Introduction to Reverse Engineering

Assembly Language

- Different processors will have different instruction sets
  - X86, ARM, etc…
  - For the purposes of this workshop we will be looking at general concepts

Assembly vs “regular” programming languages

Regular

- Data is stored in variables
- Computer handles return addresses and function calls
- Data and program are seen as separate

Assembly

- Data can be stored either in Registers or memory
- Typically we have to determine how the return address is stored
- Data and program exist together – we must make the distinction of which is which

Registers

- Designed to hold a set amount of data typically anywhere from 4-8 bytes
- Used as temporary storage
- Limited in number

Special registers

- Program Counter – holds the location of the next instruction
- Status Register – holds flags that are set by comparison operations
- Stack Pointer – points to the top of the stack (more on that)

The stack

- Used to store data between subroutines
- Saves any registers that needs to be changed and restores them
- Holds the return address (IMPORTANT)
- This is what we call the calling convention

Typical calling routine

- Main program calls the function
- Return address gets stored onto the stack
- Function must store any registers it changes onto the stack
- Once function is done it will then restore the registers
- Return address is used to reset the program counter

General goals of reverse engineering

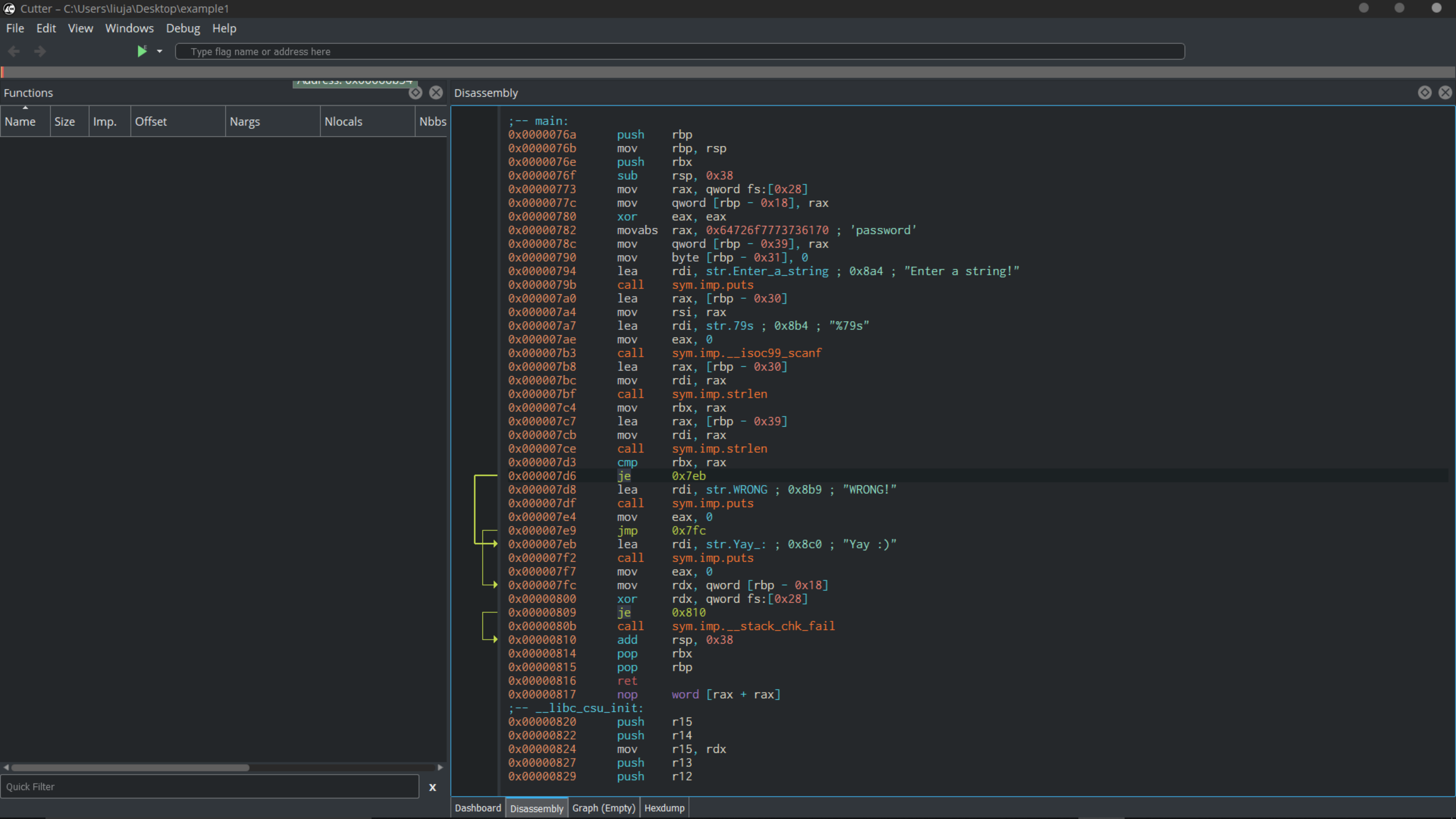
- If I have an unknown program what input does it require?
  - Used in malware reverse engineering
    - Stuxnet – targeted only a specific microprocessor – which processor? How did it tell?

