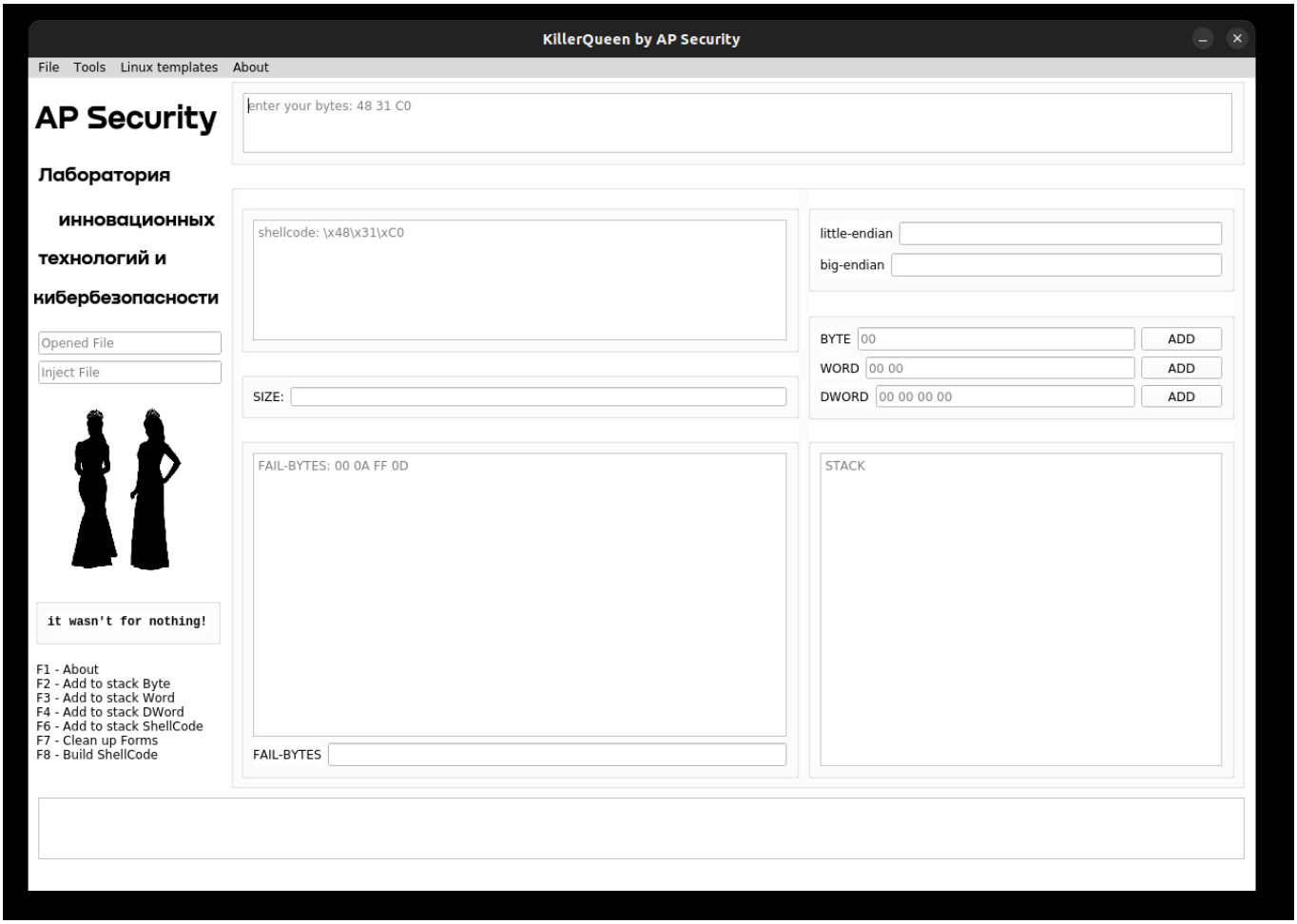
Сегодня пентестерам необходимо множество инструментов для проверки системы на безопасность. Одной из таких проверок является внедрения в систему бэкдоров и посылка ВПО цели для тестирования на проникновения. В данной статье будет рассмотрена утилита для реверс инжиниринга под названием **KillerQueen**.

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.

