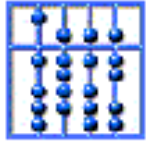




TU Muenchen - Fakultät fuer Informatik
Lehrstuhl IV: Software and Systems Engineering
Seminar: Management of software systems



REVERSE ENGINEERING

Catinca Golesteanu

Coordinator: Dr. Jan Juerjens

13th December 2005

Why reverse engineering?





About reverse engineering ...

- Definitions
- Types
- Examples of reverse engineered software
- Reverse engineering source code
 - CodeSurfer
- Reverse engineering binary software
 - Lutz Roeder's .NET Reflector
 - IDA Pro
 - Netstat command, Sysinternals PE and Filemon
- Conclusions

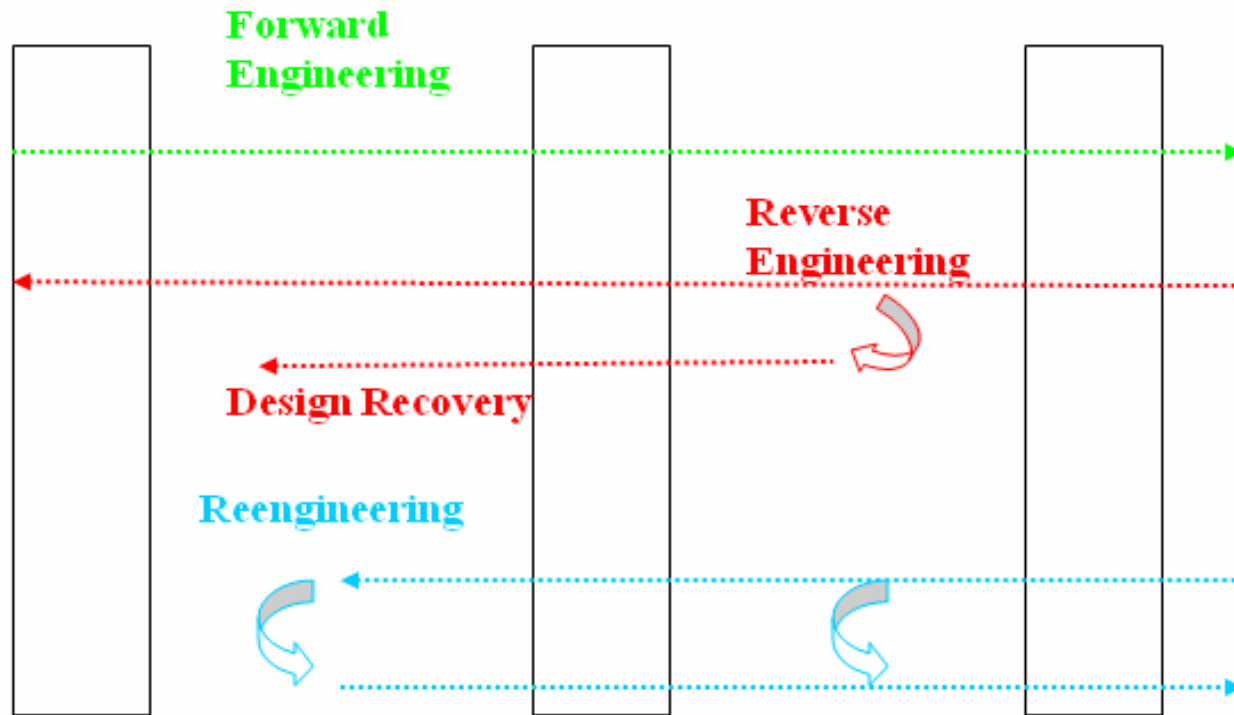


Definitions of Reverse Engineering

- Assumption: hackers and crackers ?
- 90's: “a process of analyzing a subject system to identify the system's components and their interrelationships and to create representations of the system at a higher level of abstraction”
- Wikipedia: “the process of taking something apart and analyzing its workings in detail, usually with the intention to construct a new device or program that does the same thing without actually copying anything from the original.”

Software maintenance and life cycles

Requirements Design Implementation



Restructuring



Restructuring



Restructuring
Redocumentation

[Chi]



Reverse engineering as analyzing process of ...

- **source code**

- Discovery of higher level aspects of the program

- **binary software**

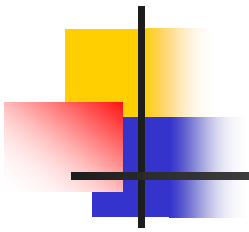
- no source code is available for the software
- Discovery of one possible source code for the software

! Reverse engineering *remains* a process of examination, and does *not* become a process of piracy and breaking software copy protection.



Examples

- the first non-IBM implementation of BIOS
 - Reverse engineering of BIOS binary image to find out BIOS functionality
- Samba software:
 - Reverse engineering of unpublished information about Windows file sharing -> for emulation on non-Windows computers
- In other domains:
 - Jerry cans
 - Tupolev Tu-4



Reverse engineering source code

- **CodeSurfer** – source code analyzer from GrammaTech
<http://www.grammatech.com>
- **Understanding for C++** - source code analyzer from Scientific Toolworks
<http://www.scitools.com>



CodeSurfer (I)

- Understanding source code
- Maintenance
- Impact analysis
- Debugging
- **Reverse engineering**
- Reuse
- deep-structure representation:
 - Abstract Syntax Trees (ASTs)
 - Normalized Program
 - Program Points
 - Control Flow Graphs (CFGs)
 - System Dependence Graph (SDG)
 - Non-local Variable Usage Information
 - Pointer Analysis Database

