



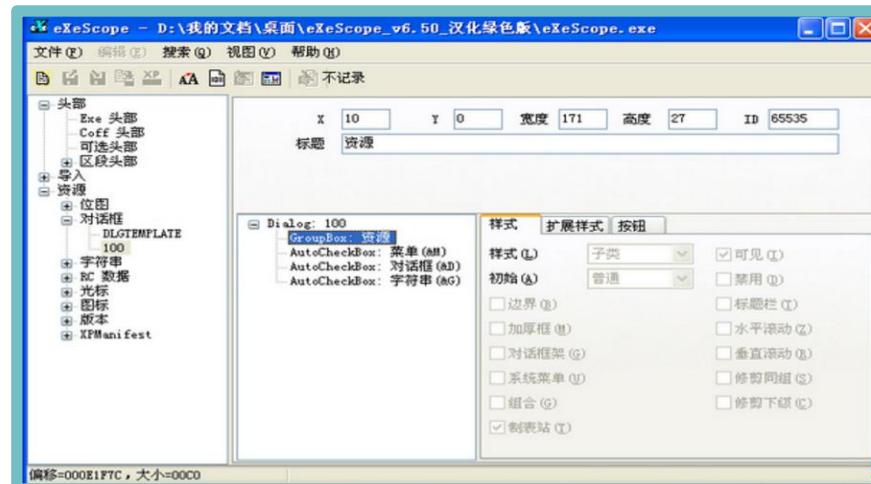
软件安全复习大纲

南京邮电大学计算机学院信息安全系

复习要点

- 围绕复习大纲，结合课程PPT进行知识点扩展复习。
- 掌握二进制程序的调试分析流程，遇到实际问题时怎么解决。
- 深入理解课程中各种CrackMe程序的逆向破解过程，能够逐语句进行分析。
- 熟练掌握各种分析工具的应用场景及快捷键，如：IDA、OD、WinDbg等。
- 掌握常见汇编指令的含义和用法，如：数据传送、比较、移位、跳转等。
- 熟悉PE文件结构，理解PE文件的加载过程。
- 熟悉软件加壳和脱壳基本原理和过程，如：基本步骤、工具等。
- 掌握软件反调试的原理类型及各种具体反调试技术。
- 掌握软件漏洞概念以及缓冲区溢出漏洞的原理。

相关工具 - 二进制查看 - exeScope



文件结构框架

文件结构解析

The screenshot shows the exeScope interface with the 'Header' tab selected for the file 'C:\Downloads\exeSc641\exeBat.exe'. The left pane shows the header structure with sections like 'Exe Header', 'Coff Header', 'Optional Header', 'Section Header', 'Import', and 'Resource'. The 'Coff Header' section is currently selected. The right pane displays a table of header fields with their addresses, values, and meanings.

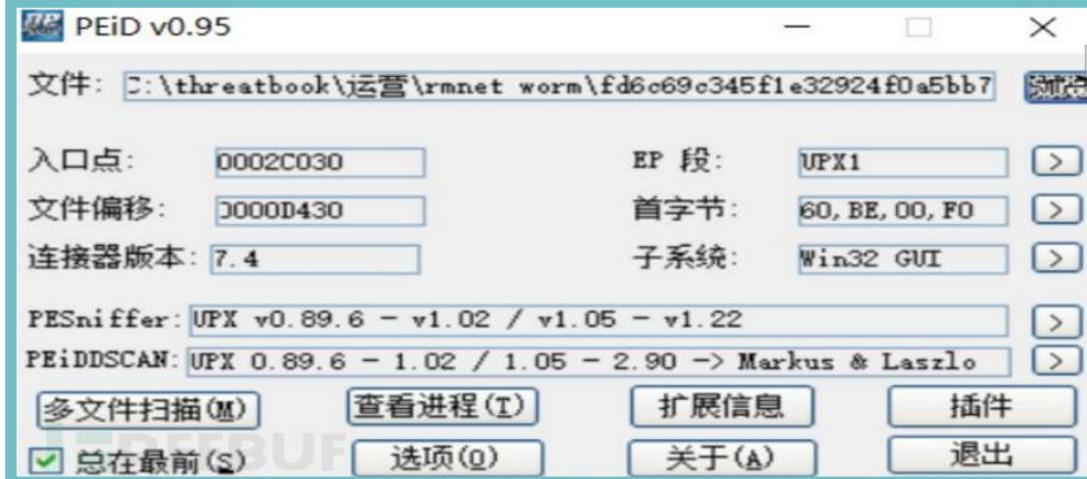
Address	Value	Meaning
00000100	00004550	Signature: PE
00000104	014C	Machine: 014C=I386
00000106	0008	Number of Sections
00000108	2A42E19	Time/Date Stamp
0000010C	00000000	Pointer to Symbol Table
00000110	00000000	Number of Symbols
00000114	00E0	Optional Header Size
00000116	818E	Characteristics