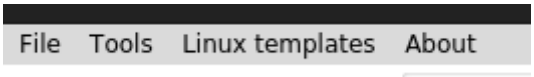
Данное программное обеспечение умеет делать сразу множество вещей:

1. Создание полноценного шеллкода для его введения в экплойт
2. Проверять шеллкод на плохие байты (`0x00 0x0A 0xFF 0x0D`)
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)

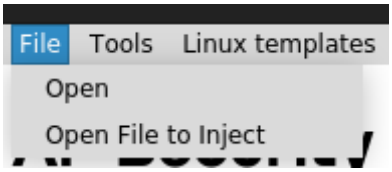
Обо всем по порядку. Сначала главный экран и основной функционал.



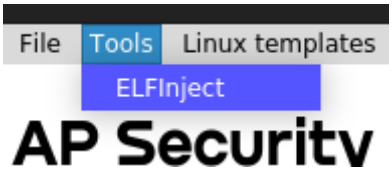
В нижнем углу представлена раскладка для горячих клавиш. Это ускоряет работу в программе. На верху присутвует ряд вкладок



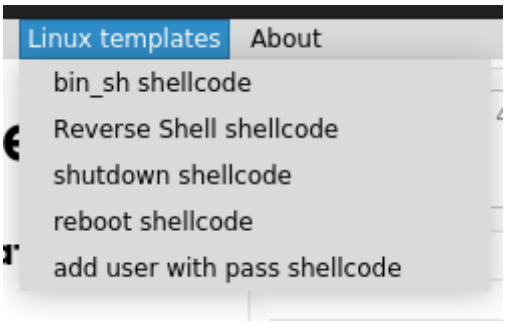
Во вкладке `File`, можно открыть файл, который должен представлять из себя уже скомпилированную программу написанной, например, на языке Ассемблер. Вторая вкладка представляет собой открытие файла для инъекции в него шеллкода.



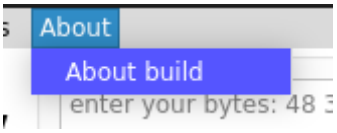
Вкладка `Tools`, в данной сборке, содержит пока что одну функцию - инъекцию в исполняемый файл шеллкод

