Troubleshooting with the Windows Sysinternals Tools

Guidance from the tools' creator

Mark Russinovich | Aaron Margosis

FREE SAMPLE CHAPTER
SHARE WITH OTHERS
# Contents at a glance

## PART I  GETTING STARTED

| CHAPTER 1 | Getting started with the Sysinternals utilities | 3 |
| CHAPTER 2 | Windows core concepts | 15 |
| CHAPTER 3 | Process Explorer | 41 |
| CHAPTER 4 | Autoruns | 113 |

## PART II  USAGE GUIDE

| CHAPTER 5 | Process Monitor | 145 |
| CHAPTER 6 | ProcDump | 193 |
| CHAPTER 7 | PsTools | 219 |
| CHAPTER 8 | Process and diagnostic utilities | 259 |
| CHAPTER 9 | Security utilities | 301 |
| CHAPTER 10 | Active Directory utilities | 351 |
| CHAPTER 11 | Desktop utilities | 373 |
| CHAPTER 12 | File utilities | 389 |
| CHAPTER 13 | Disk utilities | 401 |
| CHAPTER 14 | Network and communication utilities | 423 |
| CHAPTER 15 | System information utilities | 437 |
| CHAPTER 16 | Miscellaneous utilities | 461 |

## PART III  TROUBLESHOOTING—“THE CASE OF THE UNEXPLAINED...”

| CHAPTER 17 | Error messages | 467 |
| CHAPTER 18 | Crashes | 495 |
| CHAPTER 19 | Hangs and sluggish performance | 509 |
| CHAPTER 20 | Malware | 545 |
| CHAPTER 21 | Understanding system behavior | 607 |
| CHAPTER 22 | Developer troubleshooting | 631 |
This page intentionally left blank
| Sessions, window stations, desktops, and window messages | 35 |
| Remote desktop services sessions | 36 |
| Window stations | 37 |
| Desktops | 37 |
| Window messages | 39 |

**Chapter 3 Process Explorer**

| Procexp overview | 41 |
| Measuring CPU consumption | 43 |
| Administrative rights | 44 |
| Main window | 45 |
| Process list | 45 |
| Customizing column selections | 55 |
| Saving displayed data | 69 |
| Toolbar reference | 69 |
| Identifying the process that owns a window | 71 |
| Status bar | 71 |
| DLLs and handles | 72 |
| Finding DLLs or handles | 73 |
| DLL view | 74 |
| Handle view | 79 |
| Process details | 83 |
| Image tab | 84 |
| Performance tab | 86 |
| Performance Graph tab | 87 |
| GPU Graph tab | 88 |
| Threads tab | 89 |
| TCP/IP tab | 89 |
| Security tab | 90 |
| Environment tab | 91 |
| Strings tab | 92 |
| Services tab | 93 |
| .NET tabs | 94 |
| Job tab | 95 |
| Thread details | 96 |
| Verifying image signatures | 99 |
PART II  USAGE GUIDE

Chapter 5  Process Monitor

Getting started with Procmon .............................................. 146
Events .............................................................................. 148
Understanding the column display defaults ......................... 149
Customizing the column display ......................................... 151
Event Properties dialog box .............................................. 153
Displaying profiling events ............................................... 158
Finding an event .................................................................. 159
Copying event data .......................................................... 160
Jumping to a registry or file location ................................. 160
Searching online .............................................................. 160
Filtering, highlighting, and bookmarking ............................... 161
Configuring filters ............................................................ 161
Configuring highlighting .................................................... 164
Bookmarking .................................................................... 165
Advanced output .................................................................. 165
Saving filters for later use ................................................ 166
Chapter 8  Process and diagnostic utilities  259

VMMap ........................................................ 259
Starting VMMap and choosing a process ................. 260
The VMMap window ........................................ 262
Memory types .............................................. 264
Memory information ....................................... 265
Timeline and snapshots .................................... 266
Viewing text within memory regions ....................... 268
Finding and copying text .................................. 269
Viewing allocations from instrumented processes ......... 269
Address space fragmentation ............................... 272
Saving and loading snapshot results ....................... 273
VMMap command-line options ............................. 274
Restoring VMMap defaults .................................. 274

DebugView ..................................................... 275
What is debug output? ..................................... 275
The DebugView display ..................................... 275
Capturing user-mode debug output ......................... 277
Capturing kernel-mode debug output ....................... 278
Searching, filtering, and highlighting output ............... 279
Saving, logging, and printing ........................................ 281
Remote monitoring .................................................. 283
LiveKd .................................................................. 285
LiveKd requirements .............................................. 286
Running LiveKd .................................................... 286
Kernel debugger target types ................................... 287
Output to debugger or dump file ............................... 288
Dump contents ..................................................... 289
Hyper-V guest debugging ......................................... 290
Symbols ................................................................ 291
LiveKd examples .................................................... 291
ListDLLs ............................................................... 293
Handle .................................................................. 296
Handle list and search ............................................ 297
Handle counts ....................................................... 299
Closing handles ..................................................... 300

Chapter 9 Security utilities ........................................ 301
SigCheck ............................................................... 302
Which files to scan ............................................... 305
Signature verification ............................................. 306
VirusTotal analysis ............................................... 308
Additional file information ..................................... 310
Output format ....................................................... 312
Miscellaneous ....................................................... 313
AccessChk ............................................................ 314
What are “effective permissions”? .............................. 314
Using AccessChk .................................................. 315
Object type .......................................................... 317
Searching for access rights .................................... 320
Output options ....................................................... 321
Sysmon ................................................................. 323
Events recorded by Sysmon ..................................... 323
Installing and configuring Sysmon ......................... 331
Extracting Sysmon event data ................................. 336
AccessEnum .................................................... 337
ShareEnum ..................................................... 339
ShellRunAs ..................................................... 340
Autologon ..................................................... 342
LogonSessions .................................................. 343
SDelete ........................................................ 346
   Using SDelete ............................................. 347
   How SDelete works ....................................... 348

Chapter 10 Active Directory utilities 351
AdExplorer ..................................................... 351
   Connecting to a domain .................................... 351
   The AdExplorer display .................................... 352
Objects ................................................... 354
Attributes ................................................. 355
Searching ................................................ 357
Snapshots ................................................ 358
AdExplorer configuration ................................. 360
AdInsight ...................................................... 360
   AdInsight data capture .................................... 361
   Display options .......................................... 364
   Finding information of interest ......................... 365
   Filtering results ......................................... 368
   Saving and exporting AdInsight data ..................... 369
   Command-line options .................................... 370
AdRestore ...................................................... 371

Chapter 11 Desktop utilities 373
BgInfo ......................................................... 373
   Configuring data to display ............................... 374
   Appearance options ........................................ 377
   Saving BgInfo configuration for later use ............... 379
   Other output options ..................................... 379
   Updating other desktops ................................... 381
Chapter 12 File utilities 389

- Strings ......................................................... 389
- Streams ........................................................ 391
- NTFS link utilities ................................................ 392
  Junction .................................................. 393
  FindLinks ................................................. 394
- Disk Usage (DU) ................................................ 395
- Post-reboot file operation utilities ................................. 398
  PendMoves ............................................... 398
  MoveFile .................................................. 399

Chapter 13 Disk utilities 401

- Disk2Vhd ........................................................ 401
- Sync 408
- DiskView ....................................................... 410
- Contig ........................................................ 413
  Defragmenting existing files ........................................ 414
  Analyzing fragmentation of existing files ............................ 415
  Analyzing free-space fragmentation .................................... 416
  Creating a contiguous file ............................................. 417
- DiskExt ........................................................ 418
- LDMDump ..................................................... 419
- VolumeID .................................................. 421
## Chapter 16 Miscellaneous utilities

<table>
<thead>
<tr>
<th>Utility</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegJump</td>
<td>461</td>
</tr>
<tr>
<td>Hex2Dec</td>
<td>462</td>
</tr>
<tr>
<td>RegDelNull</td>
<td>463</td>
</tr>
<tr>
<td>Bluescreen Screen Saver</td>
<td>463</td>
</tr>
<tr>
<td>Ctrl2Cap</td>
<td>464</td>
</tr>
</tbody>
</table>

## PART III TROUBLESHOOTING—”THE CASE OF THE UNEXPLAINED...”

## Chapter 17 Error messages

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Troubleshooting error messages</td>
<td>468</td>
</tr>
<tr>
<td>The Case of the Locked Folder</td>
<td>469</td>
</tr>
<tr>
<td>The Case of the File In Use Error</td>
<td>471</td>
</tr>
<tr>
<td>The Case of the Unknown Photo Viewer Error</td>
<td>472</td>
</tr>
<tr>
<td>The Case of the Failing ActiveX Registration</td>
<td>473</td>
</tr>
<tr>
<td>The Case of the Failed Play-To</td>
<td>476</td>
</tr>
<tr>
<td>The Case of the Installation Failure</td>
<td>477</td>
</tr>
<tr>
<td>The troubleshooting</td>
<td>477</td>
</tr>
<tr>
<td>The analysis</td>
<td>480</td>
</tr>
<tr>
<td>The Case of the Unreadable Text Files</td>
<td>482</td>
</tr>
<tr>
<td>The Case of the Missing Folder Association</td>
<td>483</td>
</tr>
<tr>
<td>The Case of the Temporary Registry Profiles</td>
<td>486</td>
</tr>
<tr>
<td>The Case of the Office RMS Error</td>
<td>491</td>
</tr>
<tr>
<td>The Case of the Failed Forest Functional Level Raise</td>
<td>492</td>
</tr>
</tbody>
</table>

## Chapter 18 Crashes

<table>
<thead>
<tr>
<th>Crash</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Troubleshooting crashes</td>
<td>495</td>
</tr>
<tr>
<td>The Case of the Failed AV Update</td>
<td>498</td>
</tr>
<tr>
<td>The Case of the Crashing Proksi Utility</td>
<td>500</td>
</tr>
<tr>
<td>The Case of the Failed Network Location Awareness Service</td>
<td>501</td>
</tr>
<tr>
<td>The Case of the Failed EMET Upgrade</td>
<td>502</td>
</tr>
</tbody>
</table>
The Case of the Missing Crash Dump ...........................................504
The Case of the Random Sluggishness ...........................................505

Chapter 19 Hangs and sluggish performance 509
Troubleshooting hangs and sluggish performance ..............................510
The Case of the IExplore-Pegged CPU ...........................................511
The Case of the Runaway Website .................................................514
The Case of the Excessive ReadyBoost ..........................................517
The Case of the Stuttering Laptop Blu-ray Player .................................518
The Case of the Company 15-Minute Logons ...................................522
The Case of the Hanging PayPal Emails .......................................523
The Case of the Hanging Accounting Software .................................526
The Case of the Slow Keynote Demo .............................................528
The Case of the Slow Project File Opens .......................................533
The Compound Case of the Outlook Hangs ....................................538

