Статья - Серединный вызов API функций

xmyriy

Перечитывал давеча Криса Касперски и его шедевральную (для своего времени) книгу ТиФХА и захотелось более детальнее пощупать такую технику, как Серединный вызов API функций. Желающих прочитать полную версию отсылаю к упомянутому труду, а тут приведу краткое содержание.

Точки останова (breakpoints) ставятся на начало API функции, поэтому их можно обмануть, если начать выполнение не с первой машинной команды. Автор (КК) предлагает «выдрать» из функции несколько байт и поместить их в собственный буфер, после чего совершить переход на оставшийся «хвост» ф-ции. КК проводил эксперименты под Win2k, где, по его подсчетам, не менее 75% функций начинается с классического пролога «push ebp/mov ebp,esp», который в машинном коде выглядит как 55h 8Bh ECh. Если бряк на функцию дебаггер уже поставил (CCh), просто выходим.

Пример ф-ции, копирующей пролог API-функции в локальный стековый буфер (c) КК

Code:

```
ZenWay(char *p, char *dst)
{
int f = 0; // кол-во скопированных в буфер байт
// ОДНОБАЙТОВЫЕ ШАБЛОНЫ
switch(*(unsigned char *)p)
{
case 0xCC:
printf("hello, hacker!\n");
exit(0);
break;

case 0x6A: // засылка в стек непосредственного значения
memcpy(dst, p, 2); f += 2;
break;
case 0x57: // PUSH EDI
*dst = 0x57; f += 1;
break;
default: f+=0;
}

// ОДНОСЛОВНЫЕ ШАБЛОНЫ
switch(*(WORD *)p)
{
case 0x8B55: // стандартный пролог
*((DWORD*)dst) = 0x00EC8B55; f += 3;
break;
case 0xD22B: // SUB EDX, EDX
*((WORD*)dst) = 0xD22B; f += 2;
break;
case 0x448B: // mov eax, [esp+xx]
case 0x74FF: // PUSH что-то-там
memcpy(dst, p, 4); f += 4;
break;
default:
f+=0;
}

// ШАБЛОН РАСПОЗНАН?
if (f==0) return 0; // нет ни одного совпадения
// ФОРМИРОВАНИЕ ПЕРЕХОДА НА ХВОСТ ФУНКЦИИ
strcpy((dst+f), "\xB8HACK\xFF\xE0");
*((DWORD *)(++dst+f)) = (DWORD) (p+f);
// УСПЕШНОЕ ЗАВЕРШЕНИЕ
return f;
}
```

Попробуем реализовать вызов MessageBox и посмотреть, что скажет IDA и x32dbg (я мучал tcc)

Code:

```
HINSTANCE hdll;
char ZMessageBoxA[MAX_CODE_SIZE];

int(WINAPI *XMessageBoxA)(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType);

hdll = LoadLibrary("USER32.DLL"); if (!hdll) return 0;

XMessageBoxA =(int (WINAPI*)(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType))
GetProcAddress(hdll, "MessageBoxA"); if (!XMessageBoxA) return 0;

// Копируем первые команды функции и корректируем указатели
if (ZenWay((char *) XMessageBoxA, (char *)ZMessageBoxA)!=0)
XMessageBoxA = (int (WINAPI*)(HWND hWnd, LPCSTR lpText, LPCSTR lpCaption, UINT uType))
ZMessageBoxA;

// Вызываем
XMessageBoxA(NULL,"Hello","Caption",MB_OK);
```

Приложение отработало, окошко посмотрели.

Заглянем в IDA, import section

Address	Ordinal	Name	Library
000000000040207C		printf	msvcrt
0000000000402080		exit	msvcrt
0000000000402084		memcpy	msvcrt
0000000000402088		strcpy	msvcrt
000000000040208C		__set_app_type	msvcrt
0000000000402090		_controlfp	msvcrt
0000000000402094		__argc	msvcrt
0000000000402098		__argv	msvcrt
000000000040209C		_environ	msvcrt
00000000004020A0		__getmainargs	msvcrt
00000000004020A4		_XcptFilter	msvcrt
00000000004020A8		_exit	msvcrt
00000000004020AC		_except_handler3	msvcrt
00000000004020B4		LoadLibraryA	kernel32
00000000004020B8		GetProcAddress	kernel32

Импорт MessageBoxA отсутствует.
Теперь посмотрим сам вызов MessageBoxA

```
.text:004011EB loc_4011EB:                             ; CODE XREF: sub_40116C+73↑j
.text:004011EB                 mov     eax, 0
.text:004011F0                 push    eax
.text:004011F1                 mov     eax, offset aCaption ; "Caption"
.text:004011F6                 push    eax
.text:004011F7                 mov     eax, offset aHello ; "Hello"
.text:004011FC                 push    eax
.text:004011FD                 mov     eax, 0
.text:00401202                 push    eax
.text:00401203                 mov     eax, [ebp+var_50]
.text:00401206                 call    eax
```


Бряк на вызов MessageBoxA, установленный в x32dbg не срабатывает по описанным выше причинам.

Техника старая, но вполне имеет место быть.