Professional Rootkits

Ric Vieler

Wiley Publishing, Inc.
Professional

Rootkits
About the Author

Ric Vieler is a software engineer and a certified ethical hacker. He enjoys writing both technical manuals (such as Professional Rootkits) and science fiction novels (such as Spliced, Acknowledge, and A Stitch in Time). His love of the unexplored, mixed with a thorough understanding of computer internals, has culminated in a career that fully embraces both: professional hacking. When not hacking, reading, or writing, Ric spends his spare time with his wife, Lisa, and their two children, Samantha and Dylan.
Credits

Executive Editor  
Carol Long

Development Editor  
Ed Connor

Technical Editor  
John Heasman

Production Editor  
William A. Barton

Copy Editor  
Luann Rouff

Editorial Manager  
Mary Beth Wakefield

Production Manager  
Tim Tate

Vice President and Executive Group Publisher  
Richard Swadley

Vice President and Executive Publisher  
Joseph B. Wikert

Graphics and Production Specialists  
Brooke Graczyk  
Joyce Haughey  
Alicia B. South

Quality Control Technicians  
Christine Pingleton  
Charles Spencer

Project Coordinator  
Erin Smith

Media Development Specialists  
Angie Denny  
Kit Malone  
Kate Jenkins  
Steve Kudirka

Proofreading and Indexing  
Christine Sabooni and Estalita Slivoskey

Anniversary Logo Design  
Richard J. Pacifico
Contents

Introduction xv

Chapter 1: Tools 1

How Do I Build a Rootkit? 1
The Microsoft Driver Development Kit 4
Microsoft Visual VC++ 2005 Express 5
Microsoft Software Developers Kit 5
Sysinternals Freeware 5
IDA 6
Debugging Tools for Windows 6
Verification 6
VCVARS32.BAT 7
Other Tools to Consider 7
What to Keep Out 7
Summary 8

Chapter 2: A Basic Rootkit 9

Ghost 9
   Ghost.h 9
   Ghost.c 10
   configManager.h 13
   configManager.c 14
Alternate Data Streams 15
   fileManager.h 16
   fileManager.c 17
Installing Your Rootkit 21
   SCMLoader.c 22
   SCMUUnloader.c 25
Testing Your Rootkit 26
Summary 26

Chapter 3: Kernel Hooks 27

The System Call Table 27
Kernel Memory Protection 28
Contents

IoManager.h 106
buildController.bat 107
Handling IO within the Device Driver 107
IoManager.c 110
SOURCES 112
Injected Function Programming 114
Testing I/O Control 114
Summary 117

Chapter 6: Communications 119

The Transport Driver Interface 119
Initiating the Connection 120
An Example 120
commManager.h 121
commManager.c 122
SOURCES 130
Running the Example 133
Summary 135

Chapter 7: Filter Drivers 137

Inserting a Filter Driver 137
File Filtering 138
Network Filtering 139
Combined Filtering 140
An Example 141
filterManager.h 142
filterManager.c 142
Ghost.c 146
IoManager.h 150
IoManager.c 154
SOURCES 166
Summary 166

Chapter 8: Key Logging 167

Processing Levels 167
IRQL = PASSIVE_LEVEL 168
IRQL = APC_LEVEL 168
IRQL = DISPATCH_LEVEL 168
IRQL = DIRQL 168
Contents

A Keyboard Filter 168
Threading and Synchronization 170
Interpreting Key Codes 170
An Example 171
SOURCES 172
Ghost.c 172
filterManager.c 173
filterManager.h 174
IoManager.c 174
keyManager.h 174
keyManager.c 176
OnKeyboardRead 184
OnReadCompletion 184
GetKey 184
InitializeLogThread 184
KeyLoggerThread 185
StartKeylogger 185
StopKeylogger 185
OnCancel 185
Testing the Example 185
Summary 186

Chapter 9: Concealment 187

Registry Key Hiding 187
registryManager.h 188
registryManager.c 189
Ghost.c 198
hookManager.h 198
hookManager.c 199
Directory Hiding 203
Process Hiding 205
HideMe.c 206
Testing Concealment 211
Summary 212