CodeSurfer(II) – Project View

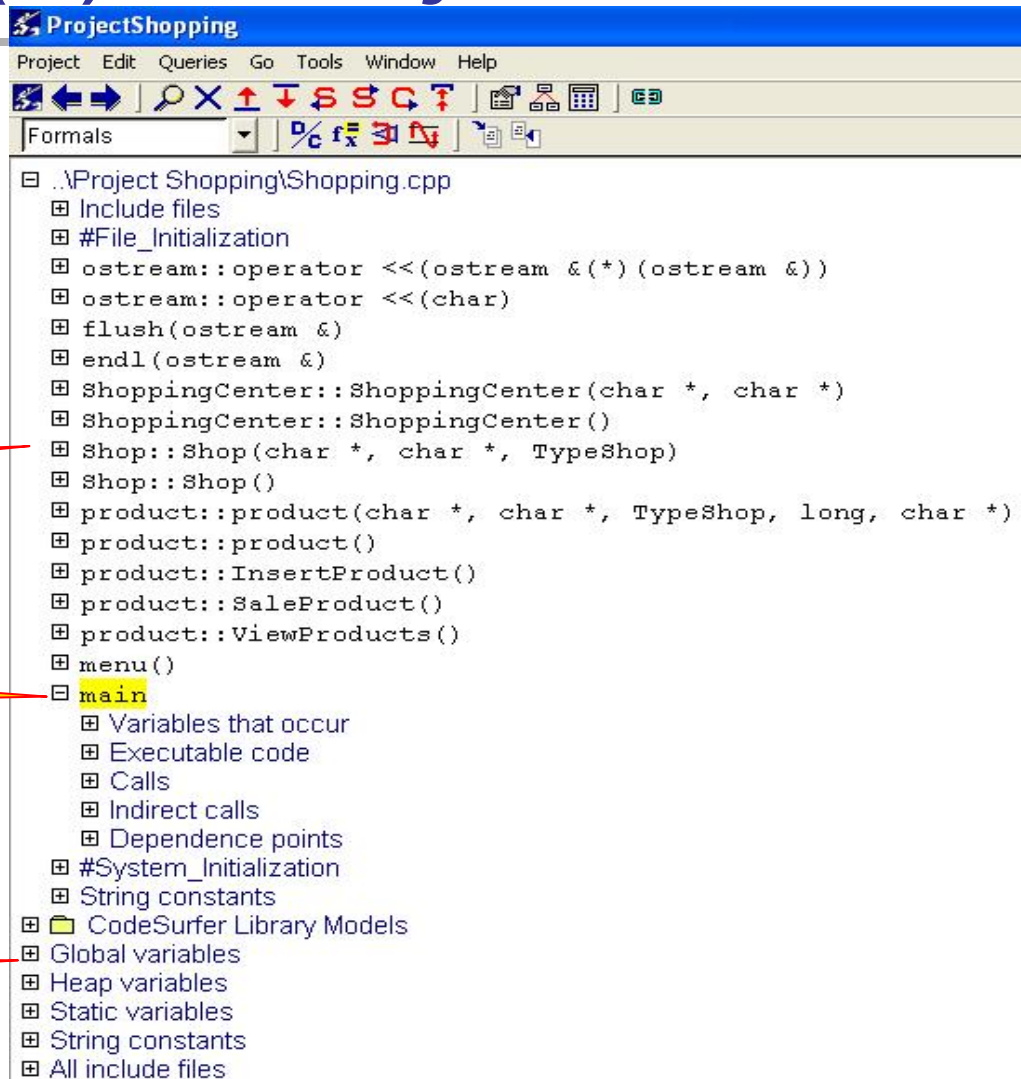
■ ProjectShopping

- Visual C++ from Visual Studio 6.0

All functions

Main function

All variables

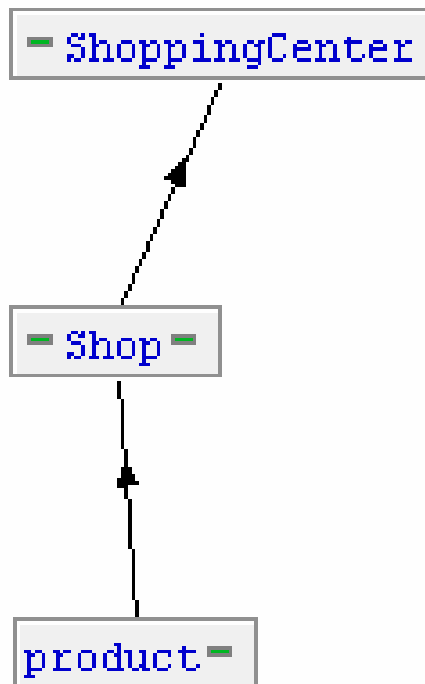


The screenshot shows the CodeSurfer interface for a project named 'ProjectShopping'. The main window displays a tree view of the file 'Shopping.cpp'. The tree is expanded to show the 'main' function, which is highlighted in yellow. The tree structure is as follows:

- ..Project Shopping\Shopping.cpp
 - Include files
 - #File_Initialization
 - ostream::operator <<(ostream &(*) (ostream &))
 - ostream::operator <<(char)
 - flush(ostream &)
 - endl(ostream &)
 - ShoppingCenter::ShoppingCenter(char *, char *)
 - ShoppingCenter::ShoppingCenter()
 - Shop::Shop(char *, char *, TypeShop)
 - Shop::Shop()
 - product::product(char *, char *, TypeShop, long, char *)
 - product::product()
 - product::InsertProduct()
 - product::SaleProduct()
 - product::ViewProducts()
 - menu()
 - main**
 - Variables that occur
 - Executable code
 - Calls
 - Indirect calls
 - Dependence points
 - #System_Initialization
 - String constants
- CodeSurfer Library Models
 - Global variables
 - Heap variables
 - Static variables
 - String constants
 - All include files

CodeSurfer(III)

➤ Class View and Class Hierarchy Graph



The screenshot shows the CodeSurfer ProjectShopping window. The menu bar includes Project, Edit, Queries, Go, Tools, Window, and Help. The toolbar contains various navigation and editing icons. The main area displays the source code for three classes: product, Shop, and ShoppingCenter.