Chapter 20 Malware 545
Troubleshooting malware ..........................................................546
Stuxnet ......................................................................................549
Malware and the Sysinternals utilities ...........................................549
The Stuxnet infection vector .........................................................550
Stuxnet on Windows XP .............................................................550
Looking deeper ...........................................................................555
Filtering to find relevant events .....................................................555
Stuxnet system modifications .......................................................558
The .PNF files. ............................................................................563
Windows 7 elevation of privilege .....................................................566
Stuxnet revealed by the Sysinternals utilities ................................569
The Case of the Strange Reboots .....................................................569
The Case of the Fake Java Updater ..................................................574
The Case of the Winwebsec Scareware ...........................................577
The Case of the Runaway GPU .....................................................587
The Case of the Unexplained FTP Connections ...............................588
The Case of the Misconfigured Service ........................................... 592
The Case of the Sysinternals-Blocking Malware .......................... 596
The Case of the Process-Killing Malware ................................. 598
The Case of the Fake System Component ................................. 600
The Case of the Mysterious ASEP .............................................. 602

Chapter 21 Understanding system behavior ............................... 607
  The Case of the Q: Drive ................................................. 607
  The Case of the Unexplained Network Connections .................. 611
  The Case of the Short-Lived Processes ................................ 612
  The Case of the App Install Recorder .................................. 617
  The Case of the Unknown NTLM Communications .................. 625

Chapter 22 Developer troubleshooting .................................... 631
  The Case of the Broken Kerberos Delegation .......................... 631
  The Case of the ProcDump Memory Leak ............................... 632

Index .................................................................... 637
Foreword

The arrival of a new edition of Troubleshooting with the Windows Sysinternals Tools is always a treat, and when mine arrived at my country estate in Scotland, I prepared myself for a ride as exciting as my first time flying. Now, I understand that, to non-magical people (we call them Sysintuggles), it appears, against all comprehension, that the authors were trying to solve the problem of “why don’t people read instruction manuals more often?” and stumbled across the baffling conclusion of “because those pamphlets are simply too small.” (And they have overachieved on solving that problem, producing a volume large enough to defend against even the most vicious lycanthrope.) But they simply don’t understand the magic that this work unlocks.

I settled in to have a read. Upon stroking the spine of this book, it opened placidly and I began to flip through it. This is a spell book of the highest quality, designed with practical magic in mind. Paired with the theory in Windows Internals, you’ll be equipped with the finest magical education available today. Using the potions and incantations included herein, it’s possible to do truly remarkable things. It can teach you to bewitch Windows and ensnare malware. It can tell you how to bottle insight, brew troubleshooting glory, and even put a stopper in bluescreens. I started annotating my book, dog-earing it, and writing related spells in the margins, and soon I had an indispensable resource. It has an honored spot on my bookshelf.

This is a powerful resource for doing truly advanced magic. If you are responsible for system administration anywhere, large or small, you have something to learn from this book. Professor Russinovich truly is the brightest wizard of his age, and he and his house-elf have created an indispensable work.

A Noted Person
May 2016
This page intentionally left blank
Introduction

The Sysinternals Suite is a set of over 70 advanced diagnostic and troubleshooting utilities for the Microsoft Windows platform written by me—Mark Russinovich—and Bryce Cogswell. Since Microsoft’s acquisition of Sysinternals in 2006, these utilities have been available for free download from Microsoft’s Windows Sysinternals website (part of Microsoft TechNet).

The goal of this book is to familiarize you with the Sysinternals utilities and help you understand how to use them to their fullest. The book will also show you examples of how I and other Sysinternals users have leveraged the utilities to solve real problems on Windows systems.

Although I coauthored this book with Aaron Margosis, the book is written as if I am speaking. This is not at all a comment on Aaron’s contribution to the book; without his hard work, this book would not exist.

Note See the “Late-breaking changes” section later in this chapter for updates that occurred as we were going to publish.

Tools the book covers

This book describes all of the Sysinternals utilities that are available on the Windows Sysinternals website (http://technet.microsoft.com/en-us/sysinternals/default.aspx) and all of their features as of the time of this writing (early summer, 2016). However, Sysinternals is highly dynamic: existing utilities regularly gain new capabilities, and new utilities are introduced from time to time. (To keep up, follow the RSS feed of the “Sysinternals Site Discussion” blog: http://blogs.technet.microsoft.com/sysinternals/) So, by the time you read this book, some parts of it might already be out of date. That said, you should always keep the Sysinternals utilities updated to take advantage of new features and bug fixes.

This book does not cover Sysinternals utilities that have been deprecated and are no longer available on the Sysinternals site. If you are still using RegMon (Registry Monitor) or FileMon (File Monitor), you should replace them with Process Monitor, described in Chapter 5. Rootkit Revealer, one of the computer industry’s first rootkit detectors (and the tool that discovered the “Sony rootkit”), has served its purpose and has been
retired. Similarly, a few other utilities (such as Newsid and EfsDump) that used to provide unique value have been retired because either they were no longer needed or equivalent functionality was eventually added to Windows.

The history of Sysinternals

The first Sysinternals utility I wrote, Ctrl2cap, was born of necessity. Before I started using Windows NT in 1995, I mostly used UNIX systems, which have keyboards that place the Ctrl key where the Caps Lock key is on standard PC keyboards. Rather than adapt to the new layout, I set out to learn about Windows NT device driver development and to write a driver that converts Caps Lock key presses into Ctrl key presses as they make their way from the keyboard into the Windows NT input system. Ctrl2cap is still posted on the Sysinternals site today, and I still use it on all my systems.

Ctrl2cap was the first of many tools I wrote to learn about the way Windows NT works under the hood while at the same time providing some useful functionality. The next tool I wrote, NTFSDOS, I developed with Bryce Cogswell. I had met Bryce in graduate school at Carnegie Mellon University, and we had written several academic papers together and worked on a startup project where we developed software for Windows 3.1. I pitched the idea of a tool that would allow users to retrieve data from an NTFS-formatted partition by using the ubiquitous DOS floppy. Bryce thought it would be a fun programming challenge, and we divided up the work and released the first version about a month later.

I also wrote the next two tools, Filemon and Regmon, with Bryce. These three utilities—NTFSDOS, Filemon, and Regmon—became the foundation for Sysinternals. Filemon and Regmon, both of which we released for Windows 95 and Windows NT, showed file system and registry activity, becoming the first tools anywhere to do so and making them indispensable troubleshooting aids.

Bryce and I decided to make the tools available for others to use, but we didn’t have a website of our own, so we initially published them on the site of a friend, Andrew Schulman, who I’d met in conjunction with his own work uncovering the internal operation of DOS and Windows 95. Going through an intermediary didn’t allow us to update the tools with enhancements and bug fixes as quickly as we wanted, so in September 1996 Bryce and I created NTInternals.com to host the tools and articles we wrote about the internal operation of Windows 95 and Windows NT. Bryce and I had also developed tools that we decided we could sell for some side income, so in the same month, we also founded Winternals Software, a commercial software company that we bootstrapped by driving traffic with a single banner ad on NTInternals.com. The first
utility we released as Winternals Software was NTRecover, a utility that enabled users to mount the disks of unbootable Windows NT systems from a working system and access them as if they were locally attached disks.

The mission of NTInternals.com was to distribute freeware tools that leveraged our deep understanding of the Windows operating system in order to deliver powerful diagnostic, monitoring, and management capabilities. Within a few months, the site, shown in the following screenshot as it looked in December 1996 (thanks to the Internet Archive's Wayback Machine), drew 1,500 visitors per day, making it one of the most popular utility sites for Windows in the early days of the internet revolution. In 1998, at the “encouragement” of Microsoft lawyers, we changed the site’s name to Sysinternals.com.

Over the next several years, the utilities continued to evolve. We added more utilities as we needed them, as our early power users suggested enhancements, or when we thought of a new way to show information about Windows.

The Sysinternals utilities fell into three basic categories: those used to help programmers, those for system troubleshooting, and those for systems management. DebugView, a utility that captures and displays program debug statements, was one of the early developer-oriented tools that I wrote to aid my own development of
device drivers. DLLView, a tool for displaying the DLLs that processes have loaded, and
HandleEx, a process-listing GUI utility that showed open handles, were two of the early
troubleshooting tools. (I merged DLLView and HandleEx to create Process Explorer in
2001.) The PsTools, discussed in Chapter 7, are some of the most popular management
utilities, bundled into a suite for easy download. PsList, the first PsTool, was inspired
initially by the UNIX ps command, which provides a process listing. The utilities grew
in number and functionality, becoming a software suite of utilities that allowed you to
easily perform many tasks on a remote system without requiring installation of special
software on the remote system beforehand.

Also in 1996, I began writing for Windows IT Pro magazine, highlighting Windows
internals and the Sysinternals utilities and contributing additional feature articles, in-
cluding a controversial article in 1996 that established my name within Microsoft itself,
though not necessarily in a positive way. The article, “Inside the Difference Between
Windows NT Workstation and Windows NT Server,” pointed out the limited differences
between Windows NT Workstation and Windows NT Server, which contradicted Micro-
soft’s marketing message.

I exacerbated Microsoft’s negative view of me by releasing Ntcrash and Ntcrash2,
tools that are now called “fuzzers,” that barraged the Windows NT system call interface
with random garbage. The tools identified several dozen system calls that had weak
parameter validation that allowed memory corruption and blue-screen crashes by
unprivileged user-mode processes. (In the threat landscape of the 1990s, these were
simply considered reliability bugs and were embarrassing—today they’d be classified as
“important” security bugs.)

As the utilities continued to evolve and grow, I began to contemplate writing a book
on Windows internals. Such a book already existed, Inside Windows NT (Microsoft Press,
1992), the first edition of which was written by Helen Custer alongside the original re-
lease of Windows NT 3.1. The second edition was rewritten and enhanced for Windows
NT 4.0 by David Solomon, a well-established operating system expert, trainer, and
writer who had worked at DEC. Instead of writing a book from scratch, I contacted him
and suggested that I coauthor the third edition, which would cover Windows 2000. My
relationship with Microsoft had been on the mend since the 1996 article as the result of
my sending Windows bug reports directly to Windows developers, but David still had
to obtain permission, which Microsoft granted.

As a result, David Solomon and I coauthored the third, fourth, fifth, and sixth
editions of the book, which we renamed Windows Internals at the fourth edition. We
brought in Alex Ionescu as a co-author beginning with the fifth edition. By the sixth
edition, the content had grown so much that we needed to split the book into two
parts. Not long after we finished Inside Windows 2000 (Microsoft Press, 2000), I joined
David to teach his Windows internals seminars, adding my own content. Offered around the world, even at Microsoft to the developers of Windows, these classes have long used the Sysinternals utilities to show students how to peer deep into Windows internals and learn more when they returned to their developer and IT professional roles at home.

By 2006, my relationship with Microsoft had been strong for several years, Winternals had a full line of enterprise management software and had grown to about 100 employees, and Sysinternals had two million downloads per month. On July 18, 2006, Microsoft acquired Winternals and Sysinternals. Not long after, Bryce and I (there we are below in 2006) moved to Redmond to become a part of the Windows team. Today, I serve as the Chief Technology Officer of Microsoft Azure, leading the technical strategy and architecture of the Azure cloud computing platform.