相关工具 - 二进制查壳 - PEID/LoadPE

识别zprotect
(加密壳)



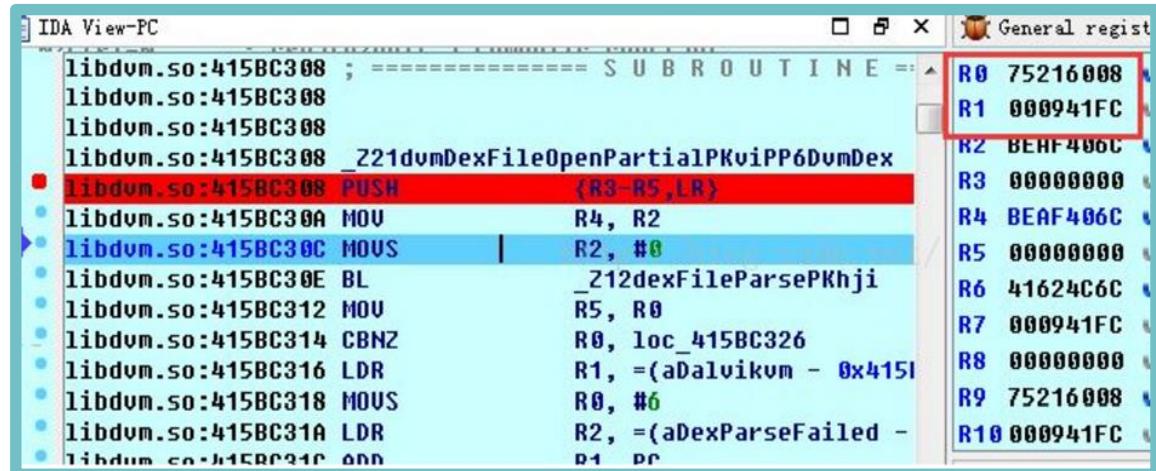
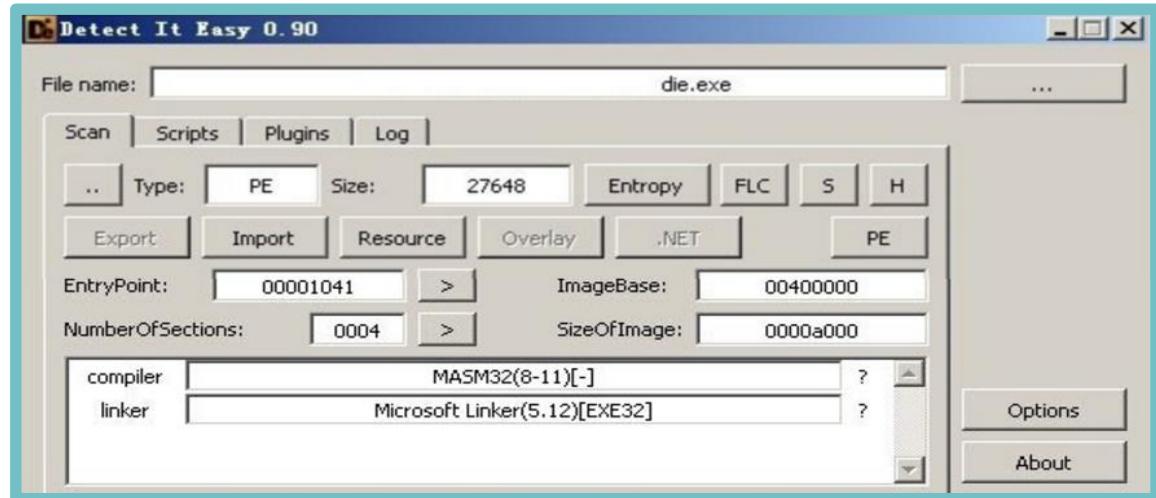
识别UPX
(压缩壳)