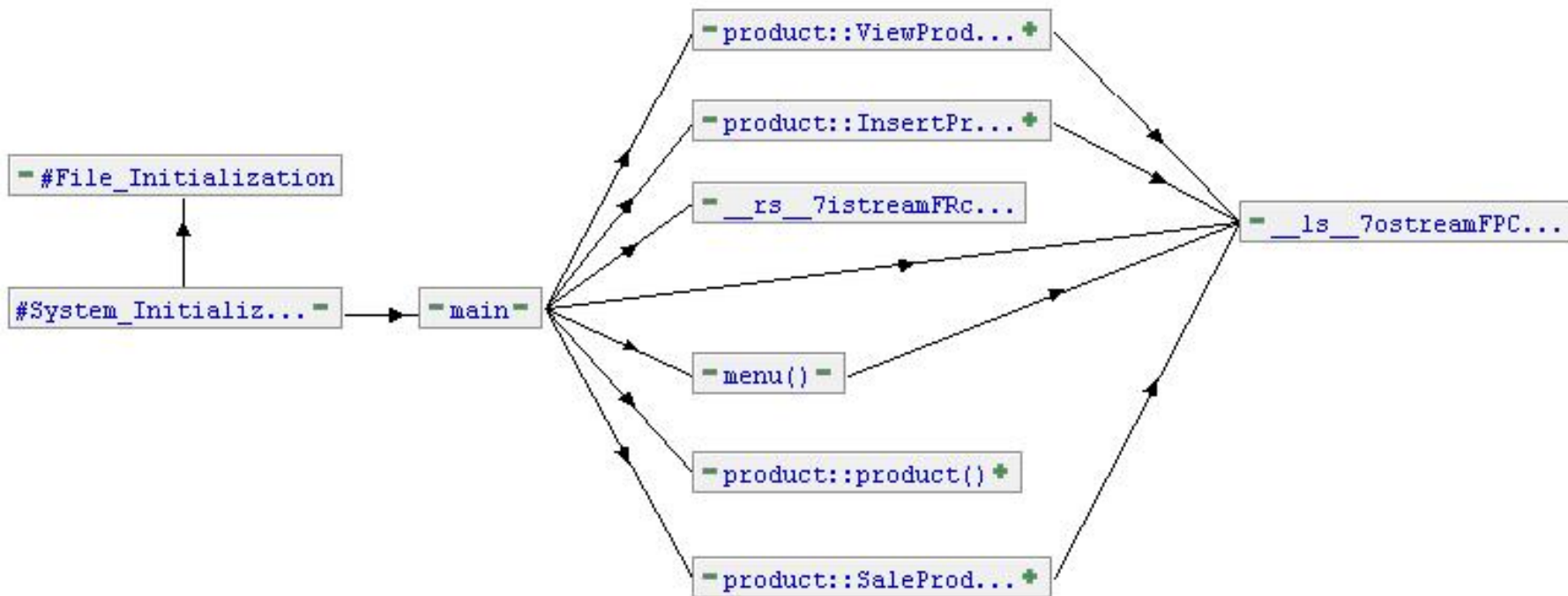
```
product
  price
  name_p
  next
  head
  product(char *, char *, TypeShop, long, ...)
  product()
  InsertProduct()
  SaleProduct()
  ViewProducts()
  product(const product &)
  operator=(const product &)

Shop
  name_s
  Shop(char *, char *, TypeShop)
  Shop()
  Shop(const Shop &)
  operator=(const Shop &)

ShoppingCenter
  name
  adress
  ShoppingCenter(char *, char *)
  ShoppingCenter()
  InsertProduct()
  SaleProduct()
  ViewProducts()
  ShoppingCenter(const ShoppingCenter &)
  operator=(const ShoppingCenter &)
```