Chapter 10: E-mail Filtering 215

Microsoft Outlook E-mail Filtering 215
OutlookExtension.h 216
OutlookExtension.cpp 218
Contents

| Installing an Outlook Client Filter | 231 |
| Testing the Outlook Client Extension | 231 |
| Lotus Notes E-mail Filtering | 232 |
| LotusExtension.h | 234 |
| LotusExtension.c | 234 |
| LotusExtension.def | 240 |
| LotusExtension.mak | 240 |
| readme.txt | 241 |
| Installing a Lotus Notes Client Filter | 241 |
| Testing the Lotus Notes Client Extension | 242 |
| Summary | 242 |

Chapter 11: Installation Considerations 243

| Intended Installation | 243 |
| Intended Installation Software | 244 |
| End User License Agreements (EULAs) | 244 |
| Unintended Installation | 245 |
| Privilege Escalation | 245 |
| Persistence | 245 |
| ZwSetSystemInformation with SystemLoadAndCallImage | 246 |
| Registry Possibilities | 247 |
| Initialization Files | 248 |
| Installing onto Machines That Visit Your Website | 249 |
| Removing the Traces of an Installation | 251 |
| Testing Your Installation Techniques | 254 |
| Summary | 254 |

Chapter 12: Ghost Tracker 255

| The Controller | 255 |
| The Connection | 257 |
| Tamper Detection | 257 |
| An Example | 258 |
| GhostTracker.cs | 260 |
| ControlForm.cs | 262 |
| TargetController.cs | 268 |
| Listen.cs | 270 |
| GhostTracker | 273 |
| ControlForm | 273 |
| Summary | 274 |
Contents

**Chapter 13: Detecting Rootkits** 275

Detection Methods 275
Detection Software 279
  Strider GhostBuster 280
  RootkitRevealer 280
  F-Secure Blacklight 281
  RootKit Hook Analyzer 282
  IceSword 283
  Sophos Anti-Rootkit 286

What to Do with a Detected Rootkit 287
Safe Mode 289
Summary 290

**Chapter 14: Preventing Rootkits** 291

Operating System Updates 292
Automatic Updates 292
Personal Firewalls 293
Free Personal Firewalls 294
  Tiny Personal Firewall 294
  Zone Alarm Firewall 294
  Sygate Personal Firewall 294
Other Personal Firewalls 294
  Kerio Personal Firewall 294
  Symantec/Norton Firewall 295
  Zone Alarm Professional Firewall 295
Host-based Intrusion Prevention Systems 295
Hardening 295
Virtualizing 297
Blocking Unexpected Operations 298
Rootkit Prevention Techniques 298
  Kernel Hook Prevention 298
  Service Load Prevention 298
  Driver Load Prevention 298
  Code in Data Segment Prevention 299
  Stack Execution Prevention 299
Summary 299

**Appendix A: Freeware** 301

Index 317
Introduction

What Is a Professional Rootkit?

Hacker Defender, HE4Hook, Vanquish, NT Rootkit, FU, AFX Rootkit—these are the names of some of the rootkits that have found their way into millions of computers around the world. These rootkits share many similarities. Each was written by a single programmer or at best a few programmers. Each can provide unauthorized access to information. Each uses some form of stealth to avoid detection, and they all use technology not intended for their purpose.

What is the purpose of a rootkit? Many exist for the sole purpose of programmer recognition. Some find use as “botnet” clients that can work together to overload a particular site. Some end up distributing the spam we find in our e-mail every morning. Some provide conduits to otherwise secure networks. Others are used to gather our personal information for fun and profit. These purposes limit the final product to what I will term “casual software”—more precisely, software limited by purpose, where that purpose is illegal, unethical, destructive, or simply unprofitable. Software developed under these guises cannot usually garner the tools, materials, and expertise required to produce commercial-grade software.

This book looks beyond the casual rootkit into the emerging field of professional rootkits.