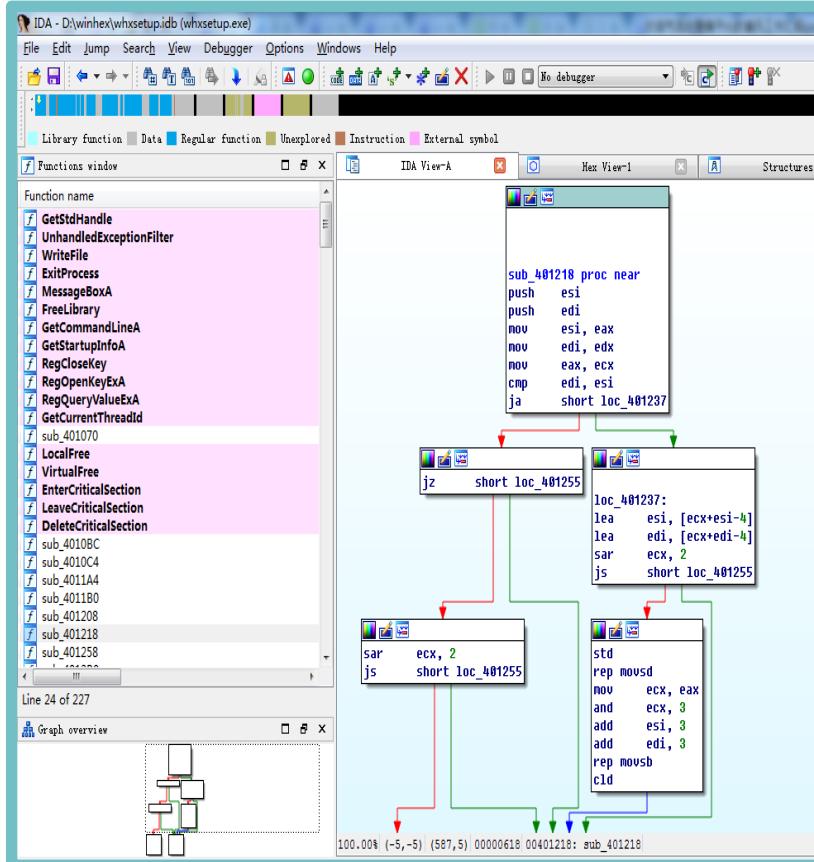
相关工具 - 二进制查壳 - DIE/手动分析

Windows/Linux/Mac
通用

IDA手动分析



静态分析工具 - Windows - IDA - 应用

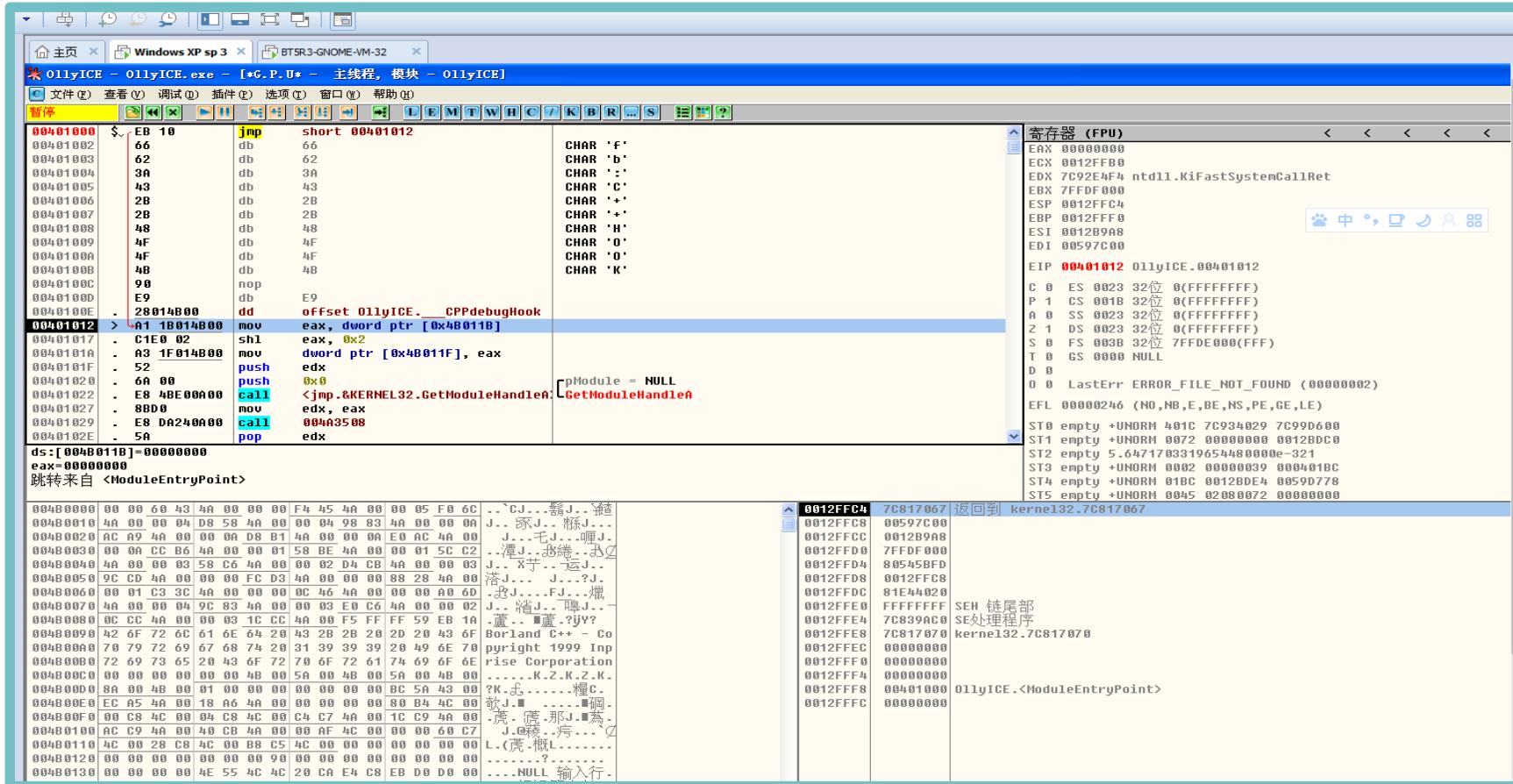


	IDA View-A	Hex View-1
004011D8	00 00 00 EB 0F 80 FB 18 77 0A 33 C0 8A C3 8A 98	...
004011E8	24 70 40 00 33 C0 8A C3 8B D6 E8 AD FF FF 5E	\$P@
004011F8	5B C3 8B C8 83 E0 7F 8B 14 24 E9 A9 FF FF F0 C3	[...]
00401208	53 8B D8 E8 54 11 00 00 89 98 04 00 00 00 58 C3	S...
00401218	56 57 89 C6 89 D7 89 C8 39 F7 77 13 74 2F C1 F9	UW...
00401228	02 78 2A F3 A5 89 C1 83 E1 03 F3 A4 5F 5E C3 8D	.x*
00401238	74 31 FC 8D 7C 39 FC C1 F9 02 78 11 FD F3 A5 89	t1...
00401248	C1 83 E1 03 83 C6 03 83 C7 03 F3 A4 FC 5F 5E C3	...
00401258	53 56 8B D8 33 F6 66 8B 43 04 66 3D B1 D7 72 2F	SU...
00401268	66 3D B3 D7 77 29 66 25 B2 D7 66 3D B2 D7 75 07	F...
00401278	8B C3 FF 53 1C 8B F0 85 F6 75 07 8B C3 FF 53 24	...
00401288	8B F0 85 F6 74 1B 8B C6 E8 73 FF FF EB 12 81	...
00401298	FB 38 80 40 00 74 0A B8 67 00 00 00 E8 5F FF FF	.8#E
004012A8	FF 8B C6 5E 5B C3 8B C0 57 89 C7 88 CD 89 C8 C1	...
004012B8	E0 10 66 89 C8 89 D1 C1 F9 02 78 09 F3 AB 89 D1	..F.