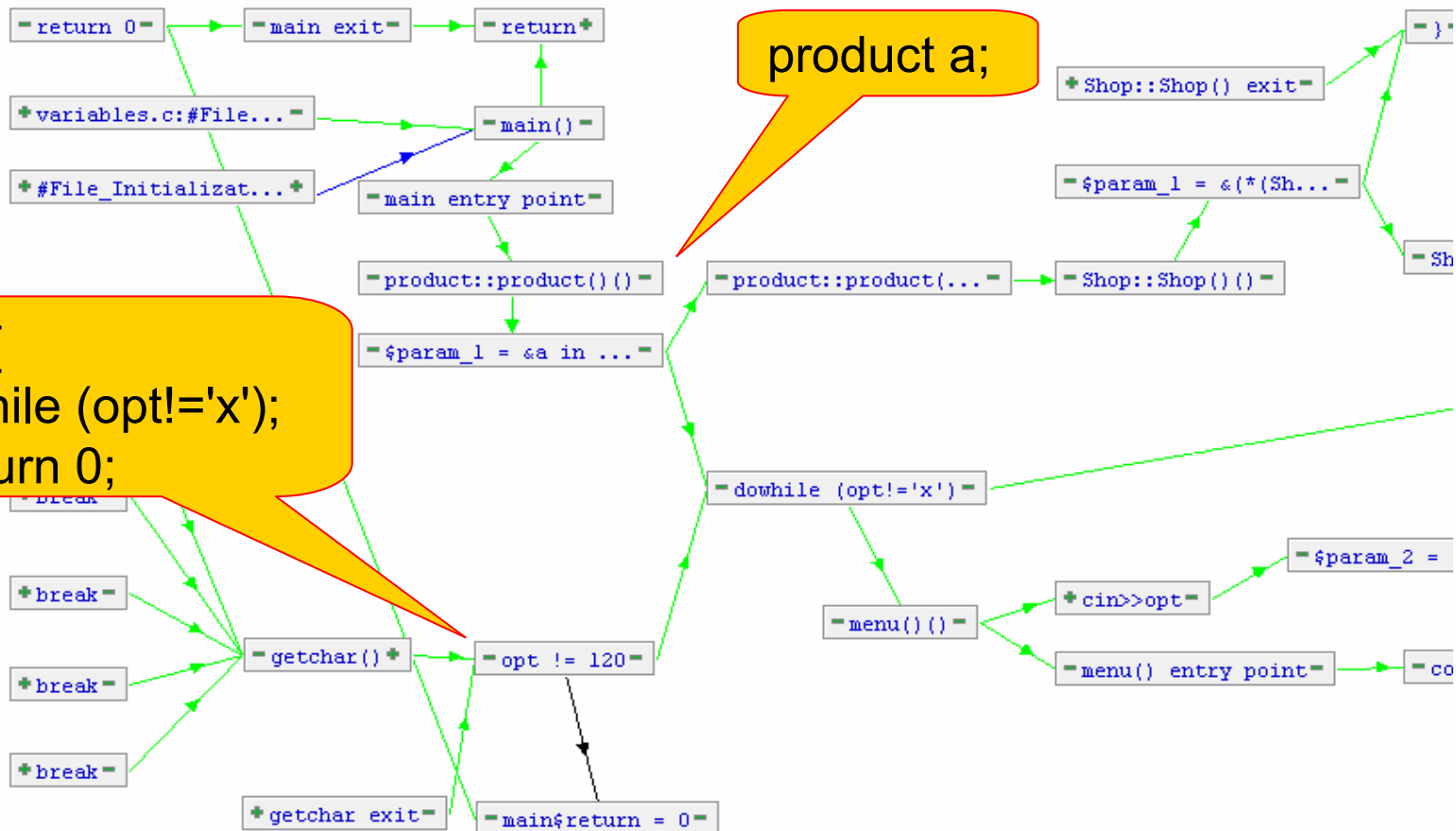
CodeSurfer(IV) - Static analyze

- Call graph for main function

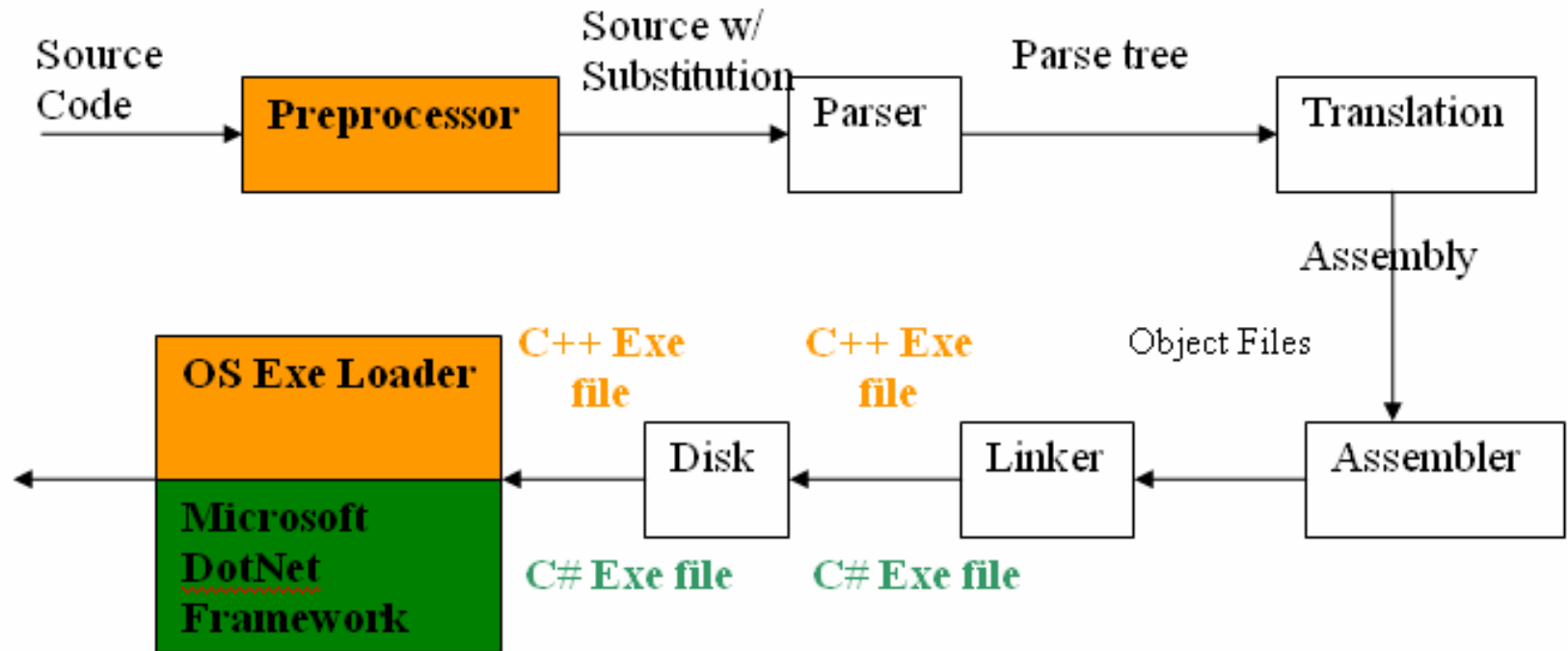


CodeSurfer(V) – Dynamic Analyze using Programming API

- Control flow graph (CFG) for main function



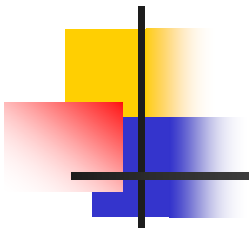
Reverse engineering binary software (I) – Compilation Process



Only for C++ Compiler from Microsoft



Only for C# Compiler



Reverse engineering binary software (II) - techniques

1. Disassembly
2. Decompilation
3. Analysis through observation of information exchange
4. Obtaining other different important information



Disassembly

- Executable files (machine code)-> assembly language
- Disassemblers:
 - **IDA Pro:** commercial (written by Ilfak Guilfanov), disassembles a wide array of different file types for different processors, for any popular operating system on the market
 - **Interactive Disassembler** – commercial, interactive
 - **Sourcer** – commercial, for x86 binaries (EXE, NE, PE)
 - **ProView (PVDasm)** - Pentium disassembler, PE editor, hex editor, process manager
 - **PE Explorer Disassembler** – commercial, for win32 PE files
 - **Win32 Program Disassembler** – free, straight line disassembler of Windows 32-bit executables

IDA Pro – Shopping.exe (I)

The screenshot displays the IDA Pro interface with the following components:

- Main Function:** A yellow callout points to the `proc near` instruction in the assembly code.
- Function names:** A yellow callout points to the `main` label in the assembly code.
- Main() {Product a:** A yellow callout points to the `call j_product_product` instruction.
- All strings:** A yellow callout points to the **Strings window** on the right, which lists memory addresses and their corresponding string values.

```
.text:004019A0 ; Attributes: bp-based frame
.text:004019A0
.text:004019A0 main          proc near          ; CODE XREF: j_main↑j
.text:004019A0                = dword ptr -0B8h
.text:004019A0 var_B8                = dword ptr -78h
.text:004019A0 var_78                = dword ptr -74h
.text:004019A0 var_74                = dword ptr -70h
.text:004019A0 var_70                = dword ptr -6Ch
.text:004019A0 var_6C
.text:004019A0
.text:004019A0                push    ebp
.text:004019A0                mov     ebp, esp
.text:004019A0                sub     esp, 0B8h
.text:004019A0                push   ebx
.text:004019A0                push   esi
.text:004019A0                push   edi
.text:004019A0                lea   edi, [ebp+var_B8]
.text:004019A0                mov   ecx, 2Eh
.text:004019A0                mov   eax, 0CCCCCCCCh
.text:004019A0                rep stosd
.text:004019A0                lea   ecx, [ebp+var_6C]
.text:004019A0                call  j_product_product
...
.text:004019B7
.text:004019BC
.text:004019BE
.text:004019C1
.text:004019C6
.text:004019C6 loc_4019C6:          ; CODE XREF: main+E1↓j
.text:004019C6                call  j_menu
.text:004019C6                lea   eax, [ebp+var_70]
.text:004019C6                push  eax
.text:004019C6                mov   ecx, offset cin
.text:004019C6                call  sub_403B50
.text:004019D4
```

Names window

Name
F product::SaleProduct
F product::ViewProducts
F menu
F main
F product::product
F Shop::Shop
F ShoppingCenter::ShoppingCenter
F endl
F flush

