



二进制漏洞挖掘学习分享

CString of code audits lab
南京翰海源信息技术有限公司

Apr 16,17 2014 Shanghai

{xKungfoo 2014}



Who am i

- 袁世雄, 南京翰海源代码审计实验室成员, 网名CString.
- 主要从事
 - 软件安全测试
 - 二进制漏洞挖掘, 分析, 利用技术



漏洞的类型

逻辑型漏洞

基于数据处理的漏洞



基于数据处理漏洞挖掘过程

寻找攻击界面

定位相关处理点

分析对污染数据的处理



什么是攻击界面

用户可控的数据

网络

文件

共享内存，命名管道，邮槽

IoCode

ActiveX

自定义URL协议

Api hook中的参数

窗口消息

定位相关处理点

ida, od, windbg...

内存断点, 硬件断点

API断点

网络

- recv, recvfrom, WSAREcv, WSAREcvFrom

文件

- ReadFile

共享内存

- MapViewOfFile

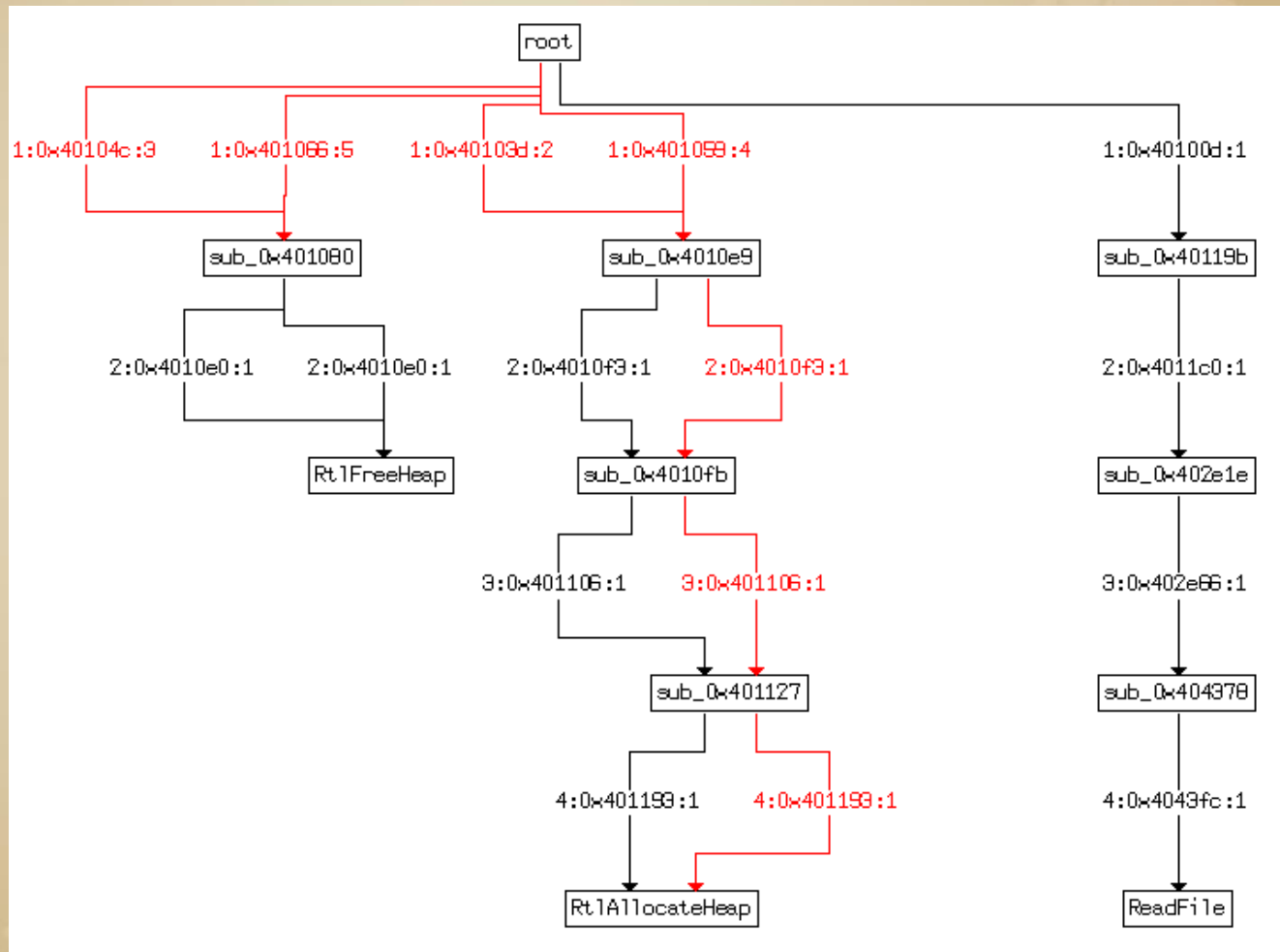
IoCode

- DeviceIoControl

.....