Two goals of the acquisition were to make sure that the tools Bryce and I developed would continue to be freely available and that the community we built would thrive, and they have. Today, the Windows Sysinternals site on technet.microsoft.com is one of the most frequently visited sites on TechNet, averaging 4.5 million downloads per month. Sysinternals power users come back time and again for the latest versions of the utilities and for new utilities, such as the recently released Sysmon and PsPing, as well as to participate in the Sysinternals community, a growing forum with over 42,000 registered users at the time of this writing. I remain dedicated to continuing to enhance the existing tools and to add new tools.

Many people suggested that a book on the tools would be valuable, but it wasn’t until David Solomon suggested that one was way overdue that I started the project. My responsibilities at Microsoft did not permit me to devote the time necessary to write another book, but David pointed out that I could find someone to help. I was pleased that Aaron Margosis agreed to partner with me. Aaron is a Principal Consultant with Microsoft Cybersecurity Services who is known for his deep understanding of Windows security and application compatibility. I have known Aaron for many years, and his
excellent writing skills, familiarity with Windows internals, and proficiency with the Sysinternals tools made him an ideal coauthor.

**Who should read this book**

This book exists for Windows IT professionals, power users, and even developers who want to make the most of the Sysinternals tools. Regardless of your experience with the tools, and whether you manage the systems of a large enterprise, a small business, or the PCs of your family and friends, you’re sure to discover new tools, pick up tips, and learn techniques that will help you more effectively troubleshoot the toughest Windows problems and simplify your system-management operations and monitoring.

**Assumptions**

This book expects that you have familiarity with the Windows operating system. Basic familiarity with concepts such as processes, threads, virtual memory, and the Windows command prompt is helpful, though some of these concepts are discussed in Chapter 2, “Windows core concepts.”

**Organization of this book**

The book is divided into three parts. Part I, “Getting started,” provides an overview of the Sysinternals utilities and the Sysinternals website, describes features common to all of the utilities, tells you where to go for help, and discusses some Windows core concepts that will help you better understand the platform and the information reported by the utilities.

Part II, “Usage guide,” is a detailed reference guide covering all of the Sysinternals utilities’ features, command-line options, system requirements, and caveats. With plentiful screenshots and usage examples, this section should answer just about any question you have about the utilities. Major utilities such as Process Explorer and Process Monitor each get their own chapter; subsequent chapters cover utilities by category, such as security utilities, Active Directory utilities, and file utilities.

Part III, “Troubleshooting—The Case of the Unexplained...,” contains stories of real-world problem solving using the Sysinternals utilities from Aaron and me, as well as from administrators and power users from around the world.
Conventions and features in this book

This book presents information using conventions designed to make the information readable and easy to follow:

- Boxed elements with labels such as “Note” provide additional information or alternative methods for completing a step successfully.
- Text that you type (apart from code blocks) appears in bold.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, “Press Alt+Tab” means that you hold down the Alt key while you press the Tab key.
- A vertical bar between two or more menu items (for example, File | Close), means that you should select the first menu or menu item, and then the next, and so on.
- In command-line syntax specifications, a vertical bar means “OR,” square braces mean “optional,” italicized text is a placeholder for information that you provide, curly braces represent groupings, and ellipses represent a repeating pattern. Consider this example:

```
procdump
[-ma | -mp | -d callback_DLL] [-64] [-r [1..5] [-a]] [-o]
[-n count] [-s secs]
[-c|-cl percent [-u]] [-m|-m1 commit] [-p|-pl counter_threshold]
[-e [1 [-g] [-b]]] [-h] [-l] [-t] [-f filter,...]
{[{-w} process_name]|service_name|PID } [dump_file | dump_folder] } |
{-x dump_folder image_file [arguments]}
```

This indicates that you can optionally use `-ma`, `-mp`, or `-d`; if you use `-d`, you must supply a value for `callback_DLL`. You can also choose to use the `-f` option; if you do, you must supply one or more `filter` values. The groupings in the last four lines show that you must specify a `process_name`, `service_name`, or `PID`, or use the `-x` option with a `dump_folder` and `image_file`. 
System requirements

The Sysinternals tools work on the following supported versions of Windows, including 64-bit editions, unless otherwise specified:

- Windows Vista
- Windows 7
- Windows 8.1
- Windows 10 (desktop)\(^1\)
- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2
- Windows Server 2016, including Nano Server

Some tools require administrative rights to run, and others implement specific features that require administrative rights.

Late-breaking changes

Just as we were finishing work on this book, I released updated versions of many of the utilities to support the Nano Server edition of Windows Server 2016. Nano Server is a small-footprint, headless installation option for Windows Server 2016 that includes a minimal number of features and services. Of particular interest to Sysinternals users is that Nano Server does not include a 32-bit subsystem nor GUI components. As described in Chapter 1, “Getting started with the Sysinternals utilities,” each Sysinternals utility has always been packaged as a single 32-bit executable, with any additional required files, such as 64-bit binaries, embedded as resources that can be extracted and executed as needed. Of course, none of these 32-bit images would work on Nano Server, so I created native 64-bit versions of the console-mode utilities, appending “64.exe” to their file names. For example, the 64-bit version of SigCheck.exe is

---

\(^1\) The Sysinternals utilities are all Win32 apps, support only x86 and x64 architectures and are not compatible with Windows 10 Mobile, IoT, Xbox, etc.
SigCheck64.exe. In addition, I created a console-mode version of the LoadOrd (Load Order) utility, LoadOrdC.exe, and a native 64-bit version, LoadOrdC64.exe.

Nano Server management relies heavily on PowerShell Remoting. PowerShell treats any output to the standard error (stderr) stream as indicative of an error. The console-mode Sysinternals utilities had always written banner and syntax information to stderr. To improve the utilities’ support for PowerShell and for Nano Server in particular, the utilities now write banner and syntax information to the standard output (stdout) stream, and use the new `–nobanner` command-line option to omit banner output. Note that this replaces the `–q` option that many of the utilities had used for the same purpose.

Acknowledgments

First, Aaron and I would like to thank Bryce Cogswell, cofounder of Sysinternals, for his enormous contribution to the Sysinternals tools. Because of our great collaboration, what Bryce and I published on Sysinternals was more than just the sum of our individual efforts. Bryce retired from Microsoft in October 2010, and we wish him luck in whatever he pursues.

We’d like to thank David Solomon for spurring Mark to write this book, providing detailed review of many chapters, and writing the Foreword for the first edition. Dave has also been one of Sysinternals most effective evangelists over the years and has suggested many valuable features.

Thanks to Luke Kim, who has been invaluable in helping upgrade the projects to the latest versions of Microsoft Visual Studio, moving the tools into Visual Studio Team Services (VSTS) source control, streamlining the build and publishing process, and managing the Sysinternals.com website and live.sysinternals.com infrastructure servers (which are running on Azure). Thanks also to Kent Sharkey for publishing updates to Sysinternals.com.

Up until a few years ago, Bryce and I were the sole authors of the tools, but I started accepting contributions from other developers. Ken Johnson, Andrew Richards, Thomas Garnier, David Magnotti, Dmitry Davydok, Daniel Pearson, Justin Jiang and the rest of the Nano Server team, Giulia Biagini, Pavel Yosifovich, and Aaron Margosis have all added significant features to specific tools.

Huge thanks to John Sheehan for his help describing previously-undocumented details about how AppContainers work; to Alex Ionescu for material relating to
protected processes; and to Ned Pyle, Marty Lichtel, and Carl Harrison for allowing us to incorporate cases they had previously published.

We are grateful to the following people who provided valuable and insightful technical review, corrections, and suggestions for this edition of the book: Andrew Richards, Bhaskar Rastogi, Bruno Aleixo, Burt Harris, Chris Jackson, Crispin Cowan, Greg Cottingham, Ken Johnson (a.k.a., Skywing), Luke Kim, Mario Raccagni, Steve Thomas, and Yong Rhee.

Aaron and I considered it a longshot when we asked Noted Person to consider writing the Foreword for this edition, and we are still giddy and starstruck that Noted Person agreed. Our unbounded thanks to N.P.²

We’d like to thank Devon Musgrave (acquisitions editor and developmental editor) and Carol Dillingham (project editor) from Microsoft Press for all the great work they have done for us on this edition, and especially for their infinite patience as we slipped our deadline from a fixed date to something closer to “infinity.” Thanks to Steve Sagman from Waypoint Press for project management and desktop publishing. Thanks also to Christophe Nasarre for technical editing and Roger LeBlanc for copyediting.

Aaron thanks his wife, Elise, and their children—Elana, Jonah, and Gabriel—for their love and support. Aaron also thanks Brenda Schrier for his author photo. Aaron also thanks the Washington Nationals Baseball Club and West Ham United F.C.

Mark thanks his wife, Daryl, and daughter, Maria, for supporting all his endeavors.

Errata, updates, and book support

We’ve made every effort to ensure the accuracy of this book. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

http://aka.ms/TroubleshootSysint/errata

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, e-mail Microsoft Press Book Support at mspinput@microsoft.com.

² Noted Person’s secret identity is Chris Jackson, a.k.a., The App Compat Guy, a.k.a., Captain Inappropriate.
Please note that product support for Microsoft software is not offered through the previous addresses. For help with Microsoft software or hardware, go to http://support.microsoft.com.

**Free ebooks from Microsoft Press**

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

http://aka.ms/mspressfree

Check back often to see what is new!

**We want to hear from you**

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

**Stay in touch**

Let’s keep the conversation going. Follow Microsoft Press on Twitter: http://twitter.com/MicrosoftPress.
This page intentionally left blank
PART I

Getting started

CHAPTER 1  Getting started with the Sysinternals utilities ....... 3
CHAPTER 2  Windows core concepts ............................. 15
CHAPTER 3  Process Explorer ................................. 41
CHAPTER 4  Autoruns  ..................................... 113
This page intentionally left blank
A question I often hear is, “Why is all this stuff running on my computer?” That’s often followed with, “How do I get rid of it?” The Microsoft Windows operating system is a highly extensible platform. Not only can programmers write applications that users can choose to run, those programmers can “add value” by having their software run automatically without troubling the user to start it, by adding visible or nonvisible features to Windows Explorer and Internet Explorer, or by supplying device drivers that can interact with custom hardware or change the way existing hardware works. Sometimes the “value” to the user is doubtful at best; sometimes the value is for someone else entirely and the software acts to the detriment of the user (which is when the software is called malware).

Autostarts is the term I use to refer to software that runs automatically without being intentionally started by a user. This type of software includes drivers and services that start when the computer is booted; applications, utilities, and shell extensions that start when a user logs on; and browser extensions that load when Internet Explorer is started. Over 200 locations in the file system and registry allow autostarts to be configured on x64 versions of Windows. These locations are often referred to as Autostart Extensibility Points, or ASEPs.

ASEPs have legitimate and valuable purposes. For example, if you want your instant messaging contacts to know when you are online, having the messaging client start when you log on is a great help. Users enjoy search toolbars and PDF readers that become part of Internet Explorer. And much of Windows itself is implemented through ASEPs in the form of drivers, services, and Explorer extensions.