Microcorruption Demo

- Create an account at https://microcorruption.com

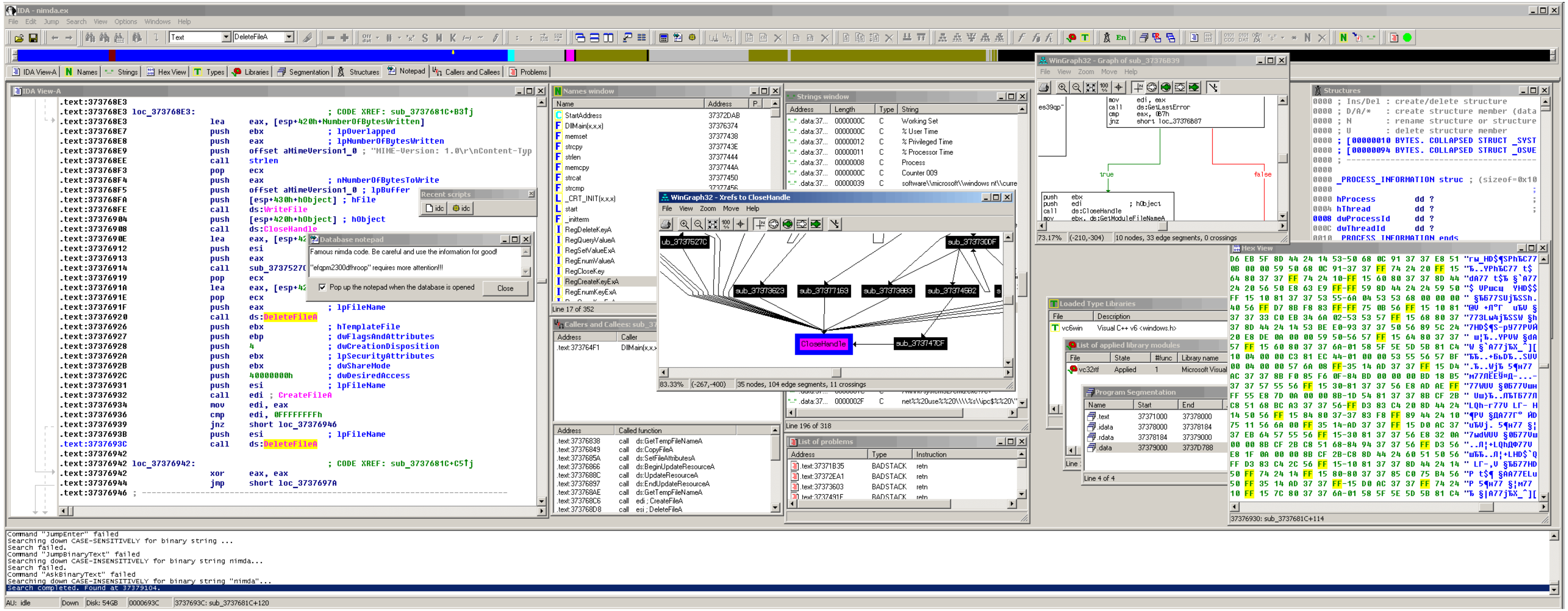
Introduction to Cutter

- Cutter is a debugger and disassembler based off of Radare2
- Install from https://github.com/radareorg/cutter/releases



```
;-- main:
0x0000076a      push    rbp
0x0000076b      mov     rbp, rsp
0x0000076e      push    rbx
0x0000076f      sub     rsp, 0x38
0x00000773      mov     rax, qword fs:[0x28]
0x0000077c      mov     qword [rbp - 0x18], rax
0x00000780      xor     eax, eax
0x00000782      movabs  rax, 0x64726f7773736170 ; 'password'
0x0000078c      mov     qword [rbp - 0x39], rax
0x00000790      mov     byte [rbp - 0x31], 0
0x00000794      lea     rdi, str.Enter_a_string ; 0x8a4 ; "Enter a string!"
0x0000079b      call    sym.imp.puts
0x000007a0      lea     rax, [rbp - 0x30]
0x000007a4      mov     rsi, rax
0x000007a7      lea     rdi, str.79s ; 0x8b4 ; "%79s"
0x000007ae      mov     eax, 0
0x000007b3      call    sym.imp.__isoc99_scanf
0x000007b8      lea     rax, [rbp - 0x30]
0x000007bc      mov     rdi, rax
0x000007bf      call    sym.imp.strlen
0x000007c4      mov     rbx, rax
0x000007c7      lea     rax, [rbp - 0x39]
0x000007cb      mov     rdi, rax
0x000007ce      call    sym.imp.strlen
0x000007d3      cmp     rbx, rax
0x000007d6      je      0x7eb
0x000007d8      lea     rdi, str.WRONG ; 0x8b9 ; "WRONG!"
0x000007df      call    sym.imp.puts
0x000007e4      mov     eax, 0
0x000007e9      jmp     0x7fc
0x000007eb      lea     rdi, str.Yay_: ; 0x8c0 ; "Yay :)"
0x000007f2      call    sym.imp.puts
0x000007f7      mov     eax, 0
0x000007fc      mov     rdx, qword [rbp - 0x18]
0x00000800      xor     rdx, qword fs:[0x28]
0x00000809      je      0x810
0x0000080b      call    sym.imp.__stack_chk_fail
0x00000810      add     rsp, 0x38
0x00000814      pop     rbx
0x00000815      pop     rbp
0x00000816      ret
0x00000817      nop     word [rax + rax]
;-- __libc_csu_init:
0x00000820      push    r15
0x00000822      push    r14
0x00000824      mov     r15, rdx
0x00000827      push    r13
0x00000829      push    r12
```

A proper debugger



IDA - nimda.ex
File Edit Jump Search View Options Windows Help
IDA View-A | Names | Strings | Hex View | Types | Libraries | Segmentation | Structures | Notepad | Callers and Callees | Problems
WinGraph32 - Xrefs to CloseHandle
WinGraph32 - Graph of sub_37376B39
Recent scripts
idc
idc

IDA View-A

```
.text:373768E3
.text:373768E3 loc_373768E3:                           ; CODE XREF: sub_3737681C+B3↑j