标记污染数据的传播



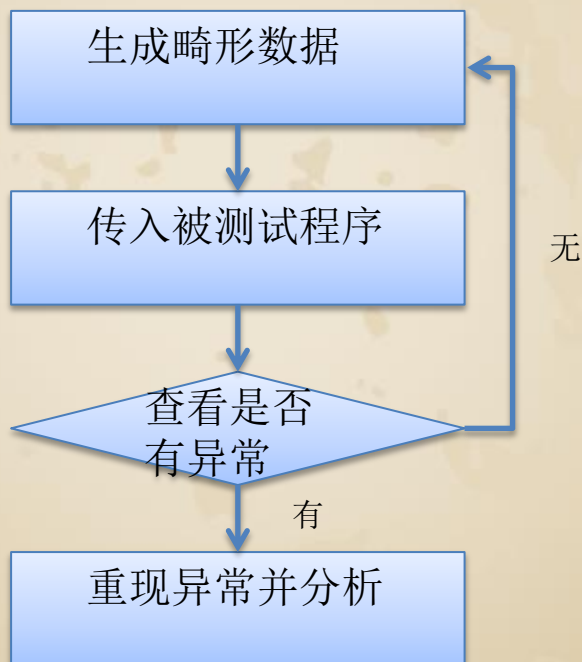


分析污染数据的处理

- 依靠经验查看处理代码
- fuzz

fuzz

• 一般流程





网络fuzz遇到的问题 1

- 需要满足特定的格式才能到达处理点

```
push    esi                ; buf
push    eax                ; s
mov     [esp+2Ch+fromlen], 10h
call    ds:recvfrom
mov     ebx, eax
test    ebx, ebx
jle     loc_1E65807
cmp     ebx, 10h
jbe     short loc_1E65730
mov     eax, [esi+6]
push    eax                ; netlong
call    ds:ntohl
cmp     eax, 10h
jbe     short loc_1E65730
cmp     eax, ebx
ja      short loc_1E65730
cmp     eax, 40000h
ja      short loc_1E65730
mov     eax, [esi]          ; esi->接收的缓冲区首地址
push    eax                ; netlong
call    ds:ntohl
cmp     eax, 20130329h      ; 检查
jnz     short loc_1E65730
```



网络fuzz遇到的问题 2

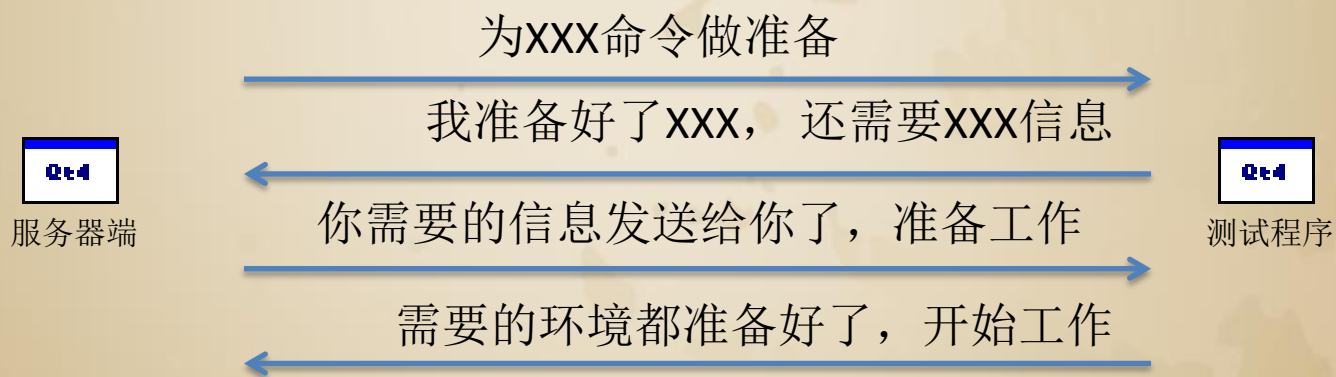
- 通讯过程加密





网络fuzz遇到的问题 3

- 需要多次交互



如果我们需要FUZZ
这些数据包怎么办?