On the other hand, consider the plethora of “free” trial versions of programs that computer manufacturers install on new computers and that fill up the taskbar notification area. Consider also the semihidden processes that legitimate vendors run all the time so that their applications can appear to start more quickly. Do you really need all these processes constantly consuming resources? On top of that, malware almost always hooks one or more ASEPs, and virtually every ASEP in Windows has been used by malware at one point or another.

Although Windows Vista and Windows 7 offer the System Configuration Utility (msconfig.exe, shown in Figure 4-1) to let you see some of these autostarts, it shows only a small subset and is of limited use. Msconfig also requires administrative rights, even just to view settings. That means it cannot identify or disable per-user autostarts belonging to nonadministrator users.
Some of MSConfig’s functionality moved into Task Manager when Windows 8 introduced a Startup tab, as shown in Figure 4-2. Although it no longer requires administrative rights, it no longer shows a process’ full command line, nor where the ASEP is configured.
Bryce and I created the Autoruns utility to expose as many autostarts as we could identify, and to make it easy to disable or remove those autostarts. The information that Autoruns exposes can be discovered manually if you know where to look in the registry and file system. Autoruns automates that task, scanning a large number of ASEPs in a few seconds, verifying entries, and making it easier to identify entries with suspicious characteristics, such as the lack of a digital signature, or that are flagged as suspicious by VirusTotal. We also created a command-line version, AutorunsC, to make it possible to capture the same information in a scripted fashion.

Using either Autoruns or AutorunsC, you can easily capture a baseline of the ASEPs on a system. That baseline can be compared against results captured at a later time so that changes can be identified for troubleshooting purposes. Many organizations use Autoruns as part of a robust change-management system, capturing a new baseline whenever the desktop image is updated.

**Autoruns fundamentals**

Launch Autoruns and it immediately begins filling its display with entries collected from known ASEPs. As shown in Figure 4-3, each shaded row represents an ASEP location, with a Regedit icon if it is a registry location or a folder icon if it is stored in the file system. The rows underneath a shaded row indicate entries configured in that ASEP. Each row includes the name of the autostart entry; the description, publisher and timestamp of the item; and the path to the file to run and an icon for that file. Each row also has a check box to temporarily disable the entry, and a column to display VirusTotal results. A panel at the bottom of the window displays details about the selected entry, including its full command line. The Everything tab, which is displayed when Autoruns starts, displays all ASEP entries on the system; you use the 19 other tabs to view just specific categories of autostarts. Each of these categories will be described later in this chapter.

The Image Path column shows the full path to the target file identified by the autostart entry. In some cases, this will be the first name in the autostart’s command line. For autostarts that use a hosting process—such as Cmd.exe, Wscript.exe, Rundll32.exe, Regsvr32.exe, or Svchost.exe—the image path identifies the target script or DLL on the command line instead of the main executable. For entries that involve levels of indirection, Autoruns follows the indirection to identify the target image. For example, the Internet Explorer “Browser Helper Objects” ASEPs are recorded as GUIDs in the registry; Autoruns identifies the corresponding InProcServer entries under HKCR\\CLSID and reports those DLLs. If the target file cannot be found in the expected location, the Image Path column will include the text “File not found” and the entry will be highlighted in yellow.

---

1 Scheduled Tasks appear with a folder icon, because configuration settings for tasks were stored in %windir%\Tasks prior to Windows Vista. As part of the re-architecting of Task Scheduler, configuration settings are now in the registry under HKLM\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Schedule\\TaskCache and in the file system under System32\\Tasks.
If the file identified in the image path is a Portable Executable (PE) file, the Timestamp column reports the date and time in the local time zone in which the image was created by the linker; otherwise—for example, for script files—the timestamp reports the last-write time of the file according to the file system. For the shaded rows that identify an ASEP location, the timestamp reports the last-modification time for the registry key or file system directory.

The Description and Publisher columns in the display are taken from the Description and Company Name version resources, respectively, for files that contain version resources, such as EXE and DLL files. If the file's digital signature has been verified, the Publisher column displays the subject name from the corresponding code-signing certificate. (See the “Verifying code signatures” section later in this chapter for more information.)

The Description and Publisher columns are left blank if the target file cannot be found, has no Description and Company Name in its version resources, or has no version resource (which is always true of script files). The VirusTotal column is blank until you request information from the VirusTotal service, as described in the “VirusTotal analysis” section later in this chapter.

Autoruns calls attention to suspicious images by highlighting their entries in pink. Autoruns considers an image file suspicious if it has no description or publisher, or if signature verification is enabled and the image doesn’t have a valid signature.

You can quickly search for an item by pressing Ctrl+F and entering text to search for. Autoruns will select the next row that contains the search text. Pressing F3 repeats the search from the current location. Pressing Ctrl+C copies the text of the selected row to the clipboard as tab-delimited text.
On the Options menu, the Scan Options entry is disabled while Autoruns is scanning the system. To cancel the scan so that you can change those options (which are described later in this chapter), press the Esc key. A change to any selection in Scan Options takes effect during the next scan. To run a new scan with the same options, press F5 or click the Refresh button on the toolbar.

Disabling or deleting autostart entries
With Autoruns, you can disable or delete autostart entries. Deleting an entry permanently removes it, and you should do this only if you’re certain you never want the software to autostart again. Select the entry in the list, and press the Del key. Because there is no Undo, Autoruns prompts for confirmation before deleting the autostart entry.

By contrast, when you disable an entry by clearing its check box, Autoruns leaves a marker behind that Autoruns recognizes and with which it can reconstitute and re-enable the entry. For example, for most registry ASEPs, Autoruns creates an AutorunsDisabled subkey in the ASEP location and copies the registry value being disabled into that subkey before deleting the original value. Windows will not process anything in that subkey, so the items in it will not run, but Autoruns displays them as disabled autostarts. Checking the entry again puts the entry back into the actual ASEP location. For ASEPs in the file system such as in the Start menu, Autoruns creates a hidden directory named AutorunsDisabled and moves disabled entries into that directory.

Note that disabling or deleting an autostart entry prevents it from being automatically started in the future. It does not stop any existing processes, nor does it delete or uninstall the ASEP’s target file.

Also note that if you disable autostarts that are critical for system boot, initialization, or correct operation, you can put the system into a state in which recovery is not possible without booting into an alternate operating system or recovery environment.

Autoruns and administrative permissions
The vast majority of ASEPs are in locations that grant Read permission to standard users. On some versions of Windows, the registry keys containing configuration information for some services are locked down, and many scheduled tasks are not standard-user readable. But for the most part, Autoruns works perfectly fine without administrative rights for the purposes of viewing autostart entries.

Administrative rights are required to view all autostarts, and they are required if you need to change the state of entries in systemwide locations, such as HKLM or the all users’ Startup directory in the Start menu. If you select or clear a check box, or try to delete one of these entries without administrative rights, Autoruns will report Access Denied. The error message dialog box includes a Run As Administrator button that lets you restart Autoruns elevated. (See Figure 4-4.) When Autoruns has administrative rights, configuration changes should succeed. You can also restart Autoruns with User Account Control (UAC) elevation by choosing Run As Administrator from the File menu.
To ensure that Autoruns has elevated rights when it launches, start Autoruns with the \texttt{-e} command-line option. This will request UAC elevation if the invoker is not already running elevated. See the “Administrative rights” section in Chapter 2, “Windows core concepts,” for more information on UAC elevation.

**Verifying code signatures**

Anyone can create a program and stick the name “Microsoft Corporation” in it. Therefore, seeing that text in the Publisher column gives only a low degree of assurance that the file in question was created by Microsoft and has not been modified since. Verifying a digital signature associated with that file gives a much higher degree of assurance of the file’s authenticity and integrity. The file format for some types of files allows for a digital signature to be embedded within the file. Files can also be \texttt{catalog-signed}, meaning that the information needed to validate a file’s content is in a separate file. Catalog signing means that even plain text files can be verified.

You can verify an entry’s digital signature by selecting the entry and choosing Verify Image from the Entry menu. If the file has been signed with a valid code-signing certificate that derived from a root certificate authority that is trusted on the computer, the text in the Publisher column changes to “(Verified)” followed by the subject name in the code-signing certificate. If the file has not been signed or the verification fails for any other reason, the text changes to “(Not verified)” followed by the company name from the file’s version resource, if present.

Instead of verifying entries one at a time, you can enable Verify Code Signatures in the Scan Options dialog box and rescan. Autoruns will then attempt to verify the signatures for all image paths as it scans autostarts. Note that the scan might take longer because it also verifies whether each signing certificate has been revoked by its issuer, which requires Internet connectivity to work reliably.

Files for which signature checks fail might be considered suspicious and therefore appear in pink. A common malware technique is to install files that on casual inspection appear to be legitimate Windows files but are not signed by Microsoft.

The Sysinternals SigCheck utility, described in Chapter 9, “Security utilities,” provides deeper detail for file signatures, including whether the file is catalog-signed and the location of the catalog.
VirusTotal analysis

VirusTotal.com is a free web service that lets users upload files to be analyzed by over 50 antivirus engines and see the results of those scans. Most users interact with VirusTotal by opening a web browser to https://www.virustotal.com and uploading one file at a time. VirusTotal also offers an API for programs such as Autoruns that makes it possible not only to scan many files at once, but also to do so much more efficiently by uploading only file hashes rather than entire files. If VirusTotal has recently received a file with the same hash, it returns the results from the most recent scan rather than performing the scan again.

You can analyze all autostart entries by enabling Check VirusTotal.com in the Scan Options dialog box and rescanning. Autoruns uploads file hashes to VirusTotal.com and writes “Hash submitted...” in the VirusTotal column. As results come back, Autoruns replaces the text in that column with the number of engines that flagged the file out of the total number of engines that returned results, rendered as a hyperlink, as shown in Figure 4-5. As an additional visual indicator, the link is colored red if any engines flagged the file as suspicious. Click the link to open the webpage where you can see details of the results. If VirusTotal has no record of the file’s hash, Autoruns reports “Unknown.”

If you also enable Submit Unknown Images in the Scan Options dialog box, Autoruns automatically uploads the entire file to VirusTotal in response to an “Unknown” report. Uploading and scanning complete files can take several minutes, during which time Autoruns displays a “Scanning...” hyperlink in the VirusTotal column. Click that link to view the progress of the analysis.

You can also analyze items one at a time by right-clicking an autostart and choosing Check VirusTotal from the popup menu. Autoruns sends the file’s hash to VirusTotal and reports the engines’ results for that entry or “Unknown.” You can then upload the full file by right-clicking the entry again and choosing Submit To VirusTotal (if it was unknown) or Resubmit To VirusTotal (to force a new scan).

![Autoruns with VirusTotal results.](image)
You have to agree to VirusTotal’s terms of service before using the Sysinternals utilities to query VirusTotal. On first use of VirusTotal, Autoruns will open your default web browser to the VirusTotal terms of service page and prompt you in a message box to agree with the terms before proceeding.

See Chapter 3, “Process Explorer,” for additional considerations regarding VirusTotal analysis, and in particular regarding uploading files to the VirusTotal service.