Line 13 of 830

Strings window

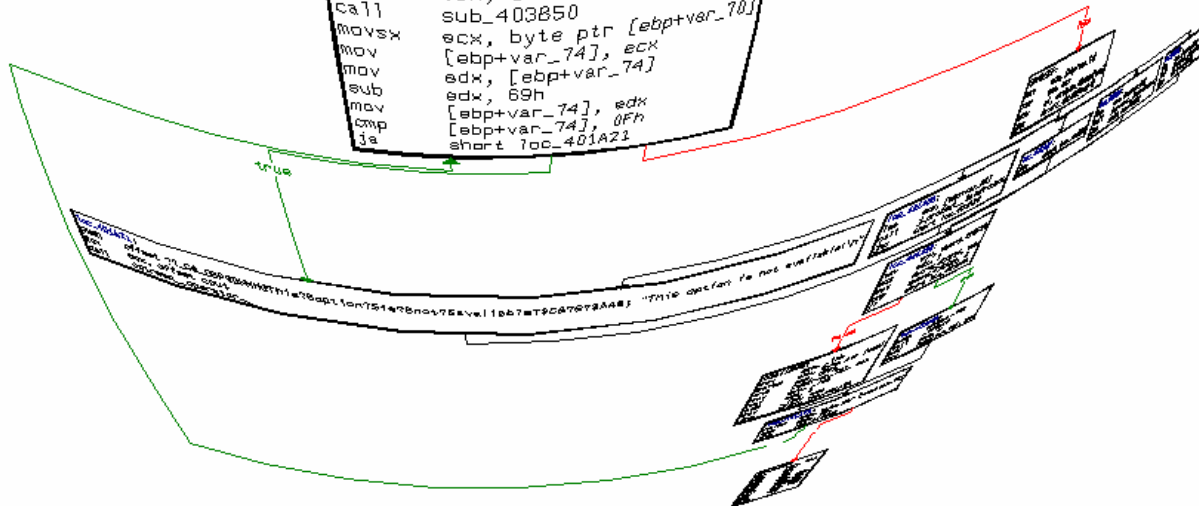
Address	Length	T...	Strin
"..." .text:00...	0000108E	C	
"..." .rdata:0...	00000024	C	\nPr
"..." .rdata:0...	00000018	C	---->P
"..." .rdata:0...	00000018	C	---->
"..." .rdata:0...	00000007	C	Error
"..." .rdata:0...	0000003A	C	---->
"..." .rdata:0...	00000016	C	---->
"..." .rdata:0...	00000014	C	---->
"..." .rdata:0...	00000039	C	\nGi

IDA Pro – Shopping.exe (II)

```
main:  
push    ebp  
mov     ebp, esp  
sub     esp, 0B8h  
push    ebx  
push    esi  
push    edi  
lea     edi, [ebp+var_B8]  
mov     ecx, 2Eh  
mov     eax, 0CCCCCCCCh  
rep    stosd  
lea     ecx, [ebp+var_6C]  
call   j_product__product
```

Flow chart
(Polar Fish
Eye)

```
loc_4019C6:  
call   j_menu  
lea    eax, [ebp+var_70]  
push  eax  
mov    ecx, offset cin  
call  sub_403850  
movsx ecx, byte ptr [ebp+var_70]  
mov    [ebp+var_74], ecx  
mov    [ebp+var_74], ecx  
sub    edx, [ebp+var_74]  
mov    edx, 69h  
mov    [ebp+var_74], edx  
cmp    [ebp+var_74], 0Fh  
je     short loc_401A21
```





Decompilation

- Executable files -> high level language source code
- Decompilers:
 - **Lutz Roeder's .NET Reflector**
 - **REC** - portable reverse engineering compiler for C code
 - **Dis# - .NET decompiler** - allows editing local variables
 - **dcc - Dos C Decompiler** - decompiles .exe files from the (i286, DOS) platform to C programs



Lutz Roeder's .NET Reflector

- Free software from <http://www.aisto.com/roeder/dotnet>
- class browser for .NET components
- assembly and namespace views
- type and member search
- XML documentation
- call and callee graphs
- IL, Visual Basic, Delphi and C# decompiler
- dependency trees with base type and derived type
- hierarchies and resource viewers