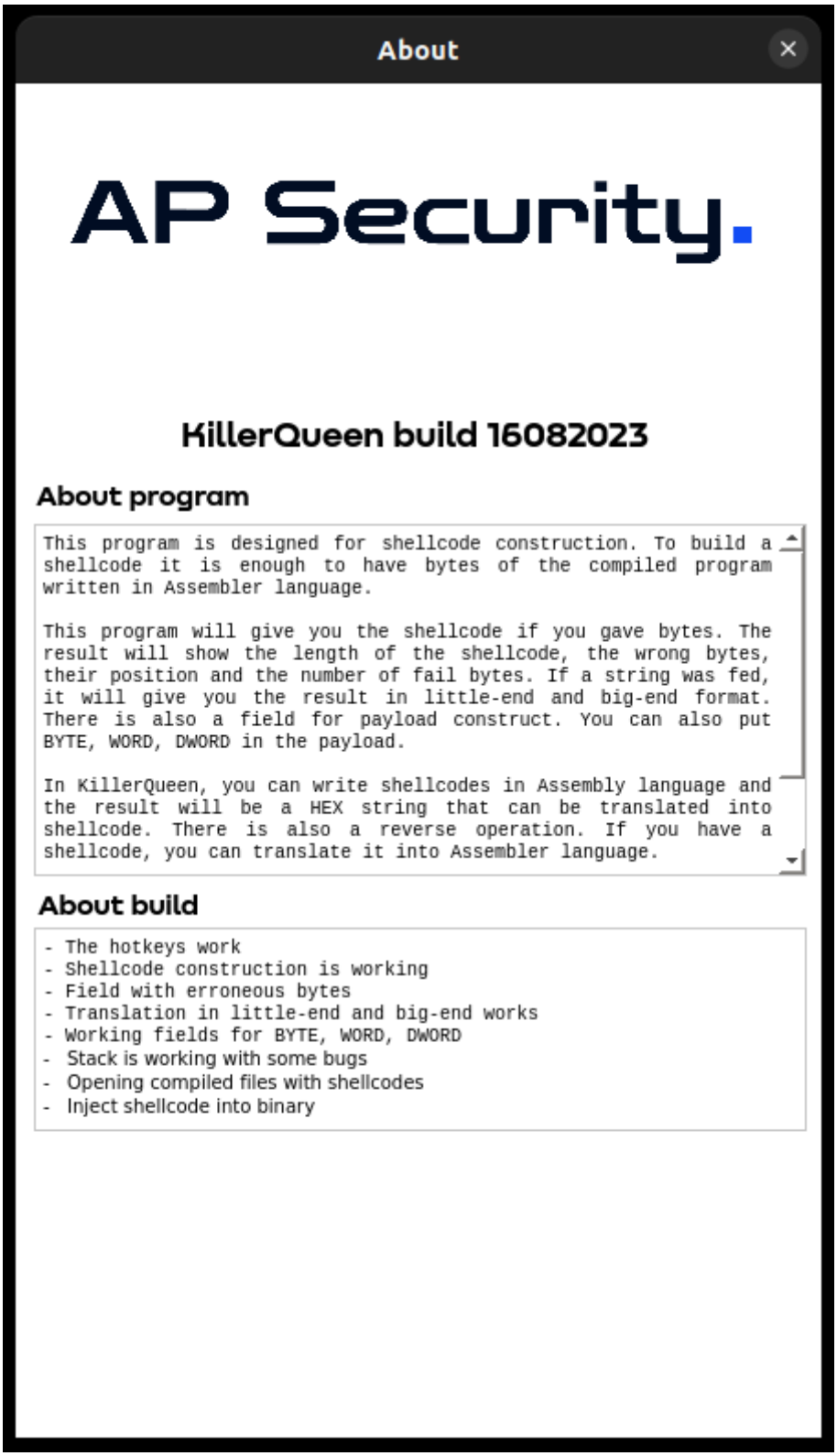


Вкладка `Linux templates` сожержит уже ряд готовых шеллкодов



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





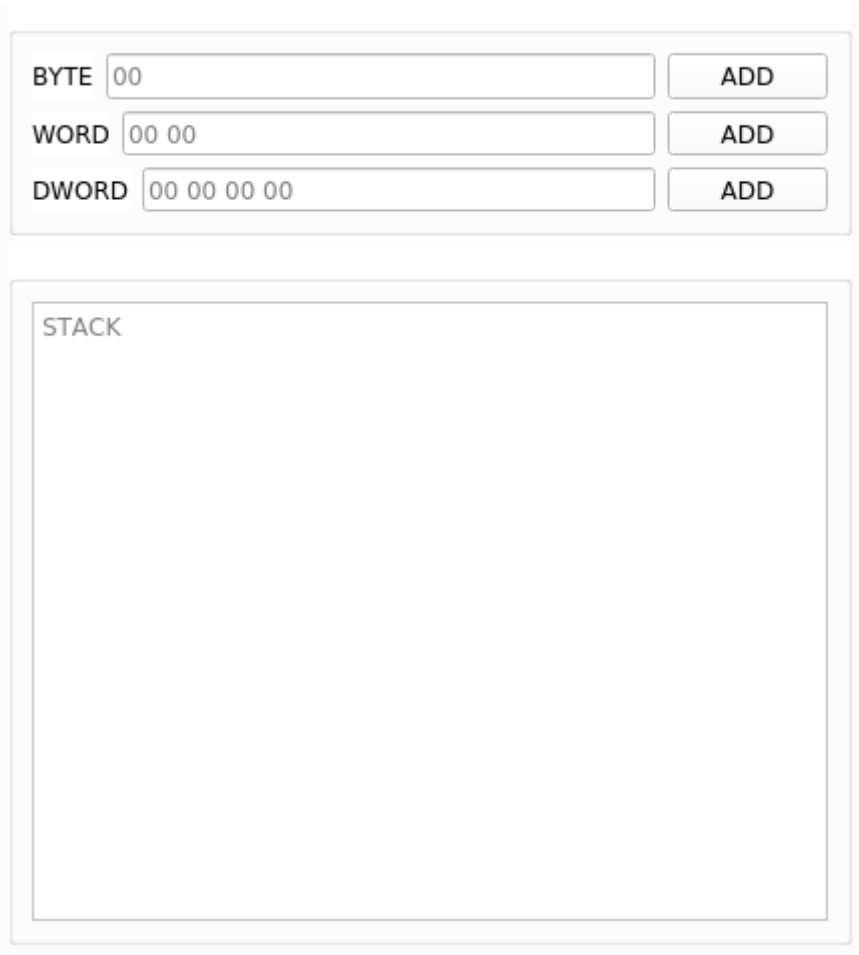
Перейдем к основному. На главном экране вверху можно увидеть поле `enter your bytes: 48 31 C0`