Hiding entries
The default list of ASEPs is always large because, as mentioned earlier, Windows itself makes extensive use of ASEPs. Typically, Windows’ own autostart entries are not of interest when troubleshooting. Likewise, autostart entries from other Microsoft-published software such as Microsoft Office are usually not the droids you’re looking for. And when enabling VirusTotal analysis, you’re probably more interested in inspecting the non-zero results than the entries that no antimalware engine has marked.

Autoruns offers several choices on the Options menu to show only those more-interesting entries, and a “filter” feature on the Autoruns toolbar to show only items containing the text you specify. None of these options requires rescanning the system; they manipulate the previously-collected results and can show hidden entries again instantly on demand.

You can choose to hide Windows and Microsoft autostart entries from the display by enabling the Hide Windows Entries or Hide Microsoft Entries from the Options menu. The Hide Windows Entries option is enabled by default. Enabling Hide Microsoft Entries also enables Hide Windows Entries. If the entry is a hosting process such as Cmd.exe or Rundll32.exe, the filter options’ logic is based on whether the target file is a Windows or Microsoft image and whether it is signed.

The behavior of these two options depends on whether Verify Code Signatures is also enabled. If signature verification is not enabled, Hide Windows Entries omits from the display all entries for which the target image file has the word “Microsoft” in the version resource’s Company Name field, and for which the image file resides in or below the %windir% directory. Hide Microsoft Entries checks only for “Microsoft” in the Company Name field and omits those entries. As mentioned earlier, it is easy for anyone to create a program that gets past this check, so the Verify Code Signatures option is highly recommended.

If signature verification is enabled, Hide Windows Entries omits entries that are signed with the Microsoft Windows code-signing certificate. (Windows components are signed with a different cer-

---

2 Cultural reference: “These aren’t the droids you’re looking for” is a quote from the film, Star Wars IV: A New Hope.
Certificate from other Microsoft products.) Hide Microsoft Entries omits entries that are signed with any Microsoft code-signing certificate that chains to a trusted root certificate authority on the computer.

**Note** Some files that ship with Windows, particularly drivers, are provided by third parties and have a third-party name in the Company Name field of the file’s version resource, but they are catalog signed with the Windows code-signing certificate. Consequently, these entries can be hidden when signature verification is enabled but displayed when verification is not enabled. The SigCheck utility described in Chapter 9 reports both the Company Name and the name from the signing certificate. The AutorunsC utility described later in this chapter can report both also.

If you enable Hide VirusTotal Clean Entries in the Options menu, Autoruns removes from the display all entries for which VirusTotal reports zero issues. Autoruns shows only entries that are flagged by one or more VirusTotal engines, that are unknown to VirusTotal, or that couldn’t be queried because the file couldn’t be found or was inaccessible to Autoruns. On a typical system, this option should hide most entries. Note that when a small number of the VirusTotal engines report an issue, it is usually a false positive.

Another great way to find items of interest is to type search text in the Filter text entry field in the toolbar, as shown in Figure 4-6. As you type, Autoruns limits the displayed entries to rows that contain the exact (case-insensitive) text that you type. To remove the filter, simply delete the text from the entry field.

![FIGURE 4-6](image)

The Filter text box limits Autoruns results to entries containing the word “onedrive.”

By default, Autoruns displays a shaded row only for ASEPs that have entries configured within them and that are not hidden. If Hide Empty Locations is disabled in the Options menu, Autoruns displays a shaded row for every ASEP that it scans, whether it has entries or not. Autoruns scans a tremendous number of ASEPs, so this increases the amount of output dramatically. Disabling this option can be useful to verify whether particular ASEPs are scanned, or to satisfy curiosity.

Scan and filter selections from the Options menu are displayed in the status bar and are saved in the registry. They’ll remain in effect the next time the same user starts Autoruns.
Getting more information about an entry

Right-clicking an entry displays the Entry submenu as a popup context menu. Five of those menu items use other programs to display more information about the selected entry than is displayed in Autoruns:

- **Jump To Entry**  Opens the location where the autostart entry is configured. For ASEPs configured in the registry, Jump To Entry starts the registry editor (Regedit.exe) and sends it simulated keystrokes to navigate to the autostart entry. (If Regedit does not navigate to the correct location the first time, try the Jump To Entry command again.) For ASEPs configured in the file system, Jump To Entry opens a new Windows Explorer folder window in that location. For Scheduled Tasks, Jump To opens the Task Scheduler user interface; however, it does not try to navigate to the selected task. Note that Autoruns’ driving of the navigation of Regedit requires that Autoruns not be running at a lower integrity level than Regedit.

- **Jump To Image**  Opens a new Windows Explorer folder window with the file identified as the target image selected.

- **Process Explorer**  If the image path is an executable (as opposed to a script or DLL file) and a process with that name is still running, Autoruns tries to get Process Explorer (Procexp) to display its Process Properties dialog box for the process. For this option to work, Procexp needs to be in the same directory with Autoruns, found in the path, or already running. If Procexp is already running, it cannot be at a higher integrity level than Autoruns. For example, if Autoruns is not elevated and Procexp is, this option will not work.

- **Search Online**  Initiates an online search for the file name using your default browser and search engine.

- **Properties**  Displays the Windows Explorer file Properties dialog box for the target image path.

Viewing the autostarts of other users

If Autoruns is running with administrative rights, it adds a User item to the menu, listing the account names that have logged on to the computer and have an accessible user profile. Selecting a user account from that menu rescans the system, searching that user’s ASEPs, including the Run keys under that user’s HKCU and the Startup directory in that user’s profile. If Show Only Per-User Locations is selected in the Scan Options dialog box, Autoruns displays only per-user ASEPs and hides all machinewide ASEPs.

One example of when this option is useful is if a standard user has installed some harmful software. With only standard user privileges, only the user’s per-user ASEPs could have been modified. Software that has only standard user privileges cannot modify systemwide settings nor touch the accounts of other users on the system. Rather than logging on and allowing that malware to run—and possibly interfering with an Autoruns scan—you can log on to the system with an administrative
account, start Autoruns, select the potentially compromised account from the User menu, inspect the user’s ASEPs, and perform a cleanup if problems are identified. Enabling the Scan Only Per-User Locations option makes this task even easier by hiding all the ASEPs that the non-admin user could not have configured.

Viewing ASEPs of an offline system

Autoruns allows you to view the ASEPs of an offline instance of Windows from a different, known-good instance of Windows. This can be helpful in several scenarios:

- If Windows will not start, offline analysis can identify and remove faulty or misconfigured ASEPs.
- Malware, and rootkits in particular, can prevent Autoruns from accurately identifying ASEPs. For example, a rootkit that intercepts and modifies registry reads can hide the content of selected keys from Autoruns. By taking the system offline and viewing its ASEPs from an instance of Windows in which that malware is not running, those entries will not be hidden.
- Malicious files on your system might appear to be signed by a trusted publisher, when in fact the root certificate might also have come from the attacker. A known-good system in which the bogus certificate is not installed will fail the signature verification for those files.

To perform offline analysis, Autoruns must run with administrative rights and must have access to the offline instance’s file system. Choose Analyze Offline System from the File menu, and then identify the target’s Windows (System Root) directory and a user’s profile directory, as shown in Figure 4-7. Autoruns then scans that instance’s directories and registry hives for its ASEPs. Note that the registry hives cannot be on read-only media.

To perform offline analysis, Autoruns must run with administrative rights and must have access to the offline instance’s file system. Choose Analyze Offline System from the File menu, and then identify the target’s Windows (System Root) directory and a user’s profile directory, as shown in Figure 4-7. Autoruns then scans that instance’s directories and registry hives for its ASEPs. Note that the registry hives cannot be on read-only media.

![FIGURE 4-7 Picking system and user profile directories of an offline system.]

Changing the font

Choose Font from the Options menu to change the font Autoruns uses to display its results. Changing the font updates the display immediately.
When you launch Autoruns for the first time, all autostart entries on the system are displayed in one long list on the Everything tab. As Figure 4-8 shows, the display includes up to 19 other tabs that break down the complete list into categories.

**FIGURE 4-8** Autostart categories are displayed on up to 20 different tabs.

### Logon

This tab lists the “standard” autostart entries that are processed when Windows starts up and a user logs on, and it includes the ASEPs that are probably the most commonly used by applications. They include the various Run and RunOnce keys in the registry, the Startup directories in the Start menu, computer startup and shutdown scripts, and logon and logoff scripts. It also lists the initial user session processes, such as the Userinit process and the desktop shell. These ASEPs include both per-user and systemwide locations, and entries designed for control through Group Policy. Finally, it lists the Active Setup\Installed Components keys, which although never publicly documented or supported for third-party use have been reverse-engineered and repurposed both for good and for ill.

The following lists the Logon ASEPs that Autoruns inspects on a particular instance of an x64 version of Windows 10.

<table>
<thead>
<tr>
<th>The Startup directory in the “all users” Start menu</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%ALLUSERSPROFILE%\Microsoft\Windows\Start Menu\Programs\Startup</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Startup directory in the user’s Start menu</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Per-user ASEPs under HKCU\Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Microsoft\Windows\CurrentVersion\Run</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Runonce</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\RunonceEx</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Load</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Per-user ASEPs under HKCU\Software—64-bit only</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run</td>
</tr>
<tr>
<td>HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce</td>
</tr>
</tbody>
</table>
### Per-user ASEPs under HKCU\Software intended to be controlled through Group Policy

- HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\Shell
- HKCU\Software\Policies\Microsoft\Windows\System\Scripts\Logon
- HKCU\Software\Policies\Microsoft\Windows\System\Scripts\Logoff

### Systemwide ASEPs in the registry

- HKLM\Software\Microsoft\Windows\CurrentVersion\Run
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
- HKLM\Software\Microsoft\Active Setup\Installed Components
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Runonce
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\RunonceEx
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\IconServiceLib
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\AlternateShells\AvailableShells
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\AppSetup
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Taskman
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\VmApplet
- HKLM\System\CurrentControlSet\Control\SafeBoot\AlternateShell
- HKLM\System\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\StartupPrograms
- HKLM\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\InitialProgram

### Systemwide ASEPs in the registry, intended to be controlled through Group Policy

- HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\Shell
- HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Logon
- HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Logoff
- HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Startup
- HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Shutdown
- HKLM\Software\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Startup
- HKLM\Software\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Shutdown

### Systemwide ASEPs in the registry—64-bit only

- HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
- HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce
- HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnceEx
- HKLM\Software\Wow6432Node\Microsoft\Active Setup\Installed Components

### Systemwide ActiveSync ASEPs in the registry

- HKLM\Software\Microsoft\Windows CE Services\AutoStartOnConnect
- HKLM\Software\Microsoft\Windows CE Services\AutoStartOnDisconnect

### Systemwide ActiveSync ASEPs in the registry—64-bit only

- HKLM\Software\Wow6432Node\Microsoft\Windows CE Services\AutoStartOnConnect
- HKLM\Software\Wow6432Node\Microsoft\Windows CE Services\AutoStartOnDisconnect
Explorer

The Explorer tab lists common autostart entries that hook directly into Windows Explorer and usually run in-process with Explorer.exe. Again, although most entries are systemwide, there are a number of per-user entries. Key entries on the Explorer tab include the following:

- Shell extensions that add context menu items, modify property pages, and control column displays in folder windows
- Namespace extensions such as the Desktop, Control Panel, and Recycle Bin, as well as third-party namespace extensions
- Pluggable namespace handlers, which handle standard protocols such as http, ftp, and mailto, as well as Microsoft or third-party extensions such as about, mk, and res
- Pluggable MIME filters

On 64-bit versions of Windows, in-process components such as DLLs can be loaded only into processes built for the same CPU architecture. For example, shell extensions implemented as 32-bit DLLs can be loaded only into the 32-bit version of Windows Explorer—and 64-bit Windows uses the 64-bit Explorer by default. Therefore, these extensions might not appear to work at all on 64-bit Windows.