Webster defines “professional” as characterized by or conforming to the technical or ethical standards of a profession. In this case, the profession is software engineering, where the technical and ethical standards maintained by professional software engineers can produce commercial-grade products with capabilities far beyond those of a single programmer working in his or her spare time on an illegal or unprofitable project. In the case of rootkits, a well-funded team of professionals can exploit a broad range of technology to produce full-featured, robust software with extensive capabilities that are each fit-for-purpose.

Here we are again, back at purpose. Only now, with Professional Rootkits, the purpose is information leak prevention, content monitoring and filtering, copyright infringement prevention, or any similar need funded by a multimillion-dollar industry. These industries are looking for solutions, and some of those solutions require the stealth and functionality that can only be found in Professional Rootkits.

Who This Book Is For

It might appear to be a simple question, but the audience for a book about rootkit development includes more than just rootkit developers. There are software developers interested in writing code that is not adversely affected by the possibility of rootkits. There are information technology (IT) specialists who need to understand every threat that can adversely affect their networks. There are many levels of security specialists who need to understand every possible threat posed by rootkits. There are the rootkit users who need to understand what a rootkit can do and how to use it. And let’s not forget the software engineers developing anti-rootkit technology.
Introduction

Cumulatively, if you are involved in the development of interconnected software, or computer security, or just want to learn a really cool way to enhance your operating system, then this book is for you.

What This Book Covers

This book details and implements all the major components of modern rootkits and provides the ancillary programs required to load, unload, configure, test, and control those rootkits. In addition, the book begins with a chapter on building a rootkit toolkit and ends with chapters on detecting and preventing rootkits. This overall coverage is designed to provide the reader with a complete understanding of rootkit capabilities, the technology used by rootkits, the tools used to develop and test rootkits, and the detection and prevention methods used to impede the distribution of rootkits.

How This Book Is Structured

Let’s face it, developing a rootkit is a difficult task, made more difficult by the lack of structured and reliable example code from which to learn and develop. The rootkit industry is only now transitioning from individual hackers to teams of professional software engineers, leaving the professional rootkit developer with the task of starting from scratch to create robust, modular, commercial-grade software. This book is structured to offer the professional rootkit developer a detailed and robust code base from which to begin.

After detailing the tools required to build the examples in this book, nine chapters are devoted to developing a basic rootkit and progressively adding features to that rootkit. This modular approach to rootkit functionality introduces rootkit technology in steps, enabling the code base to be easily customized to provide only the features required by a specific design.

Following rootkit development are chapters on installing, controlling, detecting, and preventing rootkits. These chapters are of extreme importance to the rootkit designer and the security specialist, two professions adversely opposed to one another, but sharing the same need to learn as much about rootkits as possible.

What You Need to Use This Book

The software required to compile the examples in this book is freely available and fully detailed in Chapter 1. In addition, the book has been designed to allow the reader to follow along using the precompiled executables provided for the Windows 2000, XP, and 2003 operating systems. There are no specific requirements beyond the optional software. It would be helpful to know the C programming language, but the code is commented and documented to the point where knowledge of the language is not required. It would also be helpful to have an understanding of Windows device drivers, but if you don’t, this book should make an excellent primer.

Though the x86 assembly language and the C# programming language are used in this book, there is no requirement to understand either. In general, the book has been designed to require as little as possible from the reader, yet it allows for complete interaction at every stage of rootkit implementation.
Introduction

Conventions

To help you get the most from the text and keep track of what's happening, several conventions are used throughout the book.

Tips, hints, tricks, and asides to the current discussion are offset and placed in italics like this.

As for styles in the text:

- We highlight new terms and important words when we introduce them.
- We show filenames, URLs, and code within the text like so: `persistence.properties`.

We present code in two different ways:

- In code examples we highlight new and important code with a gray background.

  The gray highlighting is not used for code that's less important in the present context, or has been shown before.