提高fuzz的有效性

- 逆向分析网络数据包的结构

Diagram illustrating the structure of a network packet, showing offsets and corresponding data bytes. Three callouts provide specific requirements for certain fields:

- 这4字节需要等于 0x29031320 (This 4-byte field must equal 0x29031320)
- 这2字节需要等于数据包大小 (This 2-byte field must equal the packet size)
- 这4字节需要等于数据包长度-0x20 (This 4-byte field must equal packet length - 0x20)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	20	13	03	29	00	01	00	00	00	3E	00	00	00	1E	00	00
00000010	08	DD	A4	8C	81	01	10	56	18	02	20	A9	46	A8	06	00
00000020	12	1C	CA	B2	04	18	0A	16	00	00	10	23	12	5D	31	4A
00000030	EB	6D	30	91	00	00	00	17	0A	95	20	6A	33	F5		

- 按照结构生成畸形数据包



inline hook fuzz

- 通过逆向得到数据包解密函数

```
; int __stdcall M_Decrypt(const char *lpInBuf, int nInBufLen, char *lpOutBuf, int *lpOutBufLen)
M_Decrypt      proc near                                ; CODE XREF: sub_1E625B0+62↑p

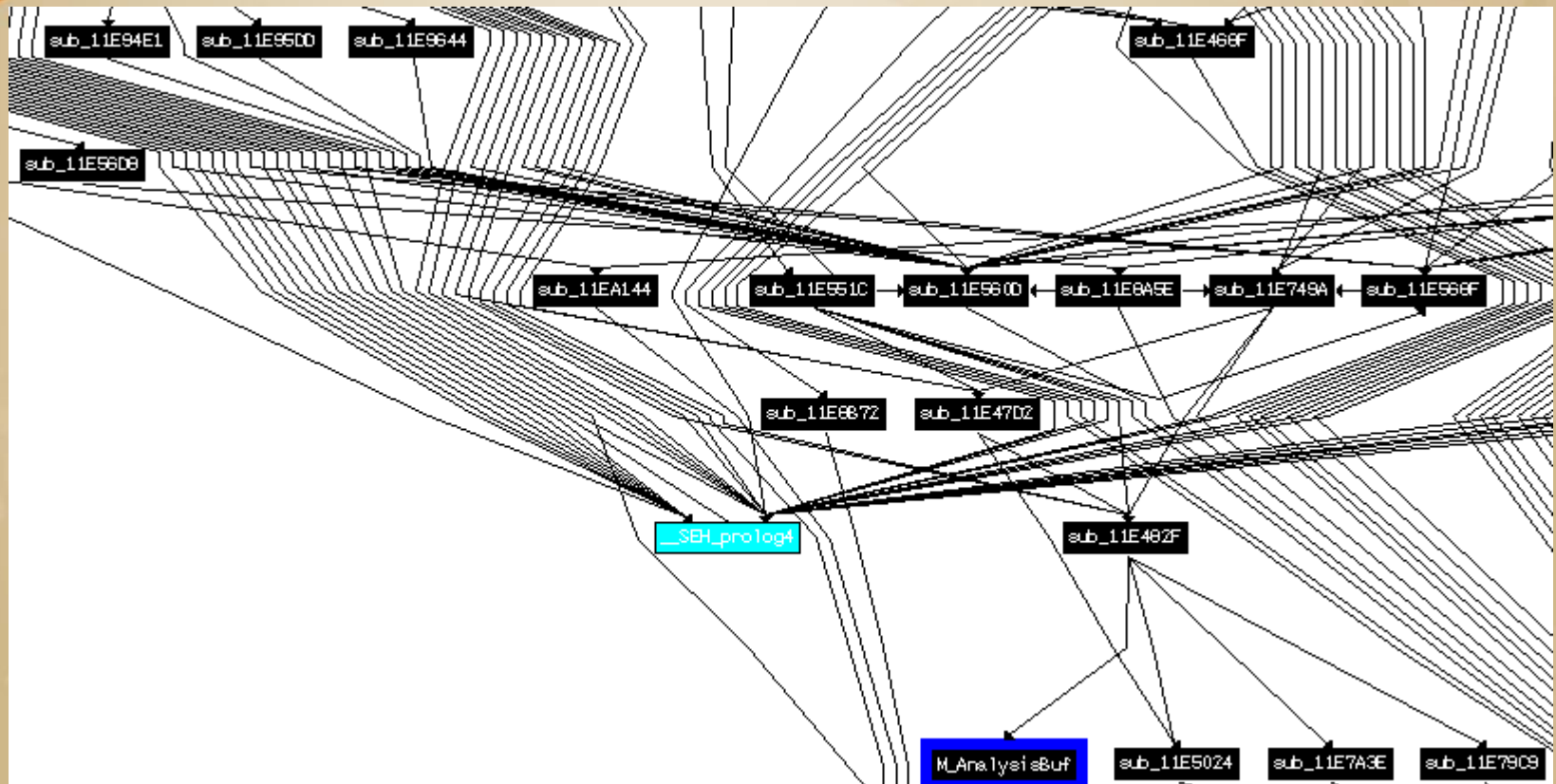
var_1C          = dword ptr -1Ch
var_18          = dword ptr -18h
var_14          = dword ptr -14h
var_10          = dword ptr -10h
var_C          = dword ptr -0Ch
var_4           = dword ptr -4
lpInBuf         = dword ptr 4
nInBufLen       = dword ptr 8
lpOutBuf        = dword ptr 0Ch
lpOutBufLen     = dword ptr 10h

                push    0FFFFFFFFh
                push    offset sub_1F030A8
                mov     eax, large fs:0
                push    eax
```