The following lists the Explorer ASEP locations that Autoruns inspects on a particular instance of an x64 version of Windows 10.

<table>
<thead>
<tr>
<th>Per-user ASEPs under HKCU\Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Classes*\ShellEx\ContextMenuHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes*\ShellEx\PropertySheetHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\AllFileSystemObjects\ShellEx\ContextMenuHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\AllFileSystemObjects\ShellEx\DragDropHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\AllFileSystemObjects\ShellEx\PropertySheetHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Cl sid{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}\Inprocserver32</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Directory\Background\ShellEx\ContextMenuHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Directory\ShellEx\ContextMenuHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Directory\ShellEx\CopyHookHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Directory\ShellEx\DragDropHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Directory\ShellEx\PropertySheetHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Drive\ShellEx\ContextMenuHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Folder\ShellEx\ColumnHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Folder\ShellEx\ContextMenuHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Folder\ShellEx\DragDropHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Folder\ShellEx\ExtShellFolderViews</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Folder\ShellEx\PropertySheetHandlers</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Protocols\Filter</td>
</tr>
<tr>
<td>HKCU\Software\Classes\Protocols\Handler</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Ctf\LangBarAddin</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Internet Explorer\Desktop\Components</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad</td>
</tr>
</tbody>
</table>

---

3 Windows Explorer was renamed “File Explorer” beginning in Windows 8.
Systemwide ASEPs in the registry

HKLM\Software\Classes\\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\\ShellEx\PropertySheetHandlers
HKLM\Software\Classes\AllFileSystemObjects\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\AllFileSystemObjects\ShellEx\DragDropHandlers
HKLM\Software\Classes\AllFileSystemObjects\ShellEx\PropertySheetHandlers
HKLM\Software\Classes\Directory\Background\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\Directory\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\Directory\ShellEx\CopyHookHandlers
HKLM\Software\Classes\Directory\ShellEx\DragDropHandlers
HKLM\Software\Classes\Directory\ShellEx\PropertySheetHandlers
HKLM\Software\Classes\Drive\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\Drive\ShellEx\DragDropHandlers
HKLM\Software\Classes\Folder\ShellEx\ColumnHandlers
HKLM\Software\Classes\Folder\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\Folder\ShellEx\DragDropHandlers
HKLM\Software\Classes\Folder\ShellEx\ExtShellFolderViews
HKLM\Software\Classes\Folder\ShellEx\PropertySheetHandlers
HKLM\Software\Classes\Protocols\Filter
HKLM\Software\Classes\Protocols\Handler

HKLM\Software\Microsoft\Ctf\LangBarAddin
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects
HKLM\Software\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad

Systemwide ASEPs in the registry—64-bit only

HKLM\Software\Wow6432Node\Classes\\ShellEx\ContextMenuHandlers
HKLM\Software\Wow6432Node\Classes\\ShellEx\PropertySheetHandlers
HKLM\Software\Wow6432Node\Classes\AllFileSystemObjects\ShellEx\ContextMenuHandlers
HKLM\Software\Wow6432Node\Classes\AllFileSystemObjects\ShellEx\DragDropHandlers
HKLM\Software\Wow6432Node\Classes\AllFileSystemObjects\ShellEx\PropertySheetHandlers
HKLM\Software\Wow6432Node\Classes\Directory\Background\ShellEx\ContextMenuHandlers
HKLM\Software\Wow6432Node\Classes\Directory\ShellEx\ContextMenuHandlers
HKLM\Software\Wow6432Node\Classes\Directory\ShellEx\CopyHookHandlers
HKLM\Software\Wow6432Node\Classes\Directory\ShellEx\DragDropHandlers
HKLM\Software\Wow6432Node\Classes\Directory\ShellEx\PropertySheetHandlers
HKLM\Software\Wow6432Node\Classes\Drive\ShellEx\ContextMenuHandlers
HKLM\Software\Wow6432Node\Classes\Drive\ShellEx\DragDropHandlers
HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\ColumnHandlers
HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\ContextMenuHandlers
HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\DragDropHandlers
HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\ExtShellFolderViews
HKLM\Software\Wow6432Node\Classes\Folder\ShellEx\PropertySheetHandlers
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad

Internet Explorer

Internet Explorer is designed for extensibility, with interfaces specifically exposed to enable Explorer bars such as the Favorites and History bars, toolbars, and custom menu items and toolbar buttons. And Browser Helper Objects (BHOs) enable almost limitless possibilities for extending the capabilities and user experiences for Internet Explorer.
However, because so much of users’ computer time is spent in a browser, and because much of the high-value information that users handle (such as passwords and credit card information) goes through the browser, it has become a primary target of attackers. The same programmatic interfaces that enable integration with third-party document readers and instant messaging have also been used by spyware, adware, and other malicious endeavors.

The following lists the Internet Explorer ASEP locations that Autoruns inspects on a particular instance of an x64 version of Windows 10.

<table>
<thead>
<tr>
<th>Per-user ASEPs under HKCU\Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Microsoft\Internet Explorer\Explorer Bars</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Internet Explorer\Extensions</td>
</tr>
<tr>
<td>HKCU\Software\Microsoft\Internet Explorer\UrlSearchHooks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Systemwide ASEPs in the registry</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKLM\Software\Microsoft\Internet Explorer\Explorer Bars</td>
</tr>
<tr>
<td>HKLM\Software\Microsoft\Internet Explorer\Extensions</td>
</tr>
<tr>
<td>HKLM\Software\Microsoft\Internet Explorer\Toolbar</td>
</tr>
<tr>
<td>HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Per-user and systemwide ASEPs in the registry—64-bit only</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Wow6432Node\Microsoft\Internet Explorer\Explorer Bars</td>
</tr>
<tr>
<td>HKCU\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions</td>
</tr>
<tr>
<td>HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Explorer Bars</td>
</tr>
<tr>
<td>HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions</td>
</tr>
<tr>
<td>HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Toolbar</td>
</tr>
<tr>
<td>HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects</td>
</tr>
</tbody>
</table>

**Scheduled Tasks**

The Scheduled Tasks tab displays entries that are configured to be launched by the Windows Task Scheduler. The Task Scheduler allows programs to be launched on a fixed schedule or upon triggering events, such as a user logging on or the computer being idle for a period of time. Commands scheduled with At.exe also appear in the list. The Task Scheduler was greatly enhanced in Windows Vista, so Windows now makes heavy use of it, and the list on the Scheduled Tasks tab will generally be long unless you hide verified Windows entries.

Because tasks can actually be disabled in Windows (unlike Start menu items), clearing the check box next to a scheduled task in Autoruns disables the task rather than copying it to a backup location.4

If you select Jump To Entry from the Entry menu for a scheduled task entry, Autoruns displays the Task Scheduler user interface, but it does not try to navigate to the selected entry.

---

4 “At” jobs cannot be disabled, whether using Autoruns or the Windows Task Scheduler. “At” jobs can be deleted. Note that AT.EXE was deprecated and no longer works on Windows 8 or newer.
Services

Windows services run in noninteractive, user-mode processes that can be configured to start independently of any user logging on, and that are controlled through a standard interface with the Service Control Manager. Multiple services can be configured to share a single process. A common example of this can be seen in Svchost.exe (Host Process for Windows Services), which is specifically designed to host multiple services implemented in separate DLLs.

Services are configured in the subkeys of HKLM\System\CurrentControlSet\Services. The Start value within each subkey determines whether and how the service starts.

Autoruns' Services tab lists services that are not disabled, unless they were disabled by Autoruns (indicated by the presence of an AutorunsDisabled value in the service's registry key). The content for the Description column comes from the text or the resource identified by the Description value in the configuration key. The image path column displays the path to the service executable; for Svchost services, Autoruns displays the path to the target DLL identified by the ServiceDll value in the service's key or its Parameters subkey. There are cases for some services in some versions of Windows where administrative rights are required to view the Parameters key; in these cases, Autoruns displays the path to Svchost.exe in the image path column.

Be certain you know what you are doing when disabling or deleting services. Missteps can leave your system with degraded performance, unstable, or unbootable. And again, note that disabling or deleting a service does not stop the service if it is already running.

One malware technique to watch for is a service that looks like it's supposed to be part of Windows but isn't, such as a file named svchost.exe in the Windows directory instead of in System32. Another technique is to make legitimate services dependent on a malware service; removing or disabling the service without fixing the dependency can result in an unbootable system. Autoruns' Jump To Entry feature is handy for verifying whether the service's configuration in the registry includes a DependOnService value that you can inspect for dependencies before making changes.

Drivers

Like services, drivers are also configured in the subkeys of HKLM\System\CurrentControlSet\Services, as well as in HKLM\Software\Microsoft\Windows NT\CurrentVersion\Font Drivers. Unlike services, drivers run in kernel mode, thus becoming part of the core of the operating system. Most are installed in System32\Drivers and have a .sys file extension. Drivers enable Windows to interact with various types of hardware, including displays, storage, smartcard readers, and human input devices. They are also used to monitor network traffic and file I/O by antivirus software (and by Sysinternals utilities such as Procmon and Procexp!). And, of course, they are also used by malware, particularly rootkits.
As with services, the Drivers tab displays drivers that are not marked as disabled, except those disabled through Autoruns. The *Description* value comes from the version resource of the driver file, and the image path points to the location of the driver file.

Most blue-screen crashes are caused by an illegal operation performed in kernel mode, and most of those are caused by a bug in a third-party driver. (Less common reasons for blue screens are faulty hardware, the termination of a system-critical process such as Csrss.exe, or an intentional crash triggered through the keyboard driver’s crash functionality, as described in Knowledge Base article 244139: http://support.microsoft.com/kb/244139.)

You can disable or delete a problematic driver with Autoruns. Doing so will usually take effect after a reboot. As with services, be absolutely certain you know what you are doing when disabling or deleting the configuration of drivers. Many are critical to the operating system, and any misconfiguration might prevent Windows from working at all.

### Codecs

The Codecs category lists executable code that can be loaded by media playback applications. Buggy or misconfigured codecs have been known to cause system slowdowns and other problems, and these ASEPs have also been abused by malware. The following lists the keys that are shown on the Codecs tab.