004012C8	83 E1 03 F3 AA 5F C3 90 53 56 8B F2 8B D8 33 C9	...
004012D8	8A 0B 8B D6 8D 43 01 E8 34 FF FF FF 33 C0 8A 03	...
004012E8	C6 04 06 00 5E 5B C3 90 53 31 DB 57 89 D7 89 C1	...
004012F8	83 EC 0A C6 07 30 DC 8B 40 70 40 00 D9 FC DC 93	Hp@
00401308	48 70 40 00 9B DF E0 9E 72 09 DC A3 48 70 40 00	H...
00401318	C6 07 31 DF 34 24 BA 08 00 00 00 47 9B 8A 04 14	..1.
00401328	88 C4 C0 E8 04 80 E4 0F 66 05 30 30 66 AB 4A 79	...
00401338	EB 83 E9 12 7C 06 B0 30 F3 AA EB 14 01 CF 80 3F	...
00401348	35 7C 00 4F FE 07 80 3F 39 7E 05 C6 07 30 EB F3	5 ...
00401358	83 C4 0A 5F 5B C3 8B C0 53 83 EC 0C 31 DB D9 C0	...
00401368	DB 3C 24 66 8B 44 24 08 66 85 C0 74 1B 66 2D FF	<\$1
00401378	3F 66 BA 10 4D 66 F7 EA 0F BF C2 F7 D8 74 16 29	?F...
00401388	C3 E8 D2 01 00 00 EB D6 83 7C 24 04 00 75 DE 83	...
00401398	3C 24 00 75 D8 83 C4 0C 89 D8 5B C3 DB 6C 24 04	<\$.L
004013A8	53 56 57 89 C3 89 D6 51 81 EC 00 01 00 00 81 FB	SUW.
004013B8	FF 00 00 00 7E 05 BB FF 00 00 00 D9 C0 DB 3C 24	...
004013C8	31 C0 66 8B 44 24 08 89 C7 C1 EF 0F 66 25 FF 7F	1.F.
004013D8	66 3D FF 7F 0F 84 3E 01 00 00 D9 E1 85 F6 0F 8D	F...