inline hook fuzz

- 逆向获知一个层次很深的数据处理点



有效性增加不止一点点

```
Microsoft Windows XP Service Pack 3 [Build 5.1.2600]
CPU: Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz
Type: EXCEPTION_ACCESS_VIOLATION
Error: Read address 0x00000000
Address: 7C9309F9

CallStack:
0x14E00000[18E] ntdll.dll: (05BDE9C8,00000000,0DCDFC84,00000000)
0x14E00000[16A] ntdll.dll: (003D0000,00000000,15690438,9706EE90)
0x14E00000[6FA] MSVC80.dll: (15690438,0DE70020,0AC4C518,000011C4)
0x14E00000[6F7] [REDACTED] d11: (0AC53220,0DE70020,0000039C,00000014)
0x14E00000[1F7] [REDACTED] d11: (0AC53220,0DE70020,0000039C,00000014)
0x14E00000[1E3] [REDACTED] (0000039C,0000001C,00000014,00000000)
0x14E00000[1DC] [REDACTED] d11: (0AC53220,00000027,00000003,00000003)
0x14E00000[6FA] [REDACTED] d11: (0A4D1618,00000004,00000000,00000000)
0x14E00000[6F7] [REDACTED] d11: (0A4D1618,00000004,00000000,00000000)
0x14E00000[6FB] [REDACTED] d11: (0AC4D2C0,17703B8E,0AC57ED4,80000001)
0x14E00000[6FA] [REDACTED] d11: (0AC57ED4,80000001,7C930208,FFFFFFFF)
0x14E00000[6F7] [REDACTED] : (0AC4D2C0,9706EDD0,7C930208,FFFFFFFF)
0x14E00000[3C8] EAX=00000000 [REDACTED] MSVC80.dll: (FFFFFFFF,7C80B713,0AC0E2F8,34303738)
0x14E00000[37C] ESI=0E24F [REDACTED] MSVC80.dll: (0AC58070,7C930208,FFFFFFFF,0AC58070)
0x14E00000[2428B] [REDACTED] MSVC80.dll: (0AC0E2F8,34303738,35393031,0AC0E2F8)
0x14E00000[240AE] [REDACTED] d11: (0AC0E15C,80000001,34303738,35393031)
0x17700000[3B8E] [REDACTED] : (0AC079E0,7AF7377A,34303738,35393031)
0x78130000[29BB] MSVC80.dll: (35393031,7C80B713,0AC0E2F8,34303738)
0x78130000[2A47] MSVC80.dll: (0AC0E2F8,34303738,35393031,0AC0E2F8)
8,00000004,00000000,00000000)
0,17703B8E,0AFB045C,80000001)
C,80000001,7C930208,FFFFFFFF)
87B8CEA,7C930208,FFFFFFFF)
```

进一步思考

- 文件格式fuzz
- 功能模块IAT hook
 - 减少判断逻辑，提高程序运行效率



工欲善其事，必先利其器

- 需要解决问题
 - 需要hook的点每次都变化
 - 灵活的修改数据
 - 方便的逻辑控制

解决思路与方法

- 通过配置文件获取需要hook的信息
 - 模块名称
 - 导出函数名称或者偏移
 - 函数的参数个数
- 调用lua脚本
 - 脚本通过C扩展修改数据

```
-- hook ReadFile

function call_prev_func(hFile, lpBuf, nReadFile, lpReadFize, lpv)
    -- do something
end

function call_back_func(nRet, hFile, lpBuf, nReadFile, lpReadFize, lpv)
    -- do something
    return nRet;
end
```