Source Code

As you work through the examples in this book, you may choose either to type in all the code manually or to use the source code files that accompany the book. All of the source code used in this book is available for download at [www.wrox.com](http://www.wrox.com). Once at the site, simply locate the book’s title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book’s detail page to obtain all the source code for the book.

Because many books have similar titles, you may find it easiest to search by ISBN; this book’s ISBN is 978-0-470-10154-4.

Once you download the code, just decompress it with your favorite compression tool. Alternately, you can go to the main Wrox code download page at [www.wrox.com/dynamic/books/download.aspx](http://www.wrox.com/dynamic/books/download.aspx) to see the code available for this book and all other Wrox books.

Before you extract the *Professional Rootkits* archive, you will need to disable any anti-virus software running on your machine. Most (if not all) of the material covered in this book is considered harmful. As such, any good anti-virus software will do everything it can to prevent the transfer of these harmful files to your computer, so you will need to disable your anti-virus software or exclude your rootkit files from anti-virus protection whenever you are working with known rootkits. Of course, you will be writing never-before-seen rootkits soon, but until then, you will need to disable or reconfigure your anti-virus software.

If you don’t already have an archive tool to unzip the Wrox/Wiley download file, I recommend ZipCentral. This is absolutely free software offered without banners, spyware, or a 30-day trial. Just download `zcsetup.exe` from [http://hemsidor.torget.se/users/z/zcentral/down.html](http://hemsidor.torget.se/users/z/zcentral/down.html) and double-click it to install ZipCentral. Once installed, you can drag and drop archives onto the application or its desktop icon, and extract these archives to the directory of your choosing.
Introduction

The extracted archive files are divided into individual chapters containing the source code, executables, and drivers mentioned in that chapter. If you intend to follow along without compiling the examples in these chapters, you can use the binaries provided for each chapter on any Windows 2000, XP, or 2003 operating system. If you intend to compile (and hopefully adapt) the source code provided, you will first need to build a rootkit toolkit, which is the topic of Chapter 1.

Errata

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, such as a spelling mistake or a faulty piece of code, we would be very grateful for your feedback. By sending in errata you may save another reader hours of frustration, and at the same time you will be helping us provide even higher quality information.

To find the errata page for this book, go to www.wrox.com and locate the title using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list, including links to each book's errata, is also available at www.wrox.com/misc-pages/booklist.shtml.

If you don't spot "your" error on the Book Errata page, go to www.wrox.com/contact/techsupport.shtml and complete the form there to send us the error you have found. We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in a subsequent edition of the book.

p2p.wrox.com

For author and peer discussion, join the P2P forums at p2p.wrox.com. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and to interact with other readers and technology users. The forums offer a subscription feature to e-mail you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At http://p2p.wrox.com you will find a number of different forums that will help you not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to p2p.wrox.com and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join as well as any optional information you wish to provide and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.
Introduction

You can read messages in the forums without joining P2P but in order to post your own messages, you must join.

Once you join, you can post new messages and respond to messages other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum e-mailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.
This chapter stresses the importance of building and saving the tools required for rootkit development. Building a full-featured rootkit toolkit before you begin development enables you to research, design, develop, test, and package your rootkit without distraction. In addition, saving the tools, utilities, samples, scripts, and even the failed experiments enables you to pick up where you left off at any time. As an example, the rootkit presented in this book was originally developed and forgotten several years ago, but came to mind when I was contacted by Wiley, the publisher. Having the code, the scripts, the utilities, and a copy of the toolkit used to develop the rootkit, all in one convenient archive, turned an otherwise complex project into a delightful experience.

This chapter includes the following:

- What must go into a rootkit toolkit
- What should go into a rootkit toolkit
- How to verify the usefulness of your rootkit toolkit

How Do I Build a Rootkit?

Assembling a complete rootkit toolkit will take a lot of time. Fortunately, everything you need to get started can be downloaded from Microsoft (http://msdn2.microsoft.com/en-us/default.aspx). The three most important tools you need are the Microsoft Driver Development Kit (DDK), a C compiler, and the Windows Platform Software Development Kit (SDK). Fortunately, these can all be downloaded from Microsoft without cost.