可与Vmware进行联动，
可与Qemu联动。

调试应用程序；
调试固件

动态分析工具 - windows - OllyDbg - 应用



Windows平台下最常用的动态分析工具

Reverse环境搭建 - windows/Linux工具集(1)

静态分析

- Winhex/UltraEdit/..
- PEID/LoadPE/DIE/..
- IDA and plugins

动态分析

- R3 debugging: Ollydbg/gdb (应用级调试)
- R0 debugging: windbg/kgdb, softice.. (内核级调试)
- VM images: vmware (常用于调试的动态模拟环境) /virtualBox/Qemu

OllyDbg工具的使用 - 基本用法

F2: 设置断点，在光标位置按F2键即可，再按则会删除断点

F3: 打开文件

F4: 运行到选定位置。作用就是直接运行到光标所在位置处暂停

F7: 单步步入。功能同(F8)，遇到 CALL 等子程序时会进入其中

F8: 单步步过。每按一次执行一条指令，遇到 CALL 等子程序不进入

F9: 运行。被调试的程序将直接开始运行

ALT+F9: 执行到用户代码。可用于从系统领空快速返回到我们调试的程序领空

ALT+M: 查看当前程序的加载模块，可用于分析目标程序主文件结构

CTRL+E: 编辑数据

CTRL+F2: 重新开始调试

CTRL+F9: 执行到返回。此命令在执行到一个 ret (返回指令) 指令时暂停，常用于从系统领空返回到我们调试的程序领空

SHIFT+F7: 忽略异常后单步执行

SHIFT+F8: 忽略异常后单步步过

OllyDbg工具的使用 - 基本汇编指令

MOV 传送字或字节 如MOV A B，就是将B中的字传给A

PUSH 把字压入堆栈

CALL 子程序调用指令

XOR 异或运算 所谓异或，就是两值不同，则为真，反之，为假

RET 子程序返回指令

CMP 比较.(两操作数作减法,仅修改标志位,不回送结果)

JNZ/jNE OPR --结果不为零转移,测试条件ZF=0

DEC 减 1 **INC** 加 1

JZ(或jE) OPR--结果为零转移, 测试条件ZF=1

SUB 减法

LEA 装入有效地址 例: LEA DX,string ;把偏移地址存到DX.

MOVSX 先符号扩展,再传送

REP 当CX/ECX0时重复

AND 与运算

TEST 测试.(两操作数作与运算,仅修改标志位,不回送结果)

OillyDbg工具的使用 - 关键位置

004011AE下一个断点,有调用到GetDlgItemTextA这个函数

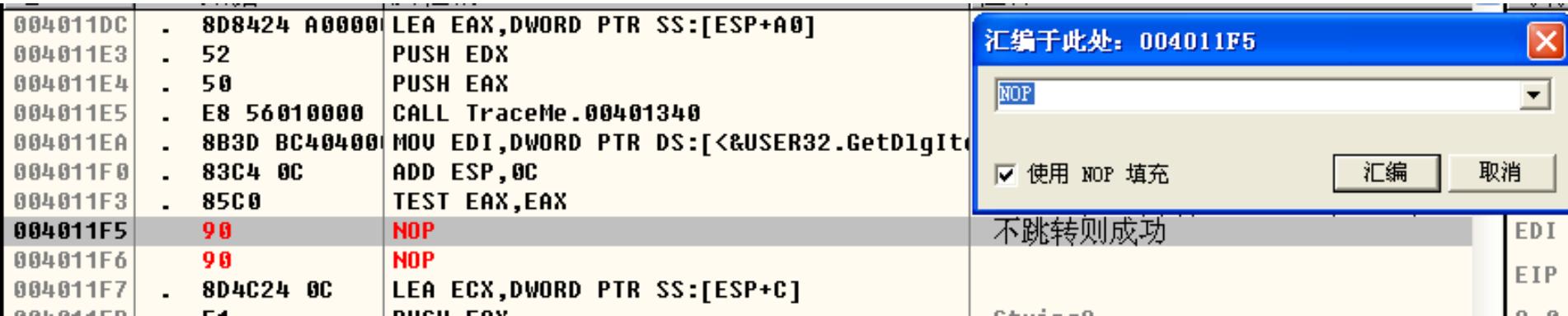
004011AA	. 8D4424 4C	LEA EAX,DWORD PTR SS:[ESP+4C]	
004011AE	. 6A 51	PUSH 51	Count = 51 (81.)
004011B0	. 50	PUSH EAX	Buffer
004011B1	. 6A 6E	PUSH 6E	ControlID = 6E (1)
004011B3	. 56	PUSH ESI	hWnd
004011B4	. FFD7	CALL EDI	GetDlgItemTextA
004011B6	. 8D8C24 9C0000	LEA ECX,DWORD PTR SS:[ESP+9C]	
004011BD	. 6A 65	PUSH 65	Count = 65 (101.)
004011BF	. 51	PUSH ECX	Buffer