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

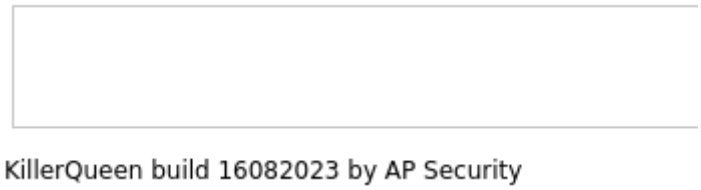
Как уже было сказано выше. Программа KillerQueen может посчитать размер шеллкода, это будет выведено в поле `SIZE`, а так же способно обнаружить плохие байты. Укзать в каком они месте, их количество и какие именно:

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

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



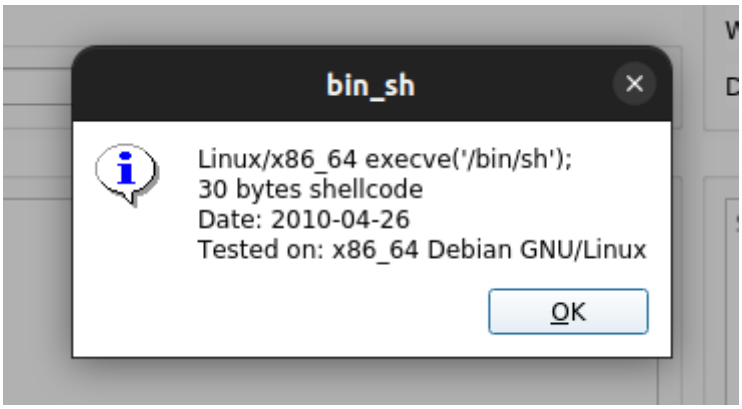
Для отслеживания результатов работы существует окно логирования, которое располагается в самом низу

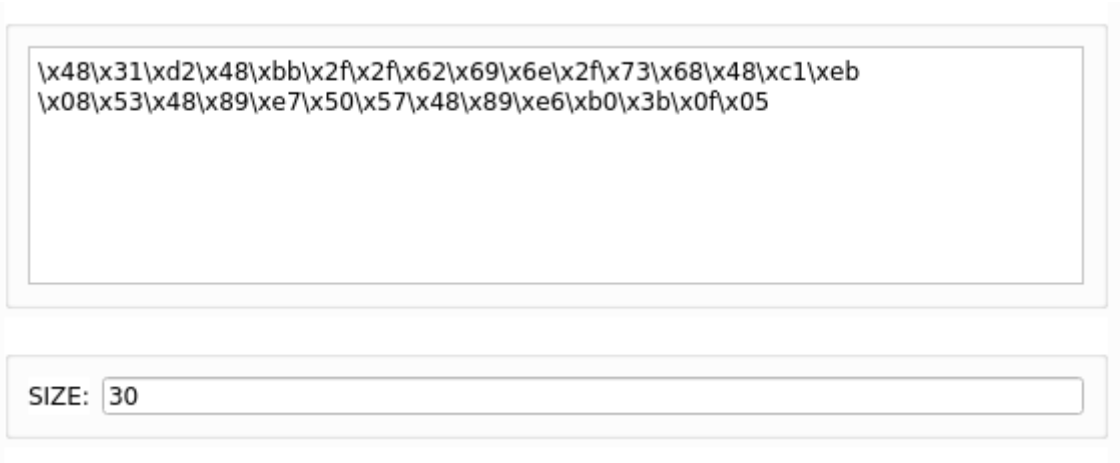


Теперь насталов время для примеров работы с программой.

