determina™

Reverse Engineering and Computer Security

Alexander Sotirov

alex@sotirov.net

Introduction

determina

- Security researcher at Determina, working on our LiveShield product
- Responsible for vulnerability analysis and reverse engineering Microsoft patches
- Exploit development experience
- Speaker at CanSecWest, REcon, SyScan (Singapore) and BlackHat USA 2006

Overview

In the next hour, we will cover:

- Introduction to reverse engineering
  - computer underground
  - security industry
- Reverse engineering Microsoft patches
  - patch statistics
  - silently patched vulnerabilities
  - reverse engineering tools
  - patch analysis demo
- Developing third party security patches
  - recent developments
  - implementation details

Part I

Introduction to Reverse Engineering

Historical Perspective

- Widely used in the computer industry
  - IBM PC clones
  - Sony vs. Accolade
  - Linux wireless drivers, Samba, IM clients
- Adopted by software crackers, virus writers and exploit developers with the rise of personal computers in the 1980s
- Invaluable in the AV and security industries

Reverse Engineering in the Computer Underground

determina

- Cracking software copy protection
  - PC software and games
  - modding the Xbox and Playstation
- Exploit development
  - hackers, botnets, spyware
- Reversing undocumented DOS and Windows API
  - virus writers
  - spyware, keyloggers, malware

Reverse Engineering in the Security Industry

- Virus and malware analysis
  - AV and Anti-Spyware companies
- Patch analysis, vulnerability analysis
  - IDS, IPS companies
- Binary code auditing
  - discovering new vulnerabilities
- Exploit development
  - penetration testing
- Interoperability and undocumented APIs
  - kernel drivers for firewalls, HIPS, other low level software

Recent Developments

- Increased demand in the security industry
- Reverse engineering tools have matured
- New tools designed by people in the security community
  - BinDiff by SABRE Security
  - Free IDA plugins by researchers at iDefense, TippingPoint, Determina and others
- More complicated exploitation techniques (heap overflows, uninitialized variables) require the use of reverse engineering
- Closed-source applications on Windows systems are the primary target of attacks

Part I

Reverse Engineering Microsoft Patches

Patch Statistics

- More than 400 security bulletins since 1998
- Most updates often address multiple vulnerabilities
    - 9 vulnerabilities in the latest IE patch
- Microsoft's definition of vulnerability: As long as the problems are in the same DLL file, it's a single vulnerability
    - 30 different integer overflows in the WMF parser were fixed by MS05-053 as a single vulnerability
- The technical information in the security bulletins is so minimal, it's practically useless:

    There is a privilege elevation vulnerability in Windows 2000 caused by improper validation of system inputs. This vulnerability could allow a logged on user to take complete control of the system.

Skeletons in Microsoft's Closet

- Security issues are often fixed silently
    - security researcher reports a vulnerability
    - Microsoft audits the affected code and discovers 5 related bugs
    - 6 bugs are fixed in the patch
    - security bulletin describes only the first bug
- Service packs always fix security issues, with no public announcements
- The users and the security community are left in the dark about the what the real issues are

Withholding information leads to…

Tuesday, July 12, 2006

- MS06-035 describes a remotely exploitable vulnerability in the Mailslot network interface
- no additional details in the security bulletin
- CORE Security develops an exploit which crashed the kernel by sending a Mailslot request
- exploit released to their customers on the same day

Wednesday, July 13, 2006

- turns out the exploit crashes fully patched systems
- CORE Security had discovered and exploited a new unpatched bug, not the one fixed in MS05-035 (Ooops!)

Microsoft patch expected in November.

Patch Analysis

- Everybody in IDS and IPS industry does it
- You have to understand the bugs to be able to detect attacks or protect against them
- People are interested in:
    - attack vectors and packet signatures - for network based IDS and IPS systems
    - the type of vulnerability (stack overflow, heap overflow, non-memory corruption) - for host based IPS systems
    - remote detection of the vulnerability - for vulnerability assessment
    - exploit development - for testing IDS and IPS systems, and penetration testing (CORE IMPACT, Canvas, Metasploit)

Week
My Patch ~~Tuesday~~

Go to bed early the day before

10am

- Patches are released between 10-11am
- Start downloading patches, read the security bulletins in the meantime

11am

- Prioritize the patches based on the available information (at this point it's just guessing)
- Use IDA Pro to disassemble the patches and BinDiff to compare them against the unpatched files
- update the priorities as we learn more

  ...