动态生成的跳转函数

00BB0000	54	push	esp	保存esp
00BB0001	55	push	ebp	保存ebp
00BB0002	50	push	eax	保存寄存器
00BB0003	53	push	ebx	
00BB0004	51	push	ecx	
00BB0005	52	push	edx	
00BB0006	56	push	esi	
00BB0007	57	push	edi	
00BB0008	68 14000000	push	0x14	参数格式*4
00BB000D	68 2600BB00	push	0xBB0026	跳转回ReadFile的地址
00BB0012	E8 47D22A57	call	FuzzD11.57E5D25E	
00BB0017	83C4 08	add	esp, 0x8	
00BB001A	5F	pop	edi	
00BB001B	5E	pop	esi	
00BB001C	5A	pop	edx	
00BB001D	59	pop	ecx	
00BB001E	5B	pop	ebx	
00BB001F	58	pop	eax	
00BB0020	83C4 08	add	esp, 0x8	
00BB0023	C2 1400	retn	0x14	平衡堆栈
00BB0026	6A 0C	push	0xC	
00BB0028	68 F03ECF75	push	0x75CF3EF0	
00BB002D	- E9 683E1475	jmp	kernel32.75CF3E9A	



中转函数定义

```
void CLuaManage::Transit(PVOID lpSysAddr, DWORD dwParamSize, REG_INFO RegInfo)
```

```
typedef struct _REG_INFO  
{  
    DWORD dwEDI;  
    DWORD dwESI;  
    DWORD dwEDX;  
    DWORD dwECX;  
    DWORD dwEBX;  
    DWORD dwEAX;  
    DWORD dwEBP;  
    DWORD dwESP;  
} REG_INFO, *LPREG_INFO;
```

中转函数调用call_prev_func

```
//获得参数的个数
int nParamCount = dwParamSize / sizeof(void*);
_asm
{
    mov ecx, dwParamSize;
    //分配参数大小的栈空间
    sub esp, ecx;
    mov edi, esp;
    mov esi, RegInfo.dwESP;
    /*
        esp指向返回地址
        所以这里需要+4, 获得参数的首地址
    */
    add esi, 4;
    //将参数拷贝到栈空间
    rep movs byte ptr es:[edi], byte ptr [esi];
    push nParamCount;
    /*
        void CLuaManage::PrevCall(int nParamCount, ...)
        C调用方式, 第一个参数是参数个数
    */
    call CLuaManage::PrevCall;
    /*
        C调用的堆栈平衡
    */
    add esp, 4;
    add esp, dwParamSize;
}
```




调用脚本call_prev_func

```
void CLuaManage::PrevCall(int nParamCount, ...)
{
    lua_State* lpState = luaL_newstate();
    if (NULL != lpState)
    {
        luaL_openlibs(lpState);
        if (0 == luaL_loadbuffer(lpState, st_lpProjectBuf, st_nProjecSize, NULL))
        {
            lua_pcall(lpState, 0, 0, 0);
            CKerLuaFunc::DataLuaFuncInitialize(lpState);
            CLuabindFunc::LuaFuncInitialize(lpState);
            va_list ap;
            va_start(ap, nParamCount);
            lua_getglobal(lpState, "call_prev_func");
            for (int i = 0; i < nParamCount; i++)
            {
                int nParam = va_arg(ap, int);
                lua_pushinteger(lpState, nParam);
            }
            if (0 != lua_pcall(lpState, nParamCount, 0, 0))
            {
                string strError = "Lua failed: ";
                strError += lua_tostring(lpState, -1);
                CLuabindFunc::L_SendMsg(strError, true);
            }
            va_end(ap);
            lua_close(lpState);
        }
    }
}
```

调用原始的API

```
_asm
{
    mov ecx, dwParamSize;
    sub esp, ecx;
    mov edi, esp;
    mov esi, RegInfo.dwESP;
    add esi, 4;
    rep movs byte ptr es:[edi], byte ptr [esi];
    //恢复寄存器环境
    mov eax, RegInfo.dwEAX;
    mov ebx, RegInfo.dwEBX;
    mov ecx, RegInfo.dwECX;
    mov edx, RegInfo.dwEDX;
    mov edi, RegInfo.dwEDI;
    mov esi, RegInfo.dwESI;
    //调用原始的函数
    call lpSysAddr;
    //将调用后的寄存器保存
    mov RegInfo.dwEAX, eax;
    mov RegInfo.dwEBX, ebx;
    mov RegInfo.dwECX, ecx;
    mov RegInfo.dwEDX, edx;
    mov RegInfo.dwEDI, edi;
    mov RegInfo.dwESI, esi;
}
```

中转函数调用call_back_func

```
_asm
{
    mov ecx, dwParamSize;
    sub esp, ecx;
    mov edi, esp;
    mov esi, RegInfo.dwESP;
    add esi, 4;
    rep movs byte ptr es:[edi], byte ptr [esi];
    push nParamCount;
    push RegInfo.dwEAX;
    /*
        int CLuaManage::BackCall(int nRet, int nParamCount, ...)
        第一个参数是返回值
    */
    call CLuaManage::BackCall;
    //保存lua脚本的返回值
    mov RegInfo.dwEAX, eax;
    //平衡堆栈
    add esp, 8;
    add esp, dwParamSize;
}
```