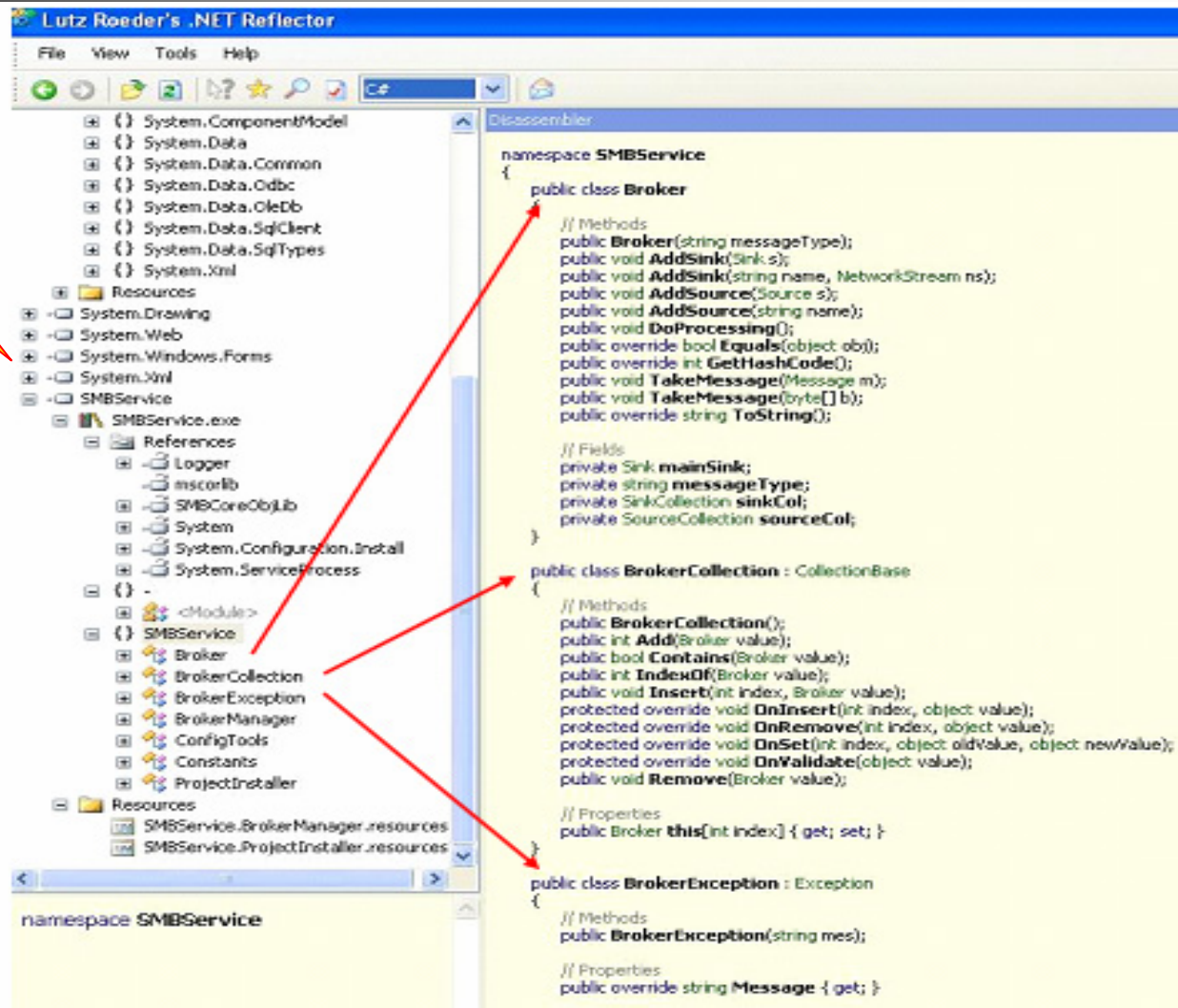
Other tools for reverse engineering



- **Bus analyzers**
- **Packet sniffers:** tcpdump, Ethereal, dSniff, [Netstat command](#)
- **Tools for System Wide Process Information -** [Sysinternals Process Explorer](#)
- **Tools for Viewing Filesystem Activity:** [Sysinternals Filemon](#), [Sysinternals Regmon](#)

SMBService.exe (I) – Simple Message Broker in C#

Project View



The screenshot shows the .NET Reflector interface. On the left, the Project View displays the structure of SMBService.exe, including references to System, System.Configuration, and System.ServiceProcess, and the SMBService namespace containing classes like Broker, BrokerCollection, and BrokerException. On the right, the Disassembler shows the C# code for the Broker class and its associated classes. Red arrows point from the Project View to the corresponding code in the Disassembler.

```
namespace SMBService
{
    public class Broker
    {
        // Methods
        public Broker(string messageType);
        public void AddSink(Sink s);
        public void AddSink(string name, NetworkStream ns);
        public void AddSource(Source s);
        public void AddSource(string name);
        public void DoProcessing();
        public override bool Equals(object obj);
        public override int GetHashCode();
        public void TakeMessage(Message m);
        public void TakeMessage(byte[] b);
        public override string ToString();

        // Fields
        private Sink mainSink;
        private string messageType;
        private SinkCollection sinkCol;
        private SourceCollection sourceCol;
    }

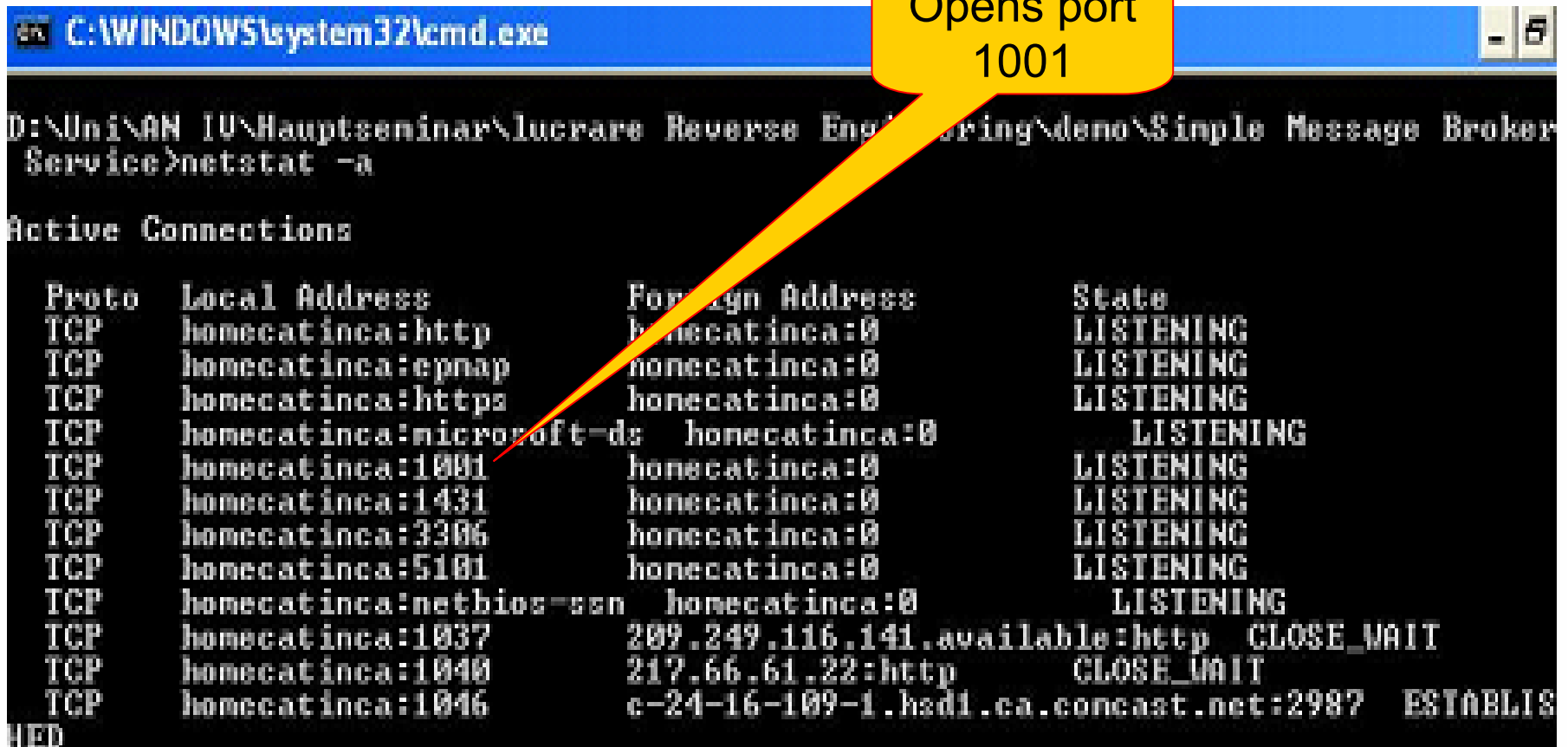
    public class BrokerCollection : CollectionBase
    {
        // Methods
        public BrokerCollection();
        public int Add(Broker value);
        public bool Contains(Broker value);
        public int IndexOf(Broker value);
        public void Insert(int index, Broker value);
        protected override void OnInsert(int index, object value);
        protected override void OnRemove(int index, object value);
        protected override void OnSet(int index, object oldValue, object newValue);
        protected override void OnValidate(object value);
        public void Remove(Broker value);

        // Properties
        public Broker this[int index] { get; set; }
    }

    public class BrokerException : Exception
    {
        // Methods
        public BrokerException(string mes);

        // Properties
        public override string Message { get; }
    }
}
```