开始分析汇编代码的意义，所以，我们在使用这个软件的时候，一定要明白这些代码的含义，此处只是爆破，接下来就是修改代码：004011F5

004011E4	. 50	PUSH EAX	
004011E5	. E8 56010000	CALL TraceMe.00401340	序列号计算的CALL
004011EA	. 8B3D BC404000	MOV EDI,DWORD PTR DS:[<&USER32.GetDlgItemTextA>]	USER32.GetDlgItem
004011F0	. 83C4 0C	ADD ESP,0C	
004011F3	. 85C0	TEST EAX,EAX	EXA=0,注册失败； EXA=1, 注册成功
004011F5	. 74 37	JE SHORT TraceMe.0040122E	不跳转则成功

OllDbg工具的使用 - 爆破

修改反汇编代码段，双击反汇编列后按空格键，键入NOP，汇编



最后 F9运行，你会看到：



面向汇编的逐句解析 - 关键点

KernelMode - Crackme3.exe - [*G.P.U* - main thread, module Crackme3]

C File View Debug Plugins Options Window Help Tools BreakPoint-> BP P VB Notepad Calc Folder

Paused

004013CB 90 nop
004013CC 90 nop
004013CD 90 nop
004013CE 90 nop
004013CF 90 nop
004013D0 . 83EC 10 sub esp,0x10
004013D3 . 8D4424 00 lea eax,dword ptr ss:[esp]
004013D7 . 56 push esi
004013D8 . 6A 0F push 0xF
004013DA . 50 push eax
004013DB . 8BF1 mov esi,ecx
004013DD . 68 E8030000 push 0x3E8
004013E2 . E8 75020000 call <jmp.&MFC42.#?GetDlgItemTextA@CWnd@@QBEHHPADH@Z_3098>
004013E7 . 83F8 08 cmp eax,0x8
004013EA . 75 5D jnz short Crackme3.00401449

面向汇编的逐句解析 - 长度判断

KernelMode - Crackme3.exe - [*G.P.U* - main thread, module Crackme3]

C File View Debug Plugins Options Window Help Tools BreakPoint-> BP P VB Notepad Calc Folder CMD

Paused

004013CB	90	nop
004013CC	90	nop
004013CD	90	nop
004013CE	90	nop
004013CF	90	nop
004013D0	. 83EC 10	sub esp,0x10
004013D3	. 8D4424 00	lea eax,dword ptr ss:[esp]
004013D7	. 56	push esi
004013D8	. 6A 0F	push 0xF
004013DA	. 50	push eax
004013DB	. 8BF1	mov esi,ecx
004013DD	. 68 E8030000	push 0x3E8
004013E2	. E8 75020000	call <jmp.&MFC42.#!GetDlgItemTextA@0
004013E7	. 83F8 08	cmp eax,0x8
004013EA	.~ 75 5D	jnz short Crackme3.00401449
004013EC	. 807C24 09 2D	cmp byte ptr ss:[esp+0x9],0x2D
004013F1	.~ 75 56	jnz short Crackme3.00401449
004013F3	. 0FBE4C24 04	movsx ecx,byte ptr ss:[esp+0x4]
004013F8	. D1E1	shl ecx,1
004013FA	. 83F9 64	cmp ecx,0x64
004013FD	.~ 75 4A	jnz short Crackme3.00401449
004013FF	. 8A4424 0B	mov al,byte ptr ss:[esp+0x8]
00401403	. 84C0	test al,al

eax=00000006

注册码长度等于8，不相等就跳出去，注册失败

Project PolyPhemous-NTS-Crackme3 X

Serial

Serial
123123

Check Close

Cyclops / REAL

面向汇编的逐句解析 - 字符判断

KernelMode - Crackme3.exe - [*G.P.U* - main thread, module Crackme3]