Though the Visual C++ compiler and the Software Development Kit (SDK) can be downloaded directly, the Driver Development Kit (DDK) can only be downloaded as an ISO image (unless you happen to have a Microsoft MSDN subscription). At the time of this writing, you can get the ISO image from www.microsoft.com/whdc/devtools/ddk/default.mspx. This image can be
transferred to a CD using the “record a disk from a disk image” feature of your CD burning software. If you do not have the capability to burn a CD from an ISO image, and you don’t have (or know someone who has) a Microsoft MSDN subscription, you can order the Windows Server 2003 SP1 DDK CD at no cost (other than a small shipping and handling fee) from www.microsoft.com/whdc/devtools/ddk/orderdckd.mspx.

Currently, Microsoft Visual C++ 2005 Express is available for download, free of charge, from http://msdn.microsoft.com/vstudio/express/visualc/download. This development environment has everything needed to develop basic Windows applications. In addition, Visual C++ 2005 Express has a C compiler that will enable you to create the console programs needed to load, unload, and test the rootkits developed in this book.

The console programs you will be creating are native Win32 programs, so you will also need to download and install the Microsoft Windows Platform SDK separately. The SDK (PSDK-x86.exe) can currently be downloaded from www.microsoft.com/downloads/details.aspx?FamilyId=A55B6B43-E24F-4E8A3-A93B-40C0EC4F68B5&displaylang=en#filelist.

Combined, the Driver Development Kit, the Visual C++ compiler (or any Windows-compatible C compiler) and the Platform SDK will enable you to follow along with, compile, and run every example in this book. There are, however, several utilities that will make rootkit development much easier, the first of which is DebugView. This utility enables you to see debugging statements while executing rootkits. Though this is not technically a necessity for rootkit development, I can’t imagine writing a rootkit without it. In addition to DebugView, the good folks at Sysinternals also provide Diskmon, Filemon, and Regmon, three utilities that enable you to monitor disk activity, file system activity, and registry activity, respectively. You’ll want to have these in your toolkit as well. If you have downloaded the source code for this book, you will find individual archives for each of these utilities under “Chapter 1 Tools.”

If you want to delve deeply into the technology behind rootkits, you will also want to get a copy of IDA. IDA is the reverse-engineering tool that will be used in Chapter 4 to pick apart the PGP encryption library. At the time of this writing, IDA cannot be downloaded from the creators, DataRescue. You can purchase IDA Pro from DataRescue, but you will need to perform an Internet search to find a download link for the free version of IDA. To the best of my knowledge, the last free version of IDA is 4.1, so entering ida + “4.1 ida pro” download datarescue should get you a list that contains at least one download link. Alternately, if you have downloaded the source code for this book, you will find the individual archive IDA_4_1 under “Chapter 1 Tools.”

Another tool for delving into the deepest layers of rootkit development is “Debugging Tools for Windows.” This package contains four debuggers, one of which is a kernel-level debugger that can come in handy when your device driver isn’t working as expected and debugging statements just aren’t enough to figure out what’s going on. This package includes the most recent DDKs, so you may already have it. If not, the package can be downloaded from www.microsoft.com/whdc/devtools/debugging/installx86.mspx. Kernel-level debugging isn’t covered in this book, but “Debugging Tools for Windows” is nonetheless a valuable addition to any rootkit toolkit.

You will find that the kernel debugger mentioned above is of little value without the symbols for the operating system you are using. You can get these symbols from www.microsoft.com/whdc/devtools/debugging/symbolpkg.mspx. After downloading and installing the symbols, you need to tell your
kernel debugger where they are. From Start ➤ All Programs ➤ Debugging Tools for Windows ➤ WinDbg, select the menu option File ➤ Symbol File Path, and browse to the directory where symbols were installed. Selecting the Symbols directory will magically transform the kernel debugger into a fountain of information that can be used to both fix rootkits and investigate new rootkit technologies.