SMBService.exe (I)- Netstat command

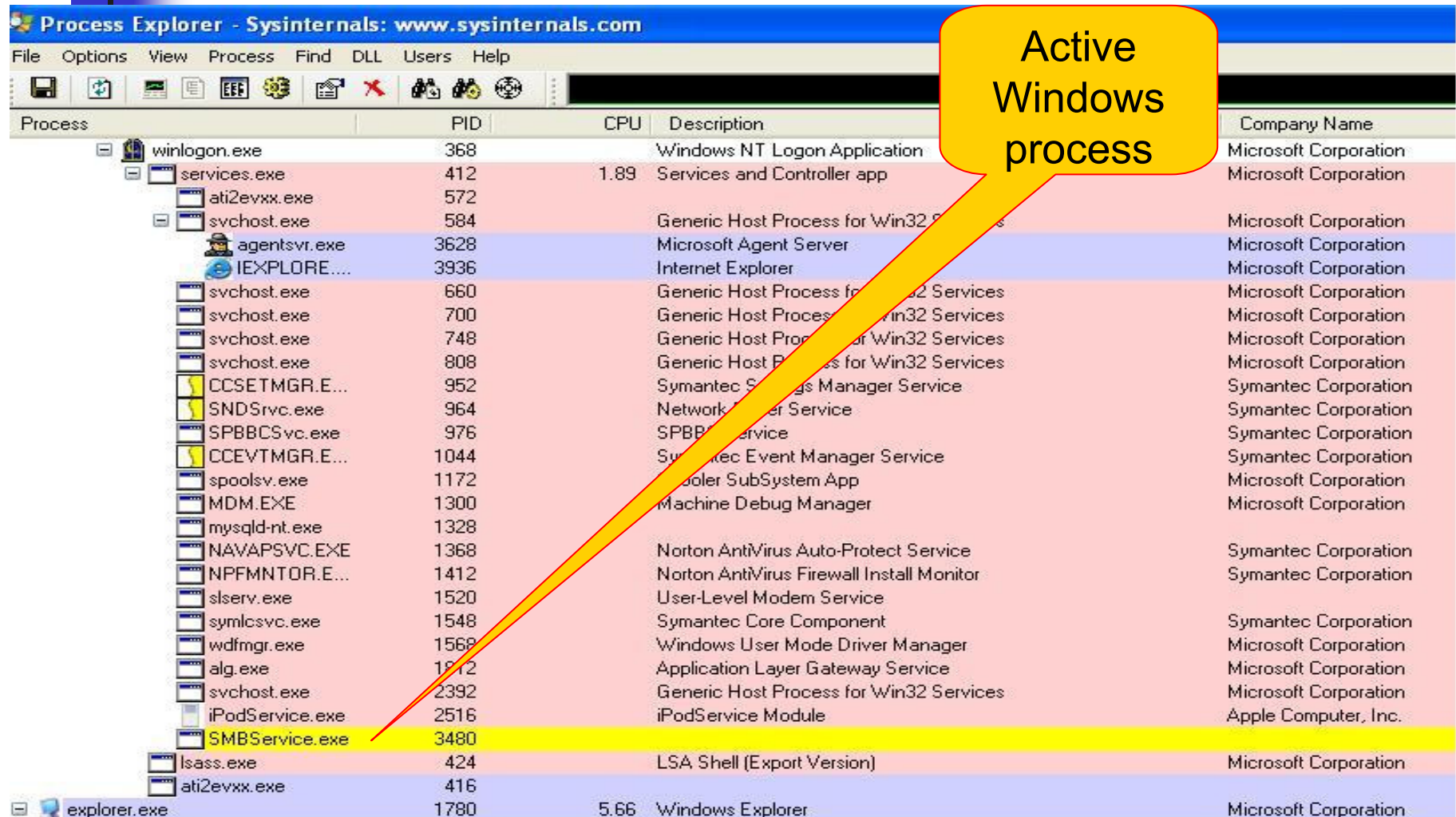


```
C:\WINDOWS\system32\cmd.exe
D:\Uni\AN 10\Hauptseminar\lucrare Reverse Engineering\deno\Simple Message Broker Service>netstat -a

Active Connections

Proto Local Address           Foreign Address         State
TCP    homecat.inca:http       homecat.inca:0         LISTENING
TCP    homecat.inca:epnap      homecat.inca:0         LISTENING
TCP    homecat.inca:https      homecat.inca:0         LISTENING
TCP    homecat.inca:microsoft-ds homecat.inca:0         LISTENING
TCP    homecat.inca:1001       homecat.inca:0         LISTENING
TCP    homecat.inca:1431       homecat.inca:0         LISTENING
TCP    homecat.inca:3306       homecat.inca:0         LISTENING
TCP    homecat.inca:5101       homecat.inca:0         LISTENING
TCP    homecat.inca:nethios-ssn homecat.inca:0         LISTENING
TCP    homecat.inca:1037       209.249.116.141:available:http CLOSE_WAIT
TCP    homecat.inca:1040       217.66.61.22:http      CLOSE_WAIT
TCP    homecat.inca:1046       c-24-16-109-1.hsdl.ca.comcast.net:2987 ESTABLIS
```