.text:373768E3                 lea     eax, [esp+420h+NumberOfBytesWritten]
.text:373768E7                 push    ebx             ; lpOverlapped
.text:373768E8                 push    eax             ; lpNumberOfBytesWritten
.text:373768E9                 push    offset aMimeVersion1_0 ; "MIME-Version: 1.0\r\nContent-Typ
.text:373768EE                 call    strlen
.text:373768F3                 pop     ecx
.text:373768F4                 push    eax             ; nNumberOfBytesToWrite
.text:373768F5                 push    offset aMimeVersion1_0 ; lpBuffer
.text:373768FA                 push    [esp+430h+hObject] ; hFile
.text:373768FE                 call    ds:WriteFile
.text:37376904                 push    [esp+420h+hObject] ; hObject
.text:37376908                 call    ds:CloseHandle
.text:3737690E                 lea     eax, [esp+42
.text:37376912                 push    esi
.text:37376913                 push    eax
.text:37376914                 call    sub_3737527C
.text:37376919                 pop     ecx
.text:3737691A                 lea     eax, [esp+42
.text:3737691E                 pop     ecx
.text:3737691F                 push    eax             ; lpFileName
.text:37376920                 call    ds:DeleteFileA
.text:37376926                 push    ebx             ; hTemplateFile
.text:37376927                 push    ebp             ; dwFlagsAndAttributes
.text:37376928                 push    4               ; dwCreationDisposition
.text:3737692A                 push    ebx             ; lpSecurityAttributes
.text:3737692B                 push    ebx             ; dwShareMode
.text:3737692C                 push    40000000h       ; dwDesiredAccess
.text:37376931                 push    esi             ; lpFileName
.text:37376932                 call    edi ; CreateFileA
.text:37376934                 mov     edi, eax
.text:37376936                 cmp     edi, 0FFFFFFFFh
.text:37376939                 jnz     short loc_37376946
.text:3737693B                 push    esi             ; lpFileName
.text:3737693C                 call    ds:DeleteFileA
.text:37376942
.text:37376942 loc_37376942:                           ; CODE XREF: sub_3737681C+C5↑j
.text:37376942                 xor     eax, eax
.text:37376944                 jmp     short loc_3737697A
.text:37376946 ; ---------------------------------------------------------------
```

Database notepad

Famous nimda code. Be careful and use the information for good!