调用脚本call_back_func

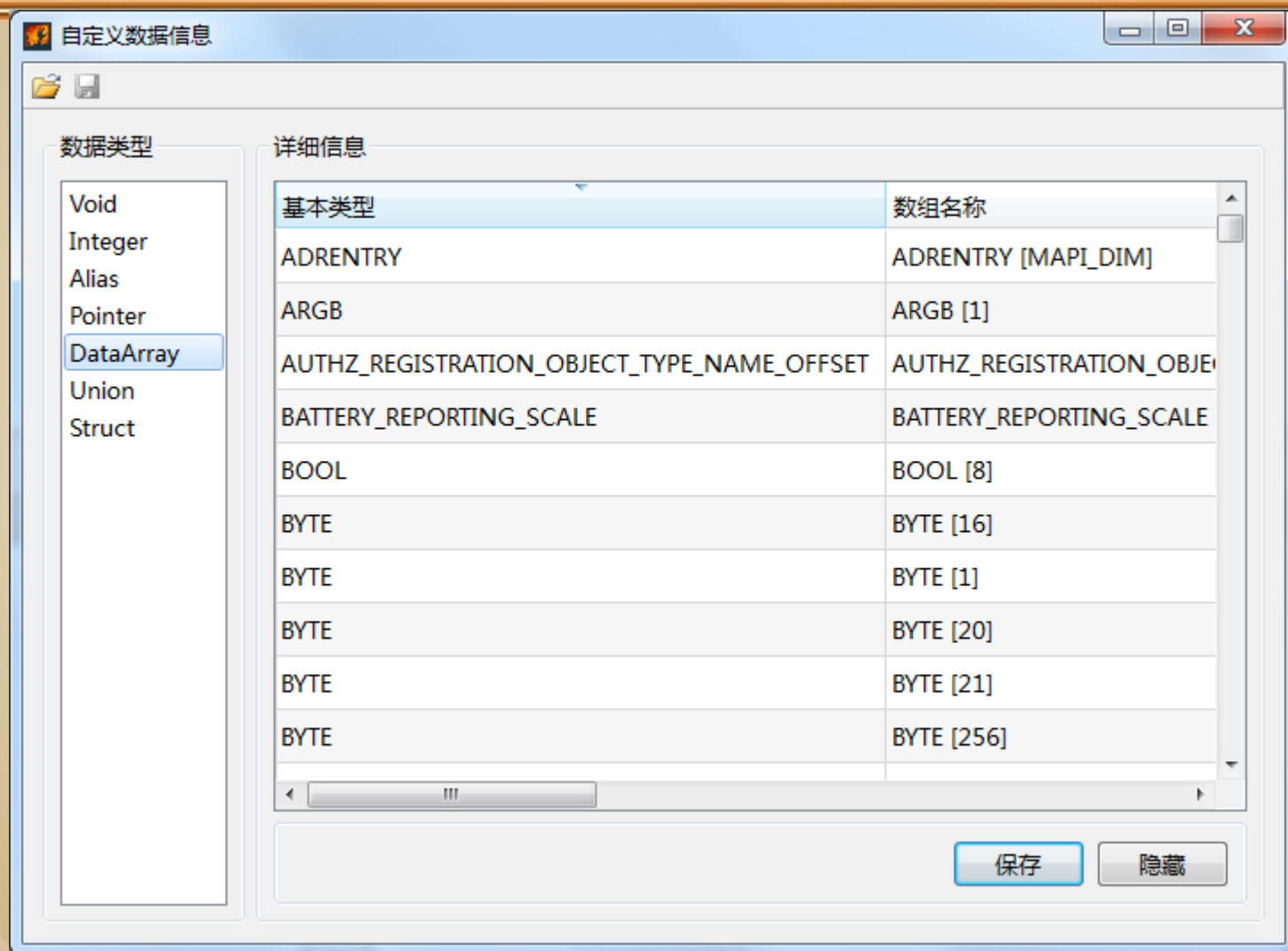
```
int CLuaManage::BackCall(int nRet, int nParamCount, ...)
{
    lua_State* lpState = luaL_newstate();
    if (NULL != lpState)
    {
        luaL_openlibs(lpState);
        if (0 == luaL_loadbuffer(lpState, st_lpProjectBuf, st_nProjecSize, NULL))
        {
            lua_pcall(lpState, 0, 0, 0);
            CKerLuaFunc::DataLuaFuncInitialize(lpState);
            CLuabindFunc::LuaFuncInitialize(lpState);
            va_list ap;
            va_start(ap, nParamCount);
            lua_getglobal(lpState, "call_back_func");
            lua_pushinteger(lpState, nRet);
            for (int i = 0; i < nParamCount; i++)
            {
                int nParam = va_arg(ap, int);
                lua_pushinteger(lpState, nParam);
            }
            if (0 != lua_pcall(lpState, nParamCount + 1, 1, 0))
            {
                string strError = "Lua failed: ";
                strError += lua_tostring(lpState, -1);
                CLuabindFunc::L_SendMsg(strError, true);
            }
            else
            {
                nRet = (int)lua_tointeger(lpState, -1);
                lua_pop(lpState, 1);
            }
            va_end(ap);
            lua_close(lpState);
        }
    }
    return nRet;
}
```



