Введение в PWN 0x3

Спикер: Павел Блинников

Руководитель группы исследования уязвимостей BI.ZONE
Капитан SPRUSH
Админ MEPhI CTF

@pturtle

Что мы узнали на прошлом занятии?

1. Больше о том, как работают бинари
2. Методы эксплуатации и “докручивания” уязвимостей
3. Больше о переполняшках и ROP
4. Как создавать контейнеры, чтобы они не падали через неделю

Что мы узнаем сегодня?

1. Разбор домашки на выбор
2. Как пользоваться IDA, чтобы решать пывны
3. Как эксплуатировать форматные строки руками и автоматически
4. Как хекать бинари с митигациями на максимум

Домашка DEMO

НИЯУ МИФИ BI.ZONE

IDA DEMO

НИЯУ МИФИ

BI.ZONE

Вспомним calling conventions

```c
void func(int count, ...) {
  va_list args;
  va_start(args, count);
  for (int i = 0; i < count; i++) {
    printf("%d\n", va_arg(args, int));
  }
  va_end(args);
}

int main() {
  func(9, 2, 3, 4, 5, 6, 7, 8, 9, 10);
}
```

Вспомним calling conventions

```
pturtle@pturtle:~ $ ./a.out
2
3
4
5
6
7
8
9
10
```

RDI
RSI
RDX
RCX
R8
R9
stack

printf

```
int printf(const char* format, ...);
```

По сути вывод данных, форматированный согласно аргументу format

Как выглядит правильное использование printf?

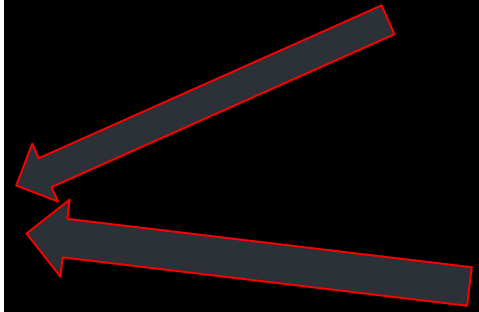

```c
#include <stdio.h>

int main() {
    char name[] = "Richard";
    char breed[] = "German shepherd";
    printf("My name is %s and I am %s\n", name, breed);
}
```

Format specifiers

specifier	Output	Example
d ***or*** i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

Format specifiers

specifier	Output	Example
d ***or*** i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%



Format string vulnerability


```
int main() {
    char buf[1024];
    read(0, buf, 1024);
    printf(buf);
}
```

Идея эксплуатации

1. Крадем поинтеры (и, если нужно, канарейку)
2. Записываемся в критичные структуры данных:
   a) GOT
   b) stack return address
   c) vtable, если доступно на запись
   d) etc, рассмотрено на предыдущей лекции
3. Исполняем свой код

Фишечька

```
int main() {
    char buf[] = "%3$s %2$s %1$s";
    printf(buf, "one", "two", "three");
}
```

```
pturtle@pturtle:~ $ ./a.out
three two one
```

Как украсть поинтеры?

%p – выводит значение как 64-битное hex

%d – выводит значение как 32-битное decimal

%x – выводит значение как 32-битное hex

%10$p – выведет 10 аргумент (4 значение со стека) как 64-битный hex

Фишечька №2

```
int main() {
    char buf[] = "%100s";
    printf(buf, "one");
}
```

Выведет one, дополненное слева 97 пробелами

Как записаться?

1. %n – записывает уже выведенное на экран количество байт по адресу, хранящемся в 1-м аргументе; количество хранится как 32-битное значение

2. %hn – то же, что и %n, но 16-битное значение

3. %hhn – то же, что и %n, но 8-битное значение

Как записаться?

1. Кладем на стек адрес, **по которому** хотим что-то записать
2. С помощью конструкции вида *%{value}c* выводим на экран то, **что** хотим записать
3. С помощью %n, %hn или %hhn записываем то, **что** хотим записать по тому, **куда** хотим записать

DEMO!