"efqpm2300dfhroop" requires more attention!!!

☑ Pop up the notepad when the database is opened

Names window

Name	Address	P..
C StartAddress	37372DAB	
F DllMain(x,x,x)	37376374	
F memset	37377438	
F strcpy	3737743E	
F strlen	37377444	
F memcpy	3737744A	
F strcat	37377450	
F strcmp	37377456	
L _CRT_INIT(x,x,x)		
L start		
F _initterm		
I RegDeleteKeyA		
I RegQueryValueA		
I RegSetValueExA		
I RegEnumValueA		
I RegCloseKey		
I RegCreateKeyExA		
I RegEnumKeyExA		

Line 17 of 352

Callers and Callees: sub_37

Address	Caller
.text:373764F1	DllMain(x,x,x

Address	Called function
.text:37376838	call ds:GetTempFileNameA
.text:37376849	call ds:CopyFileA
.text:3737685A	call ds:SetFileAttributesA
.text:37376866	call ds:BeginUpdateResourceA
.text:3737688C	call ds:UpdateResourceA
.text:37376897	call ds:EndUpdateResourceA
.text:373768AE	call ds:GetTempFileNameA
.text:373768C6	call edi ; CreateFileA
.text:373768D8	call esi ; DeleteFileA

Strings window

Address	Length	Type	String
.data:37...	0000000C	C	Working Set
.data:37...	0000000C	C	% User Time
.data:37...	00000012	C	% Privileged Time
.data:37...	00000011	C	% Processor Time
.data:37...	00000008	C	Process
.data:37...	0000000C	C	Counter 009
.data:37...	00000039	C	software\\microsoft\\windows nt\\curre
.data:37...	0000002F	C	net%%20use%%20\\\\%s\\ipc$%%20\"

Line 196 of 318

83.33% | (-267,-400) | 35 nodes, 104 edge segments, 11 crossings

List of problems

Address	Type	Instruction
.text:37371B35	BADSTACK	retn
.text:37372EA1	BADSTACK	retn
.text:37373603	BADSTACK	retn
.text:3737491F	BADSTACK	retn

WinGraph32 - Graph of sub_37376B39

```
mov     edi, eax
call    ds:GetLastError
cmp     eax, 0B7h
jnz     short loc_37376B87
```

```
push    ebx
push    edi             ; hObject
call    ds:CloseHandle
mov     ebx, ds:GetModuleFileNameA
```

73.17% | (-210,-304) | 10 nodes, 33 edge segments, 0 crossings

Structures

```
0000 ; Ins/Del : create/delete structure
0000 ; D/A/*   : create structure member (data
0000 ; N       : rename structure or structure
0000 ; U       : delete structure member
0000 ; [00000010 BYTES. COLLAPSED STRUCT _SYST
0000 ; [00000094 BYTES. COLLAPSED STRUCT _OSVE
0000 ; ---------------------------------------
0000
0000 _PROCESS_INFORMATION struc ; (sizeof=0x10
0000                                          ;
0000 hProcess        dd ?                     ;
0004 hThread         dd ?                     ;
0008 dwProcessId     dd ?
000C dwThreadId      dd ?
0010 _PROCESS_INFORMATION ends
```

Loaded Type Libraries

File	Description
vc6win	Visual C++ v6 <windows.h>

List of applied library modules

File	State	#func	Library name
vc32rtf	Applied	1	Microsoft Visual

Program Segmentation

Name	Start	End
.text	37371000	37378000
.idata	37378000	37378184
.rdata	37378184	37379000
.data	37379000	3737D788

Line 4 of 4

Hex View

37376930: sub_3737681C+114

```
Command "JumpEnter" failed
Searching down CASE-SENSITIVELY for binary string ...
Search failed.
Command "JumpBinaryText" failed
Searching down CASE-INSENSITIVELY for binary string nimda...
Search failed.
Command "AskBinaryText" failed
Searching down CASE-INSENSITIVELY for binary string "nimda"...
Search completed. Found at 37379104.
```

AU: idle | Down | Disk: 54GB | 0000693C | 3737693C: sub_3737681C+120