C File View Debug Plugins Options Window Help Tools BreakPoint-> BP P VB Notepad Calc Folder CMD

Paused

004013DB . 8BF1 mov esi,ecx
004013DD . 68 E8030000 push 0x3E8
004013E2 . E8 75020000 call <jmp.&MFC42.#!GetDlgItemTextA
004013E7 . 83F8 08 cmp eax,0x8
004013EA . 74 5D je short Crackme3.00401449
004013EC . 807C24 09 2D cmp byte ptr ss:[esp+0x9],0x2D
004013F1 . 74 56 je short Crackme3.00401449
004013F3 . 0FBEC24 04 movsx ecx,byte ptr ss:[esp+0x4]
004013F8 . D1E1 shl ecx,1
004013FA . 83F9 64 cmp ecx,0x64
004013FD . 74 4A je short Crackme3.00401449
004013FF . 8A4424 0B mov al,byte ptr ss:[esp+0xB]
00401403 . 84C0 test al,al
00401405 . 75 42 jnz short Crackme3.00401449
00401407 . 807C24 08 2B cmp byte ptr ss:[esp+0x8],0x2B
0040140C . 74 3B je short Crackme3.00401449
0040140E . 0FBEB424 05 movsx edx,byte ptr ss:[esp+0x5]
00401413 . 83C2 0A add edx,0xA
00401416 . 83FA 44 cmp edx,0x44
00401419 . 75 2E jnz short Crackme3.00401449
0040141B . 0FBEB424 07 movsx eax,byte ptr ss:[esp+0x7]
00401420 . 83E8 2E sub eax,0x2E
00401423 . 75 24 jnz short Crackme3.00401449

edx=00000032

注册码长度等于8，不相等就跳出去，注册失败
字符“-”所以注册码第6位为-
注册码第1位扩展到ECX
左移一位
字符d,把d右移一位则为32H,说明注册码第1位是数字2
注册码第8位放入AL
第8位是否为0
第8位只要不为0即可
第5位为2B即十进制43,就是+号
注册码第2位零扩展到EDX
与000AH相加
加完之后等于44H,即第2位为44-A=3A,也就是:号

SEH结构体异常处理

- 异常处理结构体（Structure Exception Handler, SEH）是Windows异常处理机制所采用的重要数据结构
- 每个SEH结构体包含两个DWORD指针：SEH链表指针和异常处理函数句柄
- 当GUI应用程序触发一个消息时，系统将把该消息放入消息队列，然后去查找并调用窗体的回调函数，即消息处理函数
- 与之类似，异常也可视为一种消息，应用程序发生异常时就触发了该消息，系统会将异常放入SEH结构体中，调用它的回调函数，即异常处理函数

花指令

- 花指令是程序中的无用指令或者垃圾指令，故意干扰各种反汇编静态分析工具，但是程序不受任何影响，缺少了也能正常运行
- 加花指令后，IDA等分析工具对程序静态反汇编时，往往会出现错误或者遭到破坏，加大逆向静态分析的难度，从而隐藏自身的程序结构和算法，从而较好的保护自己
- 花指令有可能利用各种指令：jmp, call, ret的一些堆栈技巧，位置运算等

寻找程序入口点 (Original Entry Point, OEP)

- 软件加壳就是隐藏了OEP（或者用了假的OEP/花指令等，例如直接转到ExitProcess等处），只要找到程序真正的OEP，可以实现脱壳。一般的查壳工具无法直接识别出OEP
- ESP定律：即堆栈平衡定律，是应用频率最高的脱壳方法之一，可以应对简单的**压缩壳**（壳的种类可以由PEID等工具进行分析）。最后一次异常等方法也常用于识别OEP
- 壳实质上是一个子程序，它在程序运行时首先取得控制权并对程序进行压缩，同时隐藏程序真正的OEP

寻找程序入口点 (OEP)

- 在程序自解压过程中，多数壳会先将当前寄存器状态压栈，如使用PUSHAD，而在解压结束后，会将之前的寄存器值出栈，如使用POPAD。
- 基于PUSHAD和POPAD的对称性，可以利用硬件断点定位真正的OEP：当壳把代码解压前和解压后，必须要平衡堆栈，让执行到OEP的时候，使ESP=0012FFC4。这就是ESP定律
- 例如，PUSHAD的时候将寄存器值压入了0012FFC0到0012FFA4的堆栈中。通过在0012FFA4下硬件断点，等POPAD恢复堆栈，即可停在OEP处