**Keys inspected under both HKLM and HKCU**

- `\Software\Classes\CLSID\{083863F1-70DE-11d0-BD40-00A0C911CE86}\Instance`
- `\Software\Classes\CLSID\{7ED96837-96F0-4812-B211-F13C24117ED3}\Instance`
- `\Software\Classes\CLSID\{ABE3B9A4-257D-4B97-BD1A-294AF496222E}\Instance`
- `\Software\Classes\CLSID\{AC757296-3522-4E11-9862-C17BE5A1767E}\Instance`
- `\Software\Classes\Filter`
- `\Software\Microsoft\Windows NT\CurrentVersion\Drivers32`

**Keys inspected under both HKLM and HKCU on 64-bit Windows**

- `\Software\Wow6432Node\Classes\CLSID\{083863F1-70DE-11d0-BD40-00A0C911CE86}\Instance`
- `\Software\Wow6432Node\Classes\CLSID\{7ED96837-96F0-4812-B211-F13C24117ED3}\Instance`
- `\Software\Wow6432Node\Classes\CLSID\{ABE3B9A4-257D-4B97-BD1A-294AF496222E}\Instance`
- `\Software\Wow6432Node\Classes\CLSID\{AC757296-3522-4E11-9862-C17BE5A1767E}\Instance`
- `\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Drivers32`

### Boot Execute

The Boot Execute tab shows you Windows native-mode executables that are started by the Session Manager (Smss.exe) during system boot. BootExecute typically includes tasks, such as hard-drive verification and repair (Autochk.exe), that cannot be performed while Windows is running. The Execute, S0InitialCommand, and SetupExecute entries should never be populated after Windows has been installed. The following lists the keys that are displayed on the Boot Execute tab.
Keys that are displayed on the Boot Execute tab

HKLM\System\CurrentControlSet\Control\ServiceControlManagerExtension
HKLM\System\CurrentControlSet\Control\Session Manager\BootExecute
HKLM\System\CurrentControlSet\Control\Session Manager\Execute
HKLM\System\CurrentControlSet\Control\Session Manager\S0InitialCommand
HKLM\System\CurrentControlSet\Control\Session Manager\SetupExecute

Image hijacks

*Image hijacks* is the term I use for ASEPs that run a different program from the one you specify and expect to be running. The Image Hijacks tab displays four types of these redirections:

- **exefile** Changes to the association of the .exe or .cmd file types with an executable command. The file-association user interfaces in Windows have never exposed a way to change the association of the .exe or .cmd file types, but they can be changed in the registry. Note that there are per-user and systemwide versions of these ASEPs.

- **htmlfile** Changes to the association of the .htm or .html file types with an executable command. Some malware that hijacks these ASEPs can come into play when you open an HTML file. Verify that the executable command is a legitimate browser.

- **Command Processor\Autorun** A command line that is executed whenever a new Cmd.exe instance is launched. The command runs within the context of the new Cmd.exe instance. There is a per-user and systemwide variant, as well as a separate version for the 32-bit Cmd.exe on 64-bit Windows.

- **Image File Execution Options (IFEO)** Subkeys of this registry location (and its echo in the 64-bit versions of Windows) are used for a number of internal and undocumented purposes. One purpose for IFEO subkeys that has been documented is the ability to specify an alternate program to start whenever a particular application is launched. By creating a subkey named for the file name of the original program and a “Debugger” value within that key that specifies an executable path to an alternate program, the alternate program is started instead and receives the original program path and command line on its command line. The original purpose of this mechanism was for the alternate program to be a debugger and for the new process to be started by that debugger, rather than having a debugger attach to the process later, after its startup code had already run. However, there is no requirement that the alternate program actually be a debugger, nor that it even look at the command line passed to it. In fact, this mechanism is how Process Explorer (described in Chapter 3) replaces Task Manager.
The following list shows the registry keys corresponding to these ASEPS that are shown on the Image Hijacks tab.

### Registry locations inspected for EXE file hijacks

<table>
<thead>
<tr>
<th>Registry location</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Classes\Exefile\Shell\Open\Command(Default)</td>
</tr>
<tr>
<td>HKCU\Software\Classes.exe</td>
</tr>
<tr>
<td>HKCU\Software\Classes\cmd</td>
</tr>
<tr>
<td>HKLM\Software\Classes\Exefile\Shell\Open\Command(Default)</td>
</tr>
<tr>
<td>HKLM\Software\Classes.exe</td>
</tr>
<tr>
<td>HKLM\Software\Classes\cmd</td>
</tr>
</tbody>
</table>

### Registry locations inspected for htmlfile hijacks

<table>
<thead>
<tr>
<th>Registry location</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Classes\Htmlfile\Shell\Open\Command(Default)</td>
</tr>
<tr>
<td>HKLM\Software\Classes\Htmlfile\Shell\Open\Command(Default)</td>
</tr>
</tbody>
</table>

### Command processor autorun keys

<table>
<thead>
<tr>
<th>Registry location</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCU\Software\Microsoft\Command Processor\Autorun</td>
</tr>
<tr>
<td>HKLM\Software\Microsoft\Command Processor\Autorun</td>
</tr>
<tr>
<td>HKLM\Software\Wow6432Node\Microsoft\Command Processor\Autorun</td>
</tr>
</tbody>
</table>

### Keys inspected for Image File Execution Options hijacks

<table>
<thead>
<tr>
<th>Registry location</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options</td>
</tr>
<tr>
<td>HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options</td>
</tr>
</tbody>
</table>

## AppInit

The idea behind AppInit DLLs surely seemed like a good idea to the software engineers who incorporated it into Windows NT 3.1. Specify one or more DLLs in the Appinit_Dlls registry key, and those DLLs will be loaded into every process that loads User32.dll (that is, virtually all user-mode Windows processes). Well, what could go wrong with that?

- The AppInit DLLs are loaded into the process during User32’s initialization—that is, while its DllMain function is executing. Developers are explicitly told not to load other DLLs within a DllMain. It can lead to deadlocks and out-of-order loads, which can lead to application crashes. And yet here, the AppInit DLL “feature” does exactly that. And yes, that has led to deadlock and application crashes.5
- A DLL that automatically gets loaded into every process on the computer sounds like a winner if you are writing malware. Although AppInit has been used in legitimate (but misguided) software, it is frequently used by malware.

Because of these problems, AppInit DLLs are deprecated and disabled by default in Windows Vista and newer. For purposes of backward compatibility, it is possible to re-enable AppInit DLL functionality, but doing so is strongly discouraged. To ensure that AppInit DLLs have not been re-enabled, verify that the LoadAppInit_DLLs DWORD value is 0 in HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows and in HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows.

---

5 Raymond Chen wrote a blog post about AppInit DLLs that he titled “AppInit_DLLs should be renamed Deadlock_Or_Crash_Randomly_DLLs”: https://blogs.msdn.microsoft.com/oldnewthing/20071213-00/?p=24183/
KnownDLLs

KnownDLLs helps improve system performance by ensuring that all Windows processes use the same version of certain DLLs, rather than choose their own from various file locations. During startup, the Session Manager maps the DLLs listed in HKLM\System\CurrentControlSet\Control\Session Manager\KnownDlls into memory as named section objects. When a new process is loaded and needs to map these DLLs, it uses the existing sections rather than searching the file system for another version of the DLL.

The Autoruns KnownDLLs tab should contain only verifiable Windows DLLs. On 64-bit versions of Windows, the KnownDLLs tab lists one ASEP, but file entries are duplicated for both 32-bit and 64-bit versions of the DLLs, in directories specified by the DllDirectory and DllDirectory32 values in the registry key. Note that the Windows-On-Windows-64 (WOW64) support DLLs are present only in the System32 directory and Autoruns will report “file not found” for the corresponding SysWOW64 directory entries. This is normal.

To verify that malware hasn’t deleted an entry from this key so that it can load its own version of a system DLL, save the Autoruns results from the suspect system and compare it against the results from a known-good instance of the same operating system. See the “Saving and comparing results” section later in this chapter for more information.

Winlogon

The Winlogon tab displays entries that hook into Winlogon.exe, which manages the Windows interactive-logon user interface. Introduced in Windows Vista, the Credential Provider interface manages the user authentication interface. Today, Windows includes many credential providers that handle password, PIN, picture-password, smartcard, and biometric logon. Most of these are shown only if you disable the Hide Windows Entry option. Third parties can supply credential providers that further customize interactive user logons.

The Winlogon tab also includes the user’s configured screen saver, which is started by Winlogon.exe after inactivity, and registered Group Policy client-side extensions (CSEs), which are DLLs that the Group Policy engine loads. The Group Policy engine used to run in the Winlogon process, but now it runs in the Group Policy Client service.
The following list specifies the registry keys that are shown on the Winlogon tab.

**Per-user specification of the screen saver**
HKCU\Control Panel\Desktop\Scrnsave.exe

**Per-user specification of the screen saver, controlled by Group Policy**
HKCU\Software\Policies\Microsoft\Windows\Control Panel\Desktop\Scrnsave.exe

**Group Policy Client-Side Extensions (CSEs)**
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\GPExtensions
HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GPExtensions

**Credential provider ASEPs**
HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\Credential Provider Filters
HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\Credential Providers
HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\PLAP Providers

**Systemwide identification of a program to verify successful boot**
HKLM\System\CurrentControlSet\Control\BootVerificationProgram\ImagePath

**ASEP for custom setup and deployment tasks**
HKLM\System\Setup\CmdLine

---

**Winsock providers**

Windows Sockets (Winsock) is an extensible API on Windows because third parties can add a *transport service provider* that interfaces Winsock with other protocols or layers on top of existing protocols to provide functionality such as proxying. Third parties can also add a *namespace service provider* to augment Winsock’s name-resolution facilities. Service providers plug into Winsock by using the Winsock *service provider interface* (SPI). When a transport service provider is registered with Winsock, Winsock uses the transport service provider to implement socket functions, such as *connect* and *accept*, for the address types that the provider indicates it implements. There are no restrictions on how the transport service provider implements the functions, but the implementation usually involves communicating with a transport driver in kernel mode.

The Winsock tab lists the providers registered on the system, including those that are built into Windows. You can hide the latter group by enabling Hide Windows Entries and Verify Code Signatures to focus on the entries that are more likely to be causing problems.

**Keys inspected for Winsock Provider Entries**
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries64
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_Catalog9\Catalog_Entries
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_Catalog9\Catalog_Entries64
Print monitors
The entries listed on the Print Monitors tab are DLLs that are configured in the subkeys of HKLM\System\CurrentControlSet\Control\Print\Monitors. These DLLs are loaded into the Spooler service, which runs as Local System.

Note One of the most common problems that affects the print spooler is misbehaving or poorly coded third-party port monitors. A good first step in troubleshooting print spooler issues is to disable third-party port monitors to see whether the problem persists.

LSA providers
This category of autostarts comprises packages that define or extend user authentication for Windows, via the Local Security Authority (LSA). Unless you have installed third-party authentication packages or password filters, this list should contain only Windows-verifiable entries. The DLLs listed in these entries are loaded by Lsass.exe or Winlogon.exe and run as Local System.