SMBService.exe (II) - Sysinternals Process Explorer



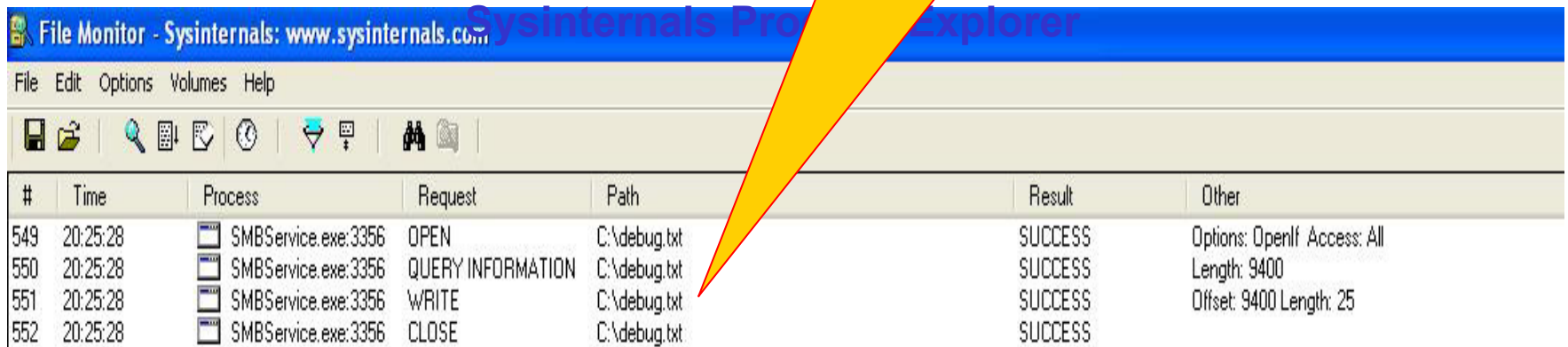
Process Explorer - Sysinternals: www.sysinternals.com

File Options View Process Find DLL Users Help

Process	PID	CPU	Description	Company Name
winlogon.exe	368		Windows NT Logon Application	Microsoft Corporation
services.exe	412	1.89	Services and Controller app	Microsoft Corporation
ati2evxx.exe	572			
svchost.exe	584		Generic Host Process for Win32 Services	Microsoft Corporation
agentsvr.exe	3628		Microsoft Agent Server	Microsoft Corporation
IEEXPLORE...	3936		Internet Explorer	Microsoft Corporation
svchost.exe	660		Generic Host Process for Win32 Services	Microsoft Corporation
svchost.exe	700		Generic Host Process for Win32 Services	Microsoft Corporation
svchost.exe	748		Generic Host Process for Win32 Services	Microsoft Corporation
svchost.exe	808		Generic Host Process for Win32 Services	Microsoft Corporation
CCSETMGR.E...	952		Symantec Settings Manager Service	Symantec Corporation
SNDSrv.exe	964		Network Driver Service	Symantec Corporation
SPBBCSvc.exe	976		SPBBC Service	Symantec Corporation
DCEVTMGR.E...	1044		Symantec Event Manager Service	Symantec Corporation
spoolsv.exe	1172		Printer SubSystem App	Microsoft Corporation
MDM.EXE	1300		Machine Debug Manager	Microsoft Corporation
mysqld-nt.exe	1328			
NAVAPSV.CE...	1368		Norton AntiVirus Auto-Protect Service	Symantec Corporation
NPFMNTOR.E...	1412		Norton AntiVirus Firewall Install Monitor	Symantec Corporation
slserv.exe	1520		User-Level Modem Service	
symlicsv.exe	1548		Symantec Core Component	Symantec Corporation
wdfmgr.exe	1568		Windows User Mode Driver Manager	Microsoft Corporation
alg.exe	1872		Application Layer Gateway Service	Microsoft Corporation
svchost.exe	2392		Generic Host Process for Win32 Services	Microsoft Corporation
iPodService.exe	2516		iPodService Module	Apple Computer, Inc.
SMBService.exe	3480			
lsass.exe	424		LSA Shell (Export Version)	Microsoft Corporation
ati2evxx.exe	416			
explorer.exe	1780	5.66	Windows Explorer	Microsoft Corporation

SMBService.exe (III) - Sysinternals Filemon

Writing file action
(C:/debug.txt)



File Monitor - Sysinternals: www.sysinternals.com

File Edit Options Volumes Help

#	Time	Process	Request	Path	Result	Other
549	20:25:28	SMBSservice.exe:3356	OPEN	C:\debug.txt	SUCCESS	Options: OpenIf Access: All
550	20:25:28	SMBSservice.exe:3356	QUERY INFORMATION	C:\debug.txt	SUCCESS	Length: 9400
551	20:25:28	SMBSservice.exe:3356	WRITE	C:\debug.txt	SUCCESS	Offset: 9400 Length: 25
552	20:25:28	SMBSservice.exe:3356	CLOSE	C:\debug.txt	SUCCESS	



Conclusions

- Reverse engineering code source or binary software
- Objectives:
 - Dealing with complexity
 - Generating alternate views
 - Recover lost information
 - Detect side effects
 - Synthesize higher abstractions
 - Facilitate reuse



Why reverse engineering?

- Because you can !
 - Have a goal!

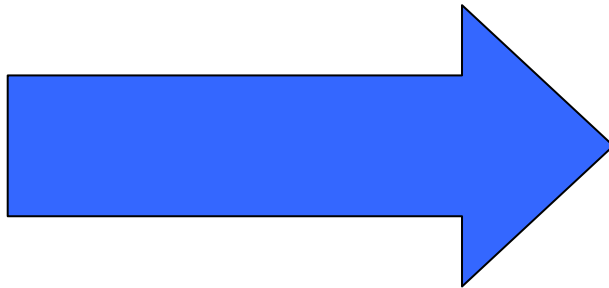


References

- [Wiki] Wikipedia, http://en.wikipedia.org/wiki/Reverse_engineering#fn_2_back
- [WiRe] http://en.wikibooks.org/wiki/Reverse_Engineering/Introduction
- [Chi] E.J. Chikofsky and J.H. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy" in IEEE Software, pp 13-17, IEEE Computer Society, January 1990
- [Rek] M.G.Rekoff, "On Reverse Engineering", IEEE Trans. Systems, Man and Cybernetics, pp.224-252, March – April 1985
- [War] R. Warden, "Re-engineering - a practical methodology with commercial applications in Software Reuse and Reverse Engineering in Practice", pp 283-305, Chapman & Hall, London, England, 1992
- [Lut] Lutz Roeder's Programming .NET, Reflector , <http://www.aisto.com/roeder/dotnet/>
- [CoS] CodeSurfer, User Guide and Technical References, <http://www.grammatech.com>
- [UnC] Understanding for C++ Help, <http://www.scitools.com>
- [Ida] IDA Pro Interactive Help, <http://www.datarescue.com/idabase/ida.htm>



Project Shopping Demo



CodeSurfer