There is one additional development tool mentioned in this book that has yet to be covered: the Visual C# compiler used to create the rootkit controller developed in Chapter 11. This is another free development environment offered by Microsoft, and can be found at http://msdn.microsoft.com/vstudio/express/visualcsharp. The Visual Studio C# 2005 development environment will not interfere with the Visual Studio C++ 2005 development environment, so feel free to download and install both. The C# compiler also makes a good addition to any rootkit toolkit.

Once you have the Microsoft DDK, a Windows C/C++ compiler, the Microsoft Windows Platform SDK, Sysinternals’ DebugView, RegMon, FileMon, and DiskMon, DataRescue’s IDA, Debugging Tools for Windows, Kernel Debugging Symbols, and Visual Studio C# 2005 Express, you will be ready to tackle basic rootkit development. Remember that the toolkit you develop can be a valuable collection for years to come, so take a moment to zip and archive the components you’ve collected before jumping into rootkit development. Figure 1-1 shows a typical rootkit toolkit.

![Figure 1-1: Rootkit Toolkit](image-url)
Chapter 1: Tools

The Microsoft Driver Development Kit

The DDK installation wizard is as straightforward as any Microsoft installation; just double-click setup.exe and answer a few questions. However, you can do a few things to make rootkit development much easier. The single most time-saving installation recommendation is to use the default installation directory; and if you absolutely must use another directory, keep the path simple, with no spaces or long directory names. This will be especially important if you are using the older XP DDK. The second recommendation is to select every possible download option. Skipping a few samples or skipping documentation to save a few megabytes of disk space will not make your life easier. You can, however, skip the debuggers offered with the 2003 SP1 DDK if you’ve already installed “Debugging Tools for Windows,” as these are the exact same debuggers.

Once you have installed the DDK, you can create two shortcuts to help with development. The shortcut examples that follow were developed for the Windows XP DDK, build number 2600; your target path may need to be altered depending upon your version of the DDK.

The first shortcut should use the following target:

```
%windir%\SYSTEM32\CMD.EXE /k C:\WINDDK\2600\bin\setenv.bat C:\WINDDK\2600 checked
```

For newer DDKs, use the following:

```
%windir%\SYSTEM32\CMD.EXE /k C:\NTDDK\bin\setenv.bat C:\NTDDK free
```

For the 2003 SP1 DDK, use

```
%windir%\SYSTEM32\CMD.EXE /k C:\WINDDK\3790.1830\bin\setenv.bat C:\WINDDK\3790.1830 free
```

and start in the %windir% directory.

This will be your “Checked DDK” icon.

The second shortcut should use this target:

```
%windir%\SYSTEM32\CMD.EXE /k C:\WINDDK\2600\bin\setenv.bat C:\WINDDK\2600 fre
```

For newer DDKs, use the following:

```
%windir%\SYSTEM32\CMD.EXE /k C:\NTDDK\bin\setenv.bat C:\NTDDK free
```

For the 2003 SP1 DDK, use

```
%windir%\SYSTEM32\CMD.EXE /k C:\WINDDK\3790.1830\bin\setenv.bat C:\WINDDK\3790.1830 free
```

and start in the same %windir% directory.
This will be your “Free DDK” icon.

The DDK uses the concept of Checked and Free driver development to differentiate between preliminary debug development and final release builds. The preceding shortcuts will set up the shells required for these two development environments. The projects covered in this book only use the Checked DDK shell, but you will eventually want to build a release version of one or all of the rootkits you will be creating.

**Microsoft Visual VC++ 2005 Express**

Unlike the DDK install, the Visual C++ 2005 Express installation might take a few minutes. For one thing, you have to use the Background Intelligent Transfer Service (BITS). If you get a message indicating that you must start this service, you will need to reconfigure the service to start automatically and then start it. This service might seem like a huge security hole just waiting to be exploited, but Microsoft has been pushing BITS for a long time, and there are currently no (publicly announced) known exploits taking advantage of this design, so you will need to go with the flow — at least until VC++ 2005 Express is fully installed.