Пример 1. Проверка рабочих шаблонов

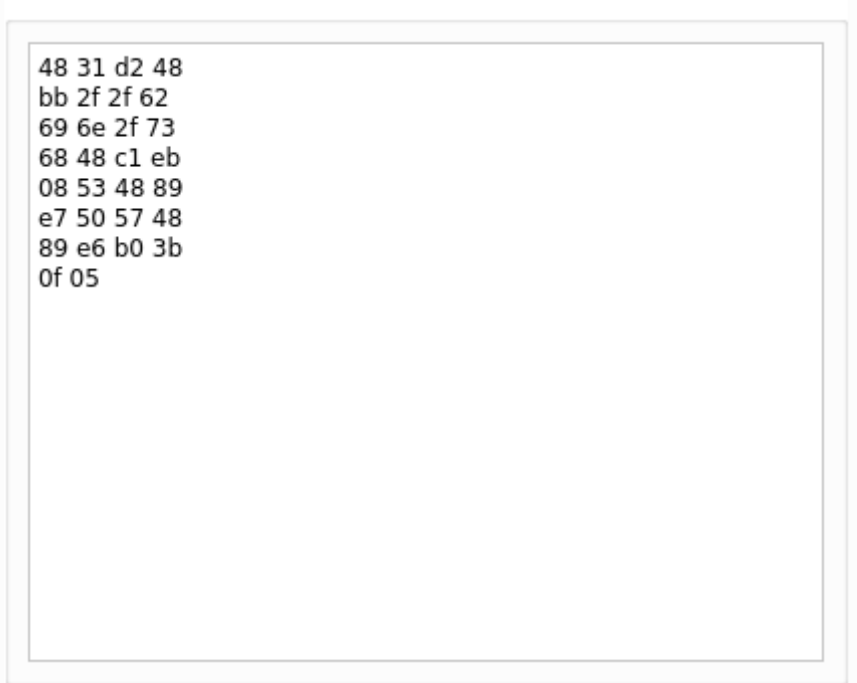
Проверка шаблонов заключается в отображении шеллкода в соответсвующем поле. Для этого необходимо нажать на `Linux templates -> bin_sh` . После этого высветится краткая справка о шеллкоде и результат





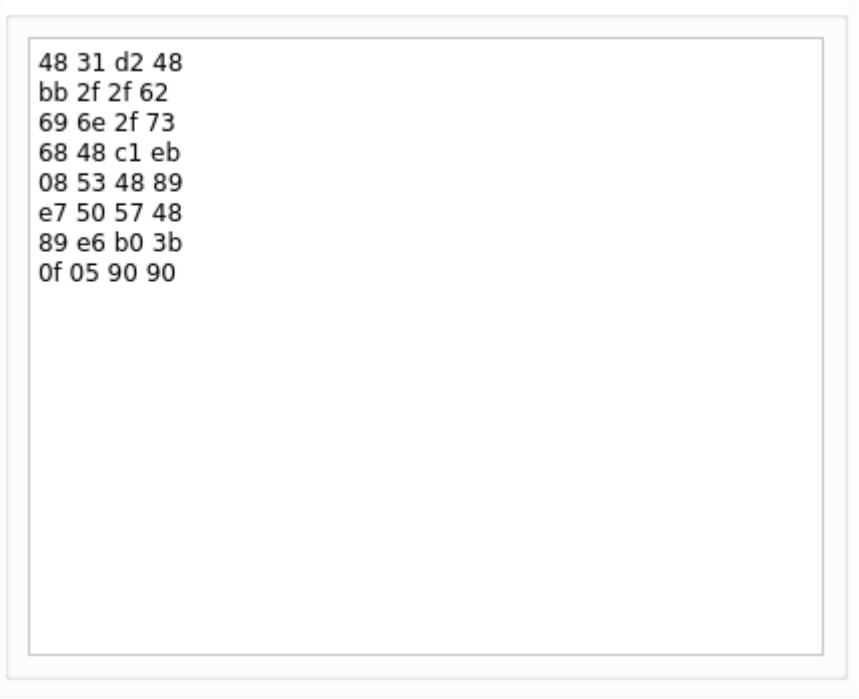
Можно смело копировать данную строку в эксплойт и запускать.