1am

- Go home, rest, repeat the next day

Reverse Engineering Tools

- IDA Pro
  - its plugin API is turning IDA into a reverse engineering platform that other tools depend on
- BinDiff
  - invaluable for binary patch analysis
- WinDbg
  - good support for debugging symbols, command line interface, frequent updates
- SoftICE
  - great for debugging code between userspace and the kernel
- VMWare
  - Workstation 5 supports multiple snapshots
  - GSX and Server provide a scripting API
  - Workstation 5.5 can be controlled with a command line tool

Binary Database

We have an internal database of binaries indexed by the name and SHA1 hash of the file. We store the following file metadata:

• name	*ntdll.dll*
• size	*654336 bytes*
• modification date	*May 01, 2003, 3:56:12 PM*
• SHA1 hash	*9c3102ea1d30c8533dbf5d9da2a47…*
• debugging symbols	*Sym/ntdll.pdb/3E7B64D65/ntdll.pdb*
• source of the file	
○ product	*Windows XP*
○ version	*SP1*
○ security update	*MS03-007*
○ build	*qfe*
○ comment	

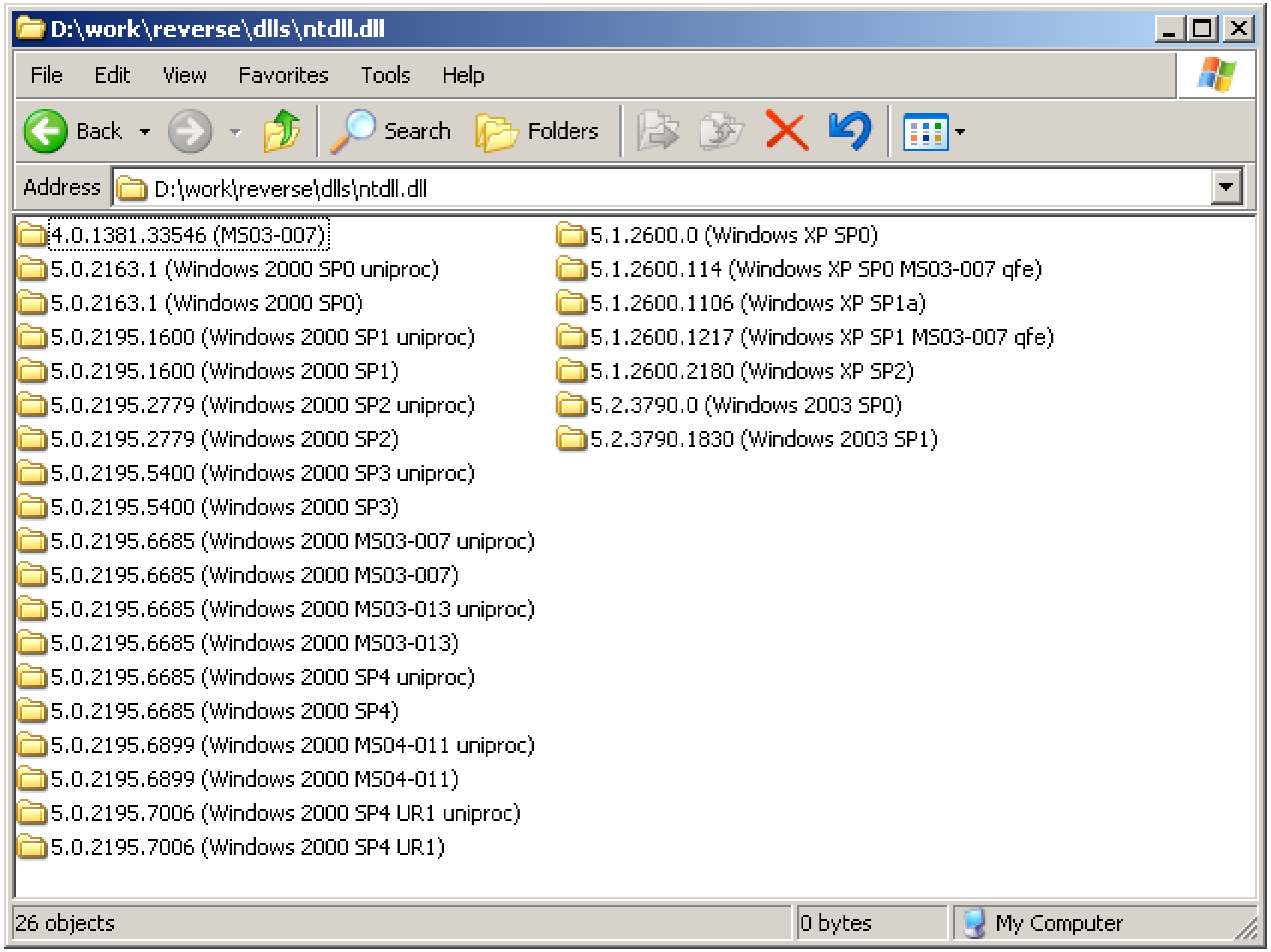
Binary Database

Current size of our database, including all service packs and security updates for Windows 2000, XP and 2003:

- 30GB of files
- 7GB of symbols
- 7500 different file names
- 28800 files total

and growing…

Binary Database


D:\work\reverse\dlls\ntdll.dll
File Edit View Favorites Tools Help
Back
Search
Folders
Address

D:\work\reverse\dlls\ntdll.dll

4.0.1381.33546 (MS03-007)
5.0.2163.1 (Windows 2000 SP0 uniproc)
5.0.2163.1 (Windows 2000 SP0)