As with the DDK, selecting the default installation path is recommended. Though an alternate path should not cause any problems, why ask for trouble? Unlike the DDK, there is no need to select all installation options. You may choose not to integrate the SQL Server. It isn’t required for any project covered in this book, but there is always the chance that you will one day develop an application that requires a database, so add it if you have the room.

The only strong recommendation that can be made is to include MSDN. Yes, it’s a large package that’s available over the Internet, but you can use Google to overcome most obstacles and you will not find a better integrated resource for developing software under VC++ 2005 Express, so do yourself a favor and check the MSDN box during installation.

**Microsoft Software Developers Kit**

The Microsoft Platform SDK installation (`PSDK-x86.exe`) is also wizard driven. Just answer the questions, agree to the license agreement, and keep clicking Next. The default for the most recent SDK install selects a custom installation. You should keep the custom installation selection, but only to add “Register Environment Variables” to the installation options. This should make compiling a little easier. Once installed, the SDK features described in the MSDN help files will be available to your programs.

**Sysinternals Freeware**

DebugView is freely available from Sysinternals at [www.sysinternals.com/Utilities/DebugView.html](http://www.sysinternals.com/Utilities/DebugView.html). Don’t let the price fool you; DebugView is an invaluable tool that will make rootkit development much easier. Download and create a shortcut for this utility before going too much farther.

You can also download Diskmon, Filemon, and Regmon from Sysinternals. These utilities can monitor disk, file, and registry activity, respectively. Debugging statements won’t always be able to tell you
what’s happening, but these utilities will. Adding them to your toolkit and creating shortcuts to them will make development that much easier. Eventually things will get complicated, as they always do; and when that happens, you’ll want all the help you can get.

**IDA**

As mentioned earlier, a Google search of `ida + “4.1 ida pro” download datarescue` should provide a list of web pages from which IDA can be downloaded. Once downloaded and installed, IDA can be used to look into Windows applications, libraries, and even device drivers. This tool provides an incredible wealth of information to the experienced user. Unfortunately, becoming an experienced user can be a daunting task. This book will walk you through basic IDA use, but if you intend to write rootkits, you will need to learn much more about IDA. If you already know that reverse engineering will be an important part of your future plans, I would recommend using this book as a primer and then replace the word “download” with the word “tutorial” in the Google search mentioned earlier.

**Debugging Tools for Windows**

The four debuggers provided in this package are exceptional tools. In particular, the Kernel Debugger can be invaluable for both fixing rootkits and investigating new rootkit technologies. Many of the difficulties encountered using undocumented Windows internals can also be overcome with this debugger. Unfortunately, this is also a complex utility that requires many hours to master. Fortunately, the Windows Debugger has a complete help system that can walk you through every step. If you are new to kernel debugging, I suggest you start with menu option Help ➪ Contents, and just keep reading.

**Verification**

To verify your Microsoft DDK installation, open a Checked shell (if you were following along you have an icon named “Checked DDK” on your desktop) and build one of the samples selected during the DDK installation. To build a sample, you need to traverse into a sample directory (any directory under the installation directory containing a “sources” file) and enter the command `build`. If you have installed properly, entering a build command from either the “Checked DDK” or “Free DDK” shell will initiate driver compilation and linking based on the “sources” file contained in that directory. Following the build, you can double-check your installation by searching for the newly created driver (*.sys) file in a directory beneath your build directory.

To verify your Microsoft VC++ 2005 Express installation, double-click the Microsoft VC++ 2005 Express icon. From the main menu, select File ➪ New ➪ Project. In the Project Types view, select Win32. From the Templates view, select Win32 Console Application. Enter the project name `myProject` and the solution name `MySolution`, and then press OK and Finish. Add the line `printf("Hello World!\n");` just before the return in `tmain`. You can now build the solution from the main menu by selecting Build ➪ Build Solution. If all is well, you should be able to open a command prompt, navigate to the solution directory defined during creation, and from the Debug directory, execute `myProject.exe`. If Microsoft VC++ 2005 Express was installed correctly, you should see Hello World! at the command prompt.
Chapter 1: Tools

To verify IDA, double-click idaw.exe (or the shortcut you’ve already created) and click OK at the opening screen. Then use Windows Explorer to navigate to your WINDOWS\System32 directory. From the System32 directory, drag and drop any dynamic link library (*.dll) onto the IDA file selection dialog. Then press OK twice (you may also have to press OK a third time to truncate data from a large segment) to load and analyze the library. Once loaded, IDA should provide an assembly code listing of the contents of the file beginning with the public start entry point.

