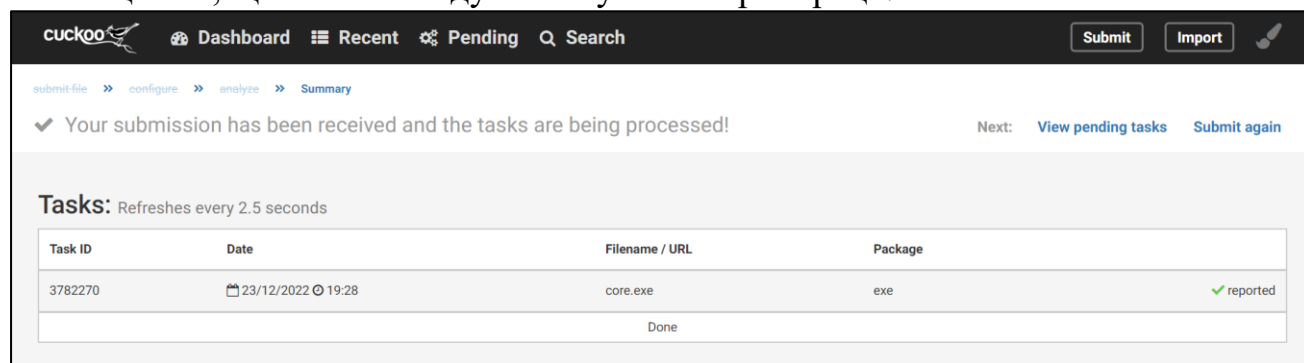
- У файлі core.xml із у папки docProps, бачимо, що між тегами `</dc:description>...</dc:description>` знаходиться деякий текст, який скоріш за все було закодовано у форматі base64.



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



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



- Схоже, що у функції CreateProcessInternalW захований підозрілий рядок.

core.exe				
PID 1992				
Parent PID 2664				
1				
Time & API	Arguments	Status	Return	Repeated
NtAllocateVirtualMemory Dec. 23, 2022, 10:59 a.m.	process_identifier: 1992 region_size: 4096 stack_dep_bypass: 0 stack_pivoted: 0 heap_dep_bypass: 0 protection: 64 (PAGE_EXECUTE_READWRITE) base_address: 0x00380000 allocation_type: 4096 (MEM_COMMIT) process_handle: 0xffffffff	1	0	0
CreateProcessInternalW Dec. 23, 2022, 11:01 a.m.	thread_identifier: 0 thread_handle: 0x00000000 process_identifier: 0 current_directory: filepath: track: 0 command_line: rem echo 59556861646d4d7a556a64534d564a305a57356f57466f7965446c5a62546c705932314650516f3d0a filepath_r: stack_pivoted: 0 creation_flags: 0 () inherit_handles: 0 process_handle: 0x00000000		0	0

- За допомогою “магічної палички” розкодуємо отриманий фрагмент.



Recipe

From Hex

Delimiter: None

From Quoted Printable

From Base64

Alphabet: A-Za-z0-9-_-

☒ Remove non-alphabet chars
 ☐ Strict mode

From Quoted Printable

From Base64

Alphabet: A-Za-z0-9-_-

☒ Remove non-alphabet chars
 ☐ Strict mode

STEP

BAKE!

Auto Bake

Input

length: 82, lines: 1

59556861646d4d7a556a64534d564a305a57356f57466f7965446c5a62546c705932314650516f3d0a

Output

start: 0, end: 20, length: 20, time: 3ms, lines: 1

hvost{GTmzxWgl}bobra

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