5.0.2195.1600 (Windows 2000 SP1 uniproc)
5.0.2195.1600 (Windows 2000 SP1)
5.0.2195.2779 (Windows 2000 SP2 uniproc)
5.0.2195.2779 (Windows 2000 SP2)
5.0.2195.5400 (Windows 2000 SP3 uniproc)
5.0.2195.5400 (Windows 2000 SP3)
5.0.2195.6685 (Windows 2000 MS03-007 uniproc)
5.0.2195.6685 (Windows 2000 MS03-007)
5.0.2195.6685 (Windows 2000 MS03-013 uniproc)
5.0.2195.6685 (Windows 2000 MS03-013)
5.0.2195.6685 (Windows 2000 SP4 uniproc)
5.0.2195.6685 (Windows 2000 SP4)
5.0.2195.6899 (Windows 2000 MS04-011 uniproc)
5.0.2195.6899 (Windows 2000 MS04-011)
5.0.2195.7006 (Windows 2000 SP4 UR1 uniproc)
5.0.2195.7006 (Windows 2000 SP4 UR1)
5.1.2600.0 (Windows XP SP0)
5.1.2600.114 (Windows XP SP0 MS03-007 qfe)
5.1.2600.1106 (Windows XP SP1a)
5.1.2600.1217 (Windows XP SP1 MS03-007 qfe)
5.1.2600.2180 (Windows XP SP2)
5.2.3790.0 (Windows 2003 SP0)
5.2.3790.1830 (Windows 2003 SP1)

26 objects | 0 bytes | My Computer

Any Vendors Reading This?

This is my advice to vendors who want to help security researchers:

- provide an accurate list of all security updates, hotfixes and other patches (preferably in XML format)
- make older releases of your software available
- don't combine security patches with other updates
- have a consistent naming and versioning system
- do not update software without updating its version or build number
- provide debugging symbols for all binaries
- release detailed descriptions of all vulnerabilities

Microsoft does almost all of the above, except for the last item.