To verify Debugging Tools for Windows, click WinDbg from Start ➪ All Programs ➪ Debugging Tools for Windows. From the Windows Debugger, select the menu option File ➪ Symbol File Path. This path should have been set after downloading and installing the symbols for your specific operating system. Check the Reload check box and press OK. If you have a Windows XP or later operating system and you have never loaded symbols, this should bring up the Local Kernel Debugger. If you have previously loaded symbols, you might need to use menu option File ➪ Kernel Debug and click OK from the Local tab to bring up the Local Kernel Debugger. In either case, the Local Kernel Debugger window should show no errors after the lines “Loading Kernel Symbols” or “Loading User Symbols.” To verify kernel debug operation, enter !process 0 0 in the command box (after lkd>). You should see a detailed list of processes.

VCVARS32.BAT

After verifying the Visual C++ build environment, you will need to prepare for manual compiling and linking. Microsoft uses the convention VCVARS32.BAT as the filename used to prepare a Command Prompt window for manual compiling and linking. If you installed Microsoft VC++ 2005 Express to the default location, VCVARS32.BAT can be found in C:\Program Files\Microsoft Visual Studio 8\VC\bin. You need to copy this file to a convenient location and execute it before manually building the user-level programs found in this book. Alternately, you can create a shortcut to cmd.exe that executes VCVARS32.BAT from its default location. Once the setup file has been executed from a Command Prompt window, you will be able to manually compile and link source code from that window.

Other Tools to Consider

This might be a bit premature, but rootkit development also depends upon rootkit detection and prevention tools. Once you have a thorough understanding of these tools, you can design and develop rootkits that defeat these detection and prevention systems. Of course, the developers of these detection and prevention systems do not consider them to be tools at all; they are more likely considered to be “security applications,” but to the rootkit developer, they are simply tools. See Chapter 13, “Detecting Rootkits,” and Chapter 14, “Preventing Rootkits,” for these tools.

What to Keep Out

Rootkits are often installed as payloads. A payload is the content section of an exploit. Exploits are the intrusions that take advantage of software vulnerabilities in order to add unintended software (payloads) to target machines. There are many types of payloads, and many exploits that can be used to deliver these payloads. This is one application detail of a rootkit that can also be applied to spyware,
viruses, and other malicious program types. Separating rootkit development from exploit development will provide an object-oriented environment in which any payload can be attached to any exploit. The advantage to this approach can be seen by using MetaSploit software (www.metasploit.com). MetaSploit enables the user to first select an exploit and then select the payload to insert using that exploit. Keeping these functions separated can be difficult if rootkit development is folded in with exploit development. Because rootkit development and exploit development require some of the same tools, it is easy to mix these development environments and end up with a rootkit that can only be compiled and linked in an exploit development environment that has changed since the last rootkit build.

Summary

At this point, you should have (as a minimum) the following:

- A Microsoft Windows Driver Development Kit (XP, 2000, or 2003)
- A C/C++ compiler (VC++ 2005 Express)
- The Microsoft Platform Software Development Kit

Also recommended are the following:

- MSDN
- A Kernel Debug Output Utility (DebugView from Sysinternals)
- IDA
- Debugging Tools for Windows

Once you have downloaded, installed, and verified the tools discussed in this chapter, you will be able to compile and run the code presented in this book. If you wish to follow along without compiling source code, binaries for each chapter can be downloaded from the Wrox website at www.wrox.com and run on any Windows 2000, XP, or 2003 operating system.