实现方便的逻辑控制

- 对lua常用的函数通过C扩展
 - CreateProcess
 - 读写管道
 - 注册表操作
 - 修改内存数据
 -

常用和自定义数据结构



小结

- 通过自定义结构体生成数据
- 思路



分享我fuzz到的那些洞



某公司游戏

- 寻找攻击点
 - 游戏录像文件
- 下载正常的录像文件
- 使用边界数据单字节替换正常文件
 - 0x00
 - 0x40
 - 0x7f
 - 0x80
 - 0xff



lua 脚本


```
function fuzz_main()
    while true do
        -- create_fuzz_file 生成畸形文件，并返回文件路径
        filepath = create_fuzz_file();
        -- C扩展 ShellExecute, 打开文件
        windows.ShellExecute(filepath, true);
        -- C扩展 Sleep 等待3秒
        windows.Sleep(1000 * 3);
        -- C扩展 FindWindowA 查看是否有自定义异常信息的窗口
        crash_hwnd = windows.FindWindowA(nil, "—");
        if (nil ~= crash_hwnd) then
            -- 有的话结束自定义异常信息的窗口
            -- 保存当前的文件
            windows.TerminateProcess(crash_hwnd, 0);
            save_poc(filepath);
        end
    end
end
```

挖到的漏洞

• 栈溢出

```
text:0E3368DF ; -----  
text:0E3368DF ;  
text:0E3368DF loc_E3368DF: ; CODE XREF: M_GetBufByMem+22↑j  
text:0E3368DF      push     edi  
text:0E3368E0      lea      edi, [edx+ebx] ; 通过当前文件偏移获取地址  
text:0E3368E3      cmp      edi, esi  
text:0E3368E5      mov      ecx, ebx  
text:0E3368E7      jl       short loc_E3368ED ; 这里有错误, 有符号比较  
text:0E3368E7      ; POC中是 0x7FFFFFFF0, 加上当前文件偏移 0x23  
text:0E3368E7      ; 等于 0x80000013, 能绕过  
text:0E3368E9      sub      esi, edx ; 如果超过, 就只读剩下的大小  
text:0E3368EB      mov      ecx, esi  
text:0E3368ED loc_E3368ED: ; CODE XREF: M_GetBufByMem+37↑j  
text:0E3368ED      mov      esi, [eax]  
text:0E3368EF      mov      edi, [ebp+1pBuf]  
text:0E3368F2      add      esi, edx  
text:0E3368F4      mov      edx, ecx  
text:0E3368F6      shr      ecx, 2  
text:0E3368F9      rep movsd ; 这里栈溢出了  
text:0E3368FB
```


越界读取



```
.text:0E3DD3C3      mov     [ebp+@CurrBuf], esi
.text:0E3DD3C6      call    M_GetBufByMem    ; 读出循环计数
.text:0E3DD3CB      mov     eax, [ebp+@CurrBuf]
.text:0E3DD3CE      mov     [ebp+nSize], esi
.text:0E3DD3D1      cmp     eax, esi
.text:0E3DD3D3      mov     [ebp+var_18], esi
.text:0E3DD3D6      jbe     loc_E3DD4E7
.text:0E3DD3DC      Next:
.text:0E3DD3DC      ; CODE XREF: sub_E3DD1C0+321↓j
.text:0E3DD3DC      lea     eax, [ebp+nSize]
.text:0E3DD3DF      push    1                ; int
.text:0E3DD3E1      push    eax               ; void *
.text:0E3DD3E2      lea     ecx, [ebp+@MemInfo]
.text:0E3DD3E5      call    M_GetBufByMem
.text:0E3DD3EA      cmp     [ebp+nSize], 1
.text:0E3DD3EE      setz    al
.text:0E3DD3F1      test    eax, eax
```

```
.text:0E3DD4D5      loc_E3DD4D5:
.text:0E3DD4D5      ; CODE XREF: sub_E3DD1C0+233↑j
.text:0E3DD4D5      mov     eax, [ebp+var_18]
.text:0E3DD4D8      mov     ecx, [ebp+@CurrBuf] ; 循环
.text:0E3DD4DB      inc     eax
.text:0E3DD4DC      cmp     eax, ecx
.text:0E3DD4DE      mov     [ebp+var_18], eax
.text:0E3DD4E1      jb      Next
```


