



d2.zip
ZIP-архив

Решаем крэки. Вот дампы кода, данных и импорта после загрузки.

Обработка заняла 18m 4s >

Крэки решён

Это VM keygenme by Genaytyk, 32-битный PE со своей виртуальной машиной. Нативный код только получает имя/серийник, запускает VM и затем проверяет байт успешного завершения.

Основные адреса:

VM-интерпретатор: 0x00401000
Таблица опкодов: 0x00403000
Конфигурация VM: 0x004034A8
Начало байткода: 0x004034BE
Размер байткода: 0x000007AC
База VM-данных: 0x00403C6A
Размер данных: 0x00000113
Флаг успеха: data[0x109] = 0x00403D73



Я декодировал весь поток — получилось 388 VM-инструкций.

Формат серийника

Серийный номер обязательно имеет вид:

XXXXXX-XXXXXX-XXXXXX



Используется собственный алфавит из 41 символа:

aAb0cBd1eCf2gDh3jEk4lFm5nGp6qHr7sJt8uKv9w



VM проверяет:

- ровно два дефиса;
- ровно шесть символов в каждой части;
- отсутствие посторонних символов;

- зависимость серийника от введенного имени.

Каждая шестисимвольная группа преобразуется в число по основанию 41:

```
value = ((((((d0 * 41 + d1) * 41 + d2) * 41 + d3) * 41 + d4) * 41 + d5) mod 2^32
```



Затем три числа предварительно смешиваются:

```
a  = ROL32(n0, 7)
b  = ROR32(n1, 9)
c  = ROR32(n2, 11)

m0 = a XOR b
m1 = b
m2 = c + m1 - m0
```



Как проходит основная проверка

Из имени VM формирует 96-байтную область и вычисляет несколько хешей.

Первая стадия:

```
block = m0 || m1
key1  = name_hash_left || name_hash_right

block = F_key1^10(block)
```



Старший DWORD результата должен совпасть с отдельной контрольной суммой имени.

После этого старший DWORD заменяется на m2 , и выполняется вторая стадия:

```
key2 = name_hash_a || 0x6B79745F

block = F_key2^15(block)
```



Итоговые восемь байт должны стать:

```
OKAY_GUY
```



Тогда VM устанавливает:

```
data[0x109] = 1
```



Почему keygen работает без перебора

Преобразование F полностью обратимо: оно состоит из XOR, циклических сдвигов, сложения и вычитания по модулю, побайтового NOT и обратимых перестановок.

Поэтому keygen делает следующее:

```
pre2 = F_key2^(-15)("OKAY_GUY")  
pre1 = F_key1^(-10)(pre2.low || name_checksum)
```



После этого отменяется предварительное смешивание:

```
n1 = ROL32(m1, 9)  
n0 = ROR32(m0 XOR m1, 7)  
n2 = ROL32(m2 - m1 + m0, 11)
```



Полученные три числа кодируются шестью символами по основанию 41.

Готовые примеры

Для имени:

```
zer0kernel
```



серииник:

```
mHmp85-7rKJ81-hd5kej
```



Другие проверенные варианты:

```
abc          → c5n8fn-dc8dcB-EFfKrb  
0leg Ivanov → nAHfs8-d1D8kF-JGcCBj
```



Я дополнительно прогнал результаты через эмулятор исходного VM-байткода: блок по смещению 0x93 становится равен OKAY_GUY, а флаг по смещению 0x109 устанавливается в 1.

Запуск keygen

```
</> Bash  
  
python3 vm_keygen_genaytyk.py "zer0kernel"
```



Проверка существующего серииника:

```
</> Bash
```



```
python3 vm_keygen_genaytyk.py "zer0kernel" \  
--verify "mHmp85-7rKJ8l-hd5kej"
```

Для кириллицы нужно учитывать, что оригинал использует `GetWindowTextA` .
Кодировка должна совпадать с ANSI code page системы:

</> Bash



```
python3 vm_keygen_genaytyk.py "Олег" --encoding cp1251
```

Файлы:

- </> [vm_keygen_genaytyk.py](#)
-  [vm_keygenme_analysis.md](#)
-  [vm_keygenme_vm_disassembly.txt](#)

vm_keygen_genaytyk.py

Код



vm_keygenme_analysis.md

Документ



vm_keygenme_vm_disassembly.txt

Документ



ChatGPT может допускать ошибки. Данные рабочей области не используются для обучения моделей.