Стековая канарейка

Стековая канарейка (stack canary) – секретное значение на стеке, хранящееся перед RBP и адресом возврата.

Перед эпилогом функция проверит, что значение канарейки не изменено. Если все-таки изменено, то программа сразу улетает в SIGABRT, не давая нам провести эксплуатацию.

Комбинируем с переполняшкой

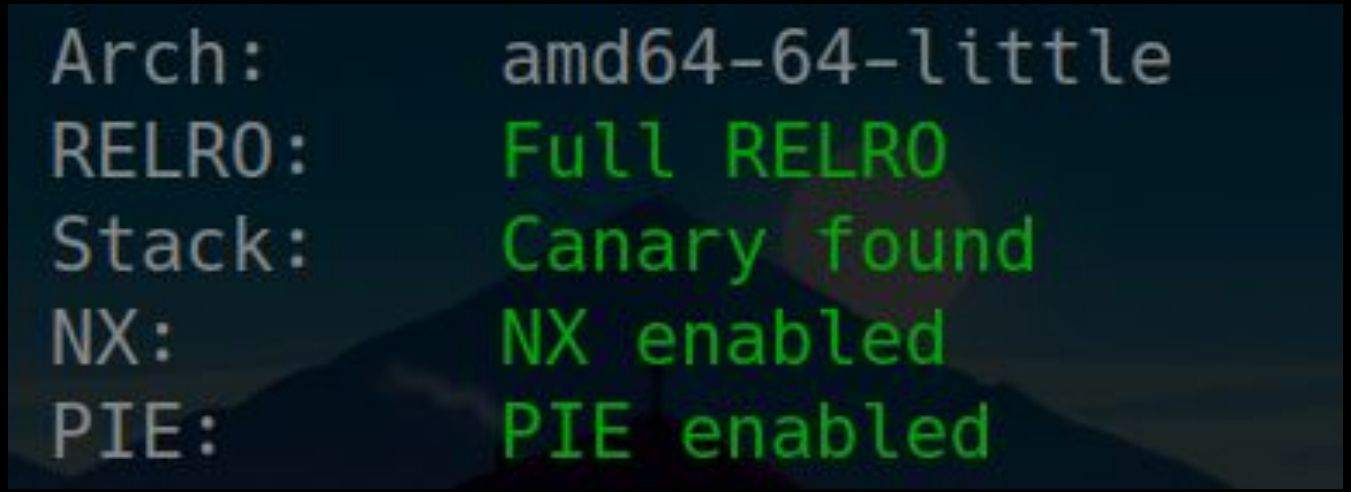
1. С помощью форматки крадем значение канарейки
2. Проводим обычную эксплуатацию переполнения буфера на стеке с помощью ROP
3. PWNED!

DEMO

НИЯУ МИФИ

BI.ZONE

Ultimate challenge



Ultimate challenge

1. Крадем адрес libc, PIE и стека
2. Переписываем **адрес возврата самой printf** на ret
3. Т.к. контролируемый нами буфер лежит сразу после адреса возврата printf, то мы выполним свой ROP :)

Ultimate challenge v.2

1. Что если наш буфер лежит не на стеке?
2. Тогда мы не можем контролировать поинтеры, по которым мы пишем, т.к. мы больше не контролируем аргументы printf

Ultimate challenge v.2

1. Что если наш буфер лежит не на стеке?
2. Тогда мы не можем контролировать поинтеры, по которым мы пишем, т.к. мы больше не контролируем аргументы printf
3. Ищем на стеке адрес, указывающий на стек
4. С помощью этого создаем свой адрес на стеке
5. Пишем по этому адресу, таким образом получаем arb write
6. Постепенно пишем ROP на стек, а потом меняем адрес возврата printf на ROP-гаджет, позволяющий нам прыгнуть на свою цепочку

НИЯУ МИФИ
BI.ZONE