本地DoS



```
.text:0C7CD992      jz         loc_C7CDFC1
.text:0C7CD998      mov         edi, 4
.text:0C7CD99D      lea         edx, [ebp+nSize]
.text:0C7CD9A0      push        edi             ; int
.text:0C7CD9A1      push        edx             ; void *
.text:0C7CD9A2      lea         ecx, [ebp+@MemInfo]
.text:0C7CD9A5      call        M_GetBufByMem    ; 从POC文件0X33A处读取4字节
.text:0C7CD9AA      mov         eax, [ebp+@Unknow_14]
.text:0C7CD9AD      mov         dword ptr [eax], 0
.text:0C7CD9B3      mov         ecx, [ebp+@Unknow_14]
.text:0C7CD9B6      mov         dword ptr [ecx+4], 0
.text:0C7CD9BD      mov         edx, [ebp+@Unknow_14]
.text:0C7CD9C0      mov         eax, [ebp+nSize]
.text:0C7CD9C3      mov         [edx+0Ch], eax
.text:0C7CD9C6      mov         ecx, [ebp+@Unknow_14]
.text:0C7CD9C9      mov         edx, [ecx+0Ch]   ; 下面的new没有检查大小
                                ; POC是0x80000000, 导致new出错
.text:0C7CD9C9
.text:0C7CD9CC      push        edx             ; unsigned int
.text:0C7CD9CD      call        ??2@YAPAXIQZ     ; operator new(uint)
.text:0C7CD9D2      mov         ecx, [ebp+@Unknow_14]
```



某公司聊天软件

- 寻找攻击点
 - 语言聊天
- 在加密函数前hook,修改发送的数据包




fuzz出来的漏洞

- 错误数据包的格式

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	C7	00	00	80	C7	00	00	C0	00	00	00	00	9D	00	00	00	Ç IÇ À
00000010	5B	57	04	00	C8	00	9C	AD	E6	30	01	00	00	00	00	00	[W È I-æ0
00000020	00	00	97	5A	30	D3	00	80	00	01	C3	00	00	00	00	00	I Z0Ó I Ã
00000030	00	00	05	00	00	00	32	00	01	A8	00	0E	00	00	00	00	2 "
00000040	00	00	01	01	9E	44	2F	BC	2F	06	55	E8	7E	02	00	29	ID/¼/ Uè~)
00000050	00	00	00	24	01	39	A0	AA	AA	A8	41	F0	1F	09	63	39	\$ 9 æ" Að c9
00000060	62	09	AC	AF	33	80	8C	0D	02	86	40	87	8D	D6	10	2F	b -3 I@I Ö /
00000070	95	E1	09	2D	C7	2E	94	77	1A	CD	21	B0	29	00	00	00	Iá -Ç.Iw Í!°)
00000080	24	09	3B	70	BB	FF	EF	98	41	5A	30	6D	65	4F	18	64	\$;p»yiIAZ0me0 d
00000090	B8	24	E8	40	10	01	06	01	70	D4	C3	03	47	20	25	82	, \$è@ pÔÃ G %I
000000A0	BB	31	74	25	7C	63	18	61	50	1B	00	00	00	1D	00	00	»1t% c aP
000000B0	00	C5	00	00	00	DC	01	00	00	4E	01	00	00	10	00	00	Å Ü N
000000C0	00	41	00	00	00	26	00	00	00	24	00	0C	11				A & \$

崩溃信息



```
(fa8.360): Access violation - code c0000005 (first chance)↵
First chance exceptions are reported before any exception handling.↵
This exception may be expected and handled.↵
eax=044968f2 ebx=00300000 ecx=000a1a3d edx=00000000 esi=0420fffe edi=0704972c↵
eip=7855ae7a esp=01a1db24 ebp=01a1db2c iopl=0         nv up ei pl nz ac po nc↵
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00210212↵
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for
C:\WINDOWS\WinSxS\x86_Microsoft.VC90.CRT_1fc8b3b9a1e18e3b_9.0.30729.6161_x-ww_31a
54e43\MSVCR90.dll -  ↵
MSVCR90!memcpy+0x5a:↵
7855ae7a f3a5             rep movs dword ptr es:[edi],dword ptr [esi]↵
0:005> dd esi↵
0420fffe  ????????? ????????? ????????? ?????????↵
0421000e  ????????? ????????? ????????? ?????????↵
0421001e  ????????? ????????? ????????? ?????????↵
0421002e  ????????? ????????? ????????? ?????????
```



Q&A