BinDiff Demo

MS06-033

ASP.NET Information Disclosure

Part II

Third Party Security Patches

A Recent Development

Dec 2005

- WMF patch by Ilfak Guilfanov, author of IDA Pro

Mar 2006

- IE createTextRange patch by eEye
- IE createTextRange patch by Determina

WMF Vulnerability

The vulnerability was already actively exploited and used for malware distribution when it was first made public on Dec 27. It took Microsoft 10 days to release a patch.

Ilfak Guilfanov released an unofficial patch on Dec 31.

- injected into all processes with the AppInit_DLLs registry key
- patches an exported function in GDI32.DLL
- aborts the processing of WMF records containing executable code

IE createTextRange Vulnerability

Privately reported to Microsoft in February, publicly disclosed on Mar 22. Microsoft did not release patch until Apr 11. Two third-party patches were released independently of each other on Mar 27 by eEye and Determina.

eEye:

- creates a copy of JSCRIPT.DLL and patches it on disk
- uses pattern matching to find the code to patch
- modifies the registry to force IE to use of the patched file

Determina:

- patches MSHTML.DLL in memory
- hardcoded patch points for 98 different versions of MSHTML
- on most versions, modifies a single byte in the vulnerable function

Why Use Third-Party Patches?

Advantages:

- Availability before the official patch
- Vulnerability based, not exploit-dependent
- Not vulnerable to evasion, unlike network based IPS

Disadvantages:

- Limited patch QA process
- Limited support for multiple OS versions and languages
- Some vulnerabilities require extensive changes or redesign of the affected application and cannot be hotpatched

Third-party patches are ideal for situations where the risk of a system compromise outweighs the risk of interoperability issues.

Part III

Implementing Hotpatching

What Is Hotpatching?

Hotpatching is a method for modifying the behavior of an application by modifying its binary code at runtime. It is a common technique with many uses:

- debugging (software breakpoints)
- runtime instrumentation
- hooking Windows API functions
- modifying the execution or adding new functionality to closed-source applications
- deploying software updates without rebooting
- fixing security vulnerabilities

In-Place Code Modification

determina

Modifying the instructions of the program by overwriting allows us to remove and change the instructions of a program:

- removing code

  overwrite the instructions with NOPs

- changing code

  overwrite an instruction with another instruction of the same or smaller size

```
call check_license              call check_license
jnz valid                       jz valid
```

- adding code

  not possible

5-Byte JMP Overwrite

determina

The most common approach to hooking functions on Windows is to overwrite the function prologue with a 5-byte JMP instruction that transfers the execution to our code.

Original code:

```
55                push  ebp
8B EC             mov   ebp, esp
53                push  ebx
56                push  esi
8B 75 08          mov   esi, [ebp+arg_0]
```

Patched code:

```
E9 6F 02 00 00    jmp   hook
8B 75 08          mov   esi, [ebp+arg_0]
```

5-Byte JMP Overwrite

Before overwriting the function prologue, we need to save the overwritten instructions. The hook routine should execute the saved instructions before returning to the patched function.

Patched function:

```
jmp    hook
mov    esi, [ebp+arg_0]
...
ret
```

Hook routine:

```
// do some work
...
// saved instructions
push   ebp
mov    ebp, esp
push   ebx
push   esi
// return
jmp    patched_function
```

Microsoft Hotpatching

The Microsoft hotpatching implementation is described in US patent application 20040107416. It is currently supported only on Windows 2003 SP1, but we'll probably see more of it in Vista.

The hotpatches are generated by an automated tool that compares the original and patched binaries. The functions that have changed are included in a file with a .hp.dll extension. When the hotpatch DLL is loaded in a running process, the first instruction of the vulnerable function is replaced with a jump to the hotpatch.

The /hotpatch compiler option ensures that the first instruction of every function is a `mov edi, edi` instruction that can be safely overwritten by the hotpatch. Older versions of Windows are not compiled with this option and cannot be hotpatched.

Conclusion

- Reverse engineering can be used for both good and bad:
  - find vulnerabilities
  - analyse patches
  - write exploits and malware
  - develop security patches
- If the bad guys are using it, we have to use it too to protect ourselves.

Questions?

alex@sotirov.net