The SecurityProviders ASEP that is also shown on this tab lists registered cryptographic providers. DLLs listed in this ASEP get loaded into many privileged and standard user processes, so this ASEP has been targeted as a malware persistence vector. (This ASEP isn’t truly related to the LSA, except that, like the LSA, it represents security-related functionality.)

Keys inspected for Authentication Providers
HKLM\System\CurrentControlSet\Control\Lsa\Authentication Packages
HKLM\System\CurrentControlSet\Control\Lsa\Notification Packages
HKLM\System\CurrentControlSet\Control\Lsa\Security Packages
HKLM\System\CurrentControlSet\Control\Lsa\OSConfig\Security Packages

Keys inspected for Registered Cryptographic Providers
HKLM\System\CurrentControlSet\Control\SecurityProviders\SecurityProviders

Network providers
The Network Providers tab lists the installed providers handling network communication, which are configured in HKLM\System\CurrentControlSet\Control\NetworkProvider\Order. On a Windows desktop operating system, for example, this tab includes the default providers that provide access to SMB (file and print) servers, Microsoft RDP (Terminal Services/Remote Desktop) servers, and access to WebDAV servers. Additional providers are often visible in this list if you have a more heterogeneous network or additional types of servers that Windows needs to connect to. All entries in this list should be verifiable.
WMI

The WMI tab lists registered WMI event consumers that can be configured to run arbitrary scripts or command lines when a particular event occurs. When you select an entry on the WMI tab, the lower panel reports information about the target file, the event consumer’s full command line, and the condition, such as a WQL query, that will trigger the event consumer to execute.

When you disable a WMI entry, Autoruns replaces the entry with a clone that has the same name but with “_disabled” appended. This breaks the binding to the event filter so that it won’t execute. By re-enabling, the original name and the event binding is reestablished.

These events and bindings are stored in the WMI repository in the \ROOT\subscription namespace.

Sidebar gadgets

On Windows Vista and Windows 7, this tab lists the Sidebar Gadgets (called “Desktop Gadgets” on Windows 7) that are configured to appear on the user’s desktop. Although gadget software is often (but not always) installed in a systemwide location such as %ProgramFiles%, the configuration of which gadgets to run is in %LOCALAPPDATA%\Microsoft\Windows Sidebar\Settings.ini, which is per-user and nonroaming. Disabling or deleting gadgets with Autoruns manipulates entries in the Settings.ini file.

The image path usually points to an XML file. The gadgets that shipped with Windows Vista and Windows 7 are catalog signed and can be verified. Gadgets were discontinued after Windows 7.

Office

The Office tab lists add-ins and plug-ins registered to hook into documented interfaces for Access, Excel, Outlook, PowerPoint, and Word. On 64-bit Windows, Office add-ins can be registered to run in 32-bit or 64-bit Office versions. 32-bit add-ins are registered in Wow6432Node subkeys on 64-bit Windows.

### Keys inspected under both HKLM and HKCU

- `\Software\Microsoft\Office\Access\Addins`
- `\Software\Microsoft\Office\Excel\Addins`
- `\Software\Microsoft\Office\Outlook\Addins`
- `\Software\Microsoft\Office\PowerPoint\Addins`
- `\Software\Microsoft\Office\Word\Addins`

### Keys inspected under both HKLM and HKCU on 64-bit Windows

- `\Software\Wow6432Node\Microsoft\Office\Access\Addins`
- `\Software\Wow6432Node\Microsoft\Office\Excel\Addins`
- `\Software\Wow6432Node\Microsoft\Office\Outlook\Addins`
- `\Software\Wow6432Node\Microsoft\Office\PowerPoint\Addins`
- `\Software\Wow6432Node\Microsoft\Office\Word\Addins`
Saving and comparing results

Autoruns results can be saved to disk in two different file formats: tab-delimited text, or a binary format that preserves all the data captured. The binary format can be loaded into Autoruns for viewing at a later time or on a different system, and it can be compared against another set of Autoruns results.

In both cases, the results are read-only: they can’t be used to roll back a system to an earlier state or configuration, and after they have been captured, you cannot add or remove options to modify the saved results. You can apply or remove the filters described in the “Hiding entries” section earlier in this chapter to control which entries Autoruns displays.

Saving as tab-delimited text

Click the Save button on the toolbar; in the Save dialog box, change the Save As Type to Text (*.txt), and specify a file in which to save the current results. The data displayed on the Everything tab is written to the file in five-column or six-column tab-delimited format, depending on whether the Check VirusTotal.com option is enabled. The rows identifying the ASEP locations (the gray-shaded rows in the Autoruns display) include the location in the first column, the location’s last-modification time-stamp in the fifth column, and empty strings in the remaining columns. The rows identifying Autorun Entries that are enabled (the check boxes are selected) are written to the file prepended with a plus sign (+); those that are disabled are prepended with an X.

The text file can be imported into Microsoft Office Excel. You should specify the first column as Text instead of General so that the leading plus signs do not get interpreted as an instruction or other special character.

The tab-delimited format respects the selections on the Options menu. If Hide Empty Locations is not enabled, the file will include all ASEPs, including those that have no entries. If Hide Microsoft Entries, Hide Windows Entries, or Hide VirusTotal Clean Entries is selected, those entries will be omitted from the output. If Verify Code Signatures is selected, the Publisher column will include Verified or Not Verified, as appropriate. If Check VirusTotal.com is selected, the output adds a sixth column with the VirusTotal column’s results.

Note that Autoruns results saved in text format cannot be read back in to Autoruns.

See the section on AutorunsC later in this chapter for a scriptable way to capture Autoruns data to other text file formats.

Saving in binary (.arn) format

The Autoruns binary file format with its default .arn file extension is the Autoruns “native” file format. Click the Save icon on the toolbar, and specify a file in which to save the results, leaving the Save As Type option as Autoruns Data (*.arn). All information captured in the most recent scan is preserved, including signature verification and VirusTotal results, even for entries that are filtered from the display.
You can automate the capture of Autoruns data and save it to a .arn file with the \texttt{-a} command-line option. The following command captures the state of autostart entries on the system to outputfile.arn, using default Autoruns options:

\texttt{Autoruns -a outputfile.arn}

To add signature verification, include the \texttt{-v} option as shown in the following example. Make sure not to put it \textit{between} the \texttt{-a} and the file name: the file name must immediately follow the \texttt{-a} parameter.

\texttt{Autoruns -v -a outputfile.arn}

**Viewing and comparing saved results**

To view the .arn file on the same or another system, choose Open from the File menu and select the saved file. When Autoruns starts, it creates a file association for .arn, so you can also open a .arn file simply by double-clicking it in Explorer. You can also open a saved file from the Autoruns command line by specifying the file path without any other switches:

\texttt{Autoruns C:\Users\Mark\Desktop\outputfile.arn}

To compare the results displayed in Autoruns—whether it’s a fresh capture or from a saved file—choose Compare from the File menu and select the saved file to compare the displayed results against. Autoruns shows only the entries that have changed between the two sets, with the ones that are present only in the original set highlighted in green, and entries that are only in the “compare” file highlighted in red. Because the content of the Publisher column depends on whether signature verification is enabled, you should compare only captures that have the same signature verification selection.

**AutorunsC**

AutorunsC is a console-mode version of Autoruns that outputs results to its standard output. It is designed primarily for use in scripts. Its purpose is data collection only: it cannot disable or delete any autostart entries.

The command-line options are listed in Table 4-1.\textsuperscript{6} They let you capture all autostarts or just specific categories, verify digital signatures, query VirusTotal, omit Microsoft entries, specify a user account for which to capture autostarts or capture all user accounts’ autostarts, and output results as comma-separated or tab-separated values (CSV) or as XML. If you don’t specify any options, AutorunsC outputs just the Logon entries without signature verification and in an indented list format designed for human reading. To capture other ASEPs, add the \texttt{-a} option followed by one or more letters indicating the ASEP categories of interest, or * to capture all ASEP categories.

\textsuperscript{6} Note that AutorunsC’s command-line syntax was completely overhauled in version 13.0, which was released in January, 2015. If you have scripts designed for earlier versions of AutorunsC, you should review and update them.
Whether in the default list format, CSV, or XML, AutorunsC’s output always includes the ASEP location, entry name, description, version, publisher, image path, command line, whether the entry is disabled, and the date and time the target file was last modified, according to the file system. CSV output also includes a row for each ASEP location and when it was last modified. Note that because Windows tracks the last write time for registry keys but not for individual registry values, the “last modified” time for a registry ASEP location will be for the key and might not reflect when a specific entry was changed. When signature verification is enabled, CSV output includes both the signing name as well as the Company Name attribute from the file’s version resource.

When file hashes are requested with the –h option, AutorunsC outputs MD5, SHA-1, SHA-256, and IMPHASH\(^7\) hashes of the target file, as well as PESHA-1 and PESHA-256 hashes that are used for Authenticode signatures and that cover only the content areas and not the filler of Portable Executable (PE) files.

CSV and XML output also explicitly name the user profile to which each entry belongs, or “System-wide” for entries that apply to the entire system.

The CSV format includes column headers, and it imports easily into Excel or relational databases. The XML format is easily consumed by Windows PowerShell or any other XML consumer. For example, the following lines of PowerShell run AutorunsC, read the XML, and then display disabled items:

```powershell
$arcx = [xml]$(autorunsc -a * -x -accepteula)
$arcx.SelectNodes("/autoruns/item") | ?{ $_.enabled -ne "Enabled" }
```

**TABLE 4-1** AutorunsC command-line options

| Autostart types: [ -a * | bcdeghiklmoprsw ] |         |
|-----------------|--------|
| *               | Shows all autostart entries |
| b               | Shows boot execute entries |
| c               | Shows codecs |
| d               | Shows AppInit DLLs |
| e               | Shows Explorer add-ons |
| g               | Shows Sidebar gadgets (Windows Vista and Windows 7) |
| h               | Shows image hijacks |
| i               | Shows Internet Explorer add-ons |
| k               | Shows known DLLs |
| l               | Shows logon autostart entries (this is the default) |
| m               | Shows WMI entries |

---

\(^7\) “Import hashing,” or IMPHASH, is based on the content and order of a module’s import tables, which lists the names of libraries and the APIs used by the module. It is designed to identify related malware samples, and it is described in more detail in [https://www.mandiant.com/blog/tracking-malware-import-hashing/](https://www.mandiant.com/blog/tracking-malware-import-hashing/). VirusTotal discusses their adoption of imphash in [http://blog.virustotal.com/2014/02/virustotal-imphash.html](http://blog.virustotal.com/2014/02/virustotal-imphash.html).
### Autostart types: [-a *|bcdeghiklmoprsw]

- **n** Shows Winsock protocol and network providers
- **o** Shows Office addins
- **p** Shows printer monitor DLLs
- **r** Shows LSA security providers
- **s** Shows services and non-disabled drivers
- **t** Shows scheduled tasks
- **w** Shows Winlogon entries

### What to scan

- **user** Specifies the name of the user account for which autostart entries will be shown. Use DOMAIN\User format for domain