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



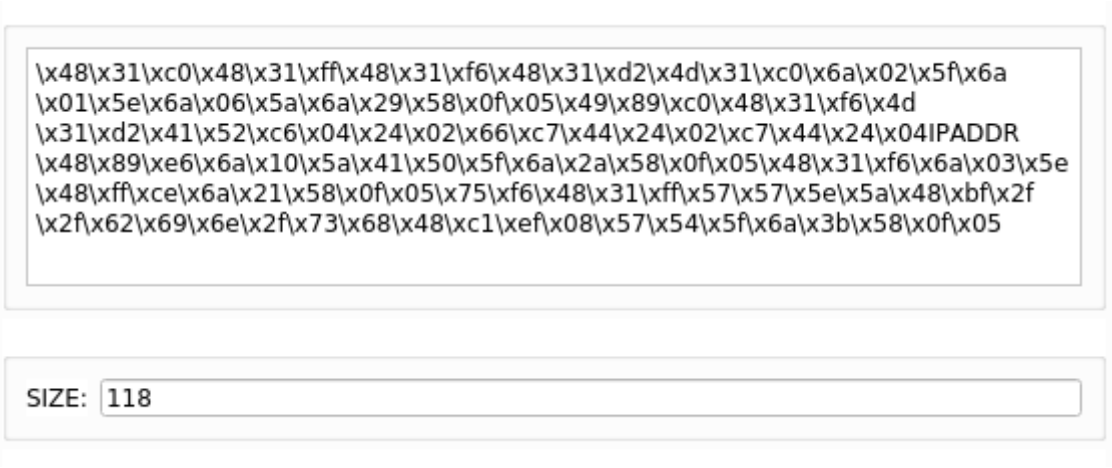
Добавим паддинг в виде двух байтов со значениями `\x90\x90`, что означает две последовательные инструкции ассемблера `NOP`

По нажатию клавиши `F3` 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



Теперь, посмотрим на окно ниже

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

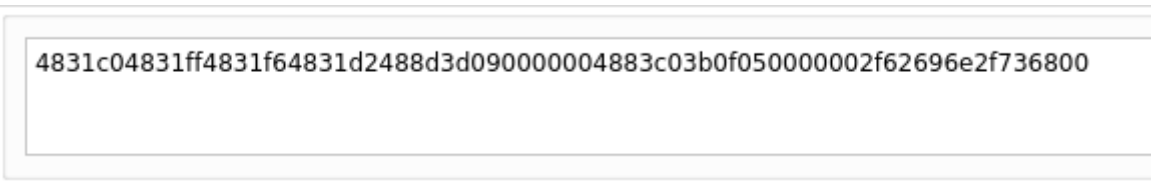
Пример 3. Генерация собственного шеллкода

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

```
BITS 64
global _start

section .text

_start:
```

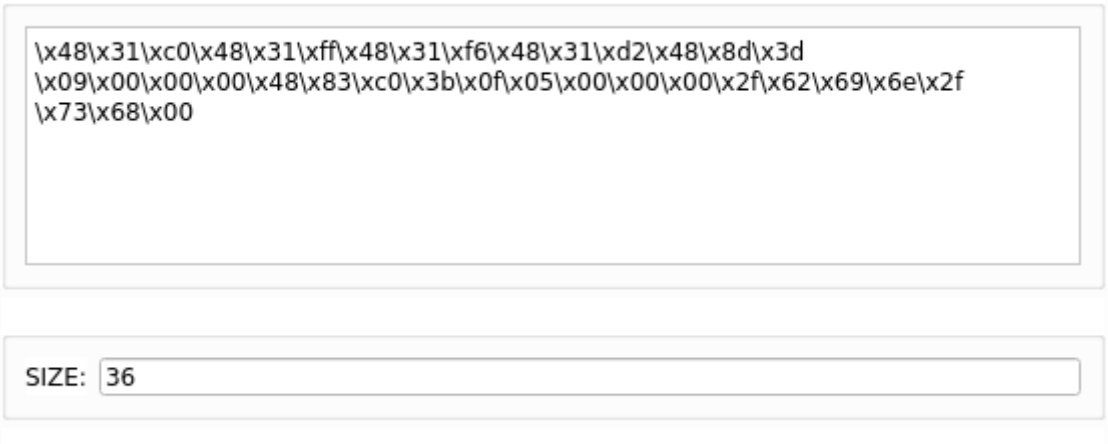
```
        xor rax, rax
        xor rdi, rdi
        xor rsi, rsi
        xor rdx, rdx

        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



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

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