手动脱壳

- 定位到OEP之后，使用LordPE等工具把程序的镜像dump出来
- 但dump得到的镜像无法运行，因为无法自动获取导入函数的地址
- 为此，需要修复导入函数地址表（Import Address Table, IAT）。通常使用importrec工具进行修复，从原文件（加壳的文件）提取信息后，对脱壳文件进行修复

Stolen Code

- 某些壳在处理OEP代码的时候，把OEP处固定的代码NOP掉，然后把这些代码（即stolen code）放到壳代码的空间中去，而且常伴随着花指令，使原程序的起始代码从壳空间开始执行，然后再JMP回原程序空间
- 如果脱掉壳，这一部分代码就会遗失，也就达到了反脱壳的目的。这就是stolen OEP code技术
- 如果dump以后修复IAT，这里OEP依旧是错误的，程序无法运行
- 原因：前面几行代码被放在壳空间中，所以不会被转储，因此也得不到执行
- 解决方法：寻找真正的OEP，找到缺失的代码

反调试技术

反调试分类：

1. 调试器检测：各种方法查看调试器是否存在
2. 识别调试器：识别是否在调试中
3. 干扰调试器：令调试失败

反调试技术

1. 调试器检测: Windows API, 手动检测数据结构, 系统痕迹检测
2. 识别调试器: 检测软件/硬件断点, 时钟检测, 父进程判断
3. 干扰调试器: TLS回调, 利用中断, 陷阱标志位

软件漏洞概述

- 内存安全违规类
 - 内存访问时引入的安全缺陷，如：缓冲区溢出、Use-after-Free、Double free等不安全指针问题。
- 输入验证类
 - 程序在对用户输入进行数据验证存在错误，没有保证输入数据 的正确性、合法性和安全性，导致可能被恶意攻击与利用；
 - 例如：格式化字符串、SQL注入、代码注入、远程文件包含、目录遍历、XSS、HTTP Header注入、HTTP响应分割错误等。
- 竞争条件类
 - 涉及多进程或多线程处理的程序中，输出或结果无法预测，依赖于其他进程事件发生的次序或时间TOCTTOU, 检查时刻与使用时刻条件状态不一致，导致攻击者可利用的漏洞
- 权限混淆与提升类
 - 程序由于自身编程疏忽或被第三方欺骗，滥用其特权，或赋予第三方不该给予的权限
 - 权限提升漏洞通常发生在一些拥有特权的应用程序 中，由于安全缺陷，使得获取特权的安全检查被绕过， 或被攻破

缓冲区溢出概述

- 缓冲区溢出是最早被发现，也是最基础的安全软件漏洞类型（特别是栈溢出）
- 缓冲区溢出基本概念
 - 本质：内存安全违规类漏洞，计算机程序向特定缓冲区内填充数据时，超出了缓冲区本身的容量，导致外溢数据覆盖了相邻内存空间的合法数据，从而改变程序执行流程破坏系统运行完整性
 - 原因：程序本身没有检查数据长度与所分配的存储空间是否匹配
 - 多见于C/C++中的`memcpy()`、`strcpy()`等内存与字符串复制函数的引用位置，这些函数不检查内存越界问题
 - 根本原因：现代计算机基础构架——冯诺依曼体系的安全缺陷，即程序的数据和指令都在同一内存中进行存储

缓冲区溢出概述

- 堆栈的基本定义
 - 堆栈是操作系统使用内存的方式，区别在？
 - 栈的操作方式：push/pop
 - 栈的生长方向：高地址 -> 低地址
- 栈溢出攻击
 - 针对函数调用过程中返回地址
 - 对栈中的存储返回地址的位置进行缓冲区溢出
 - 改写返回地址
 - 使程序跳转到攻击者指定的位置执行恶意代码

安全保护机制概述

- 微软安全保护机制：

- ✓ GS编译技术（编译层+OS层）
- ✓ SEH的安全校验机制（编译层+OS层）
- ✓ Heap Cookie, Safe Unlinking等堆安全机制（编译层）
- ✓ DEP数据执行保护（处理器层）
- ✓ ASLR加载地址随机化（OS层）
- ✓ SEHOP安全机制（编译层+OS层）

祝大家考试顺利！



南京邮電大學

Nanjing University of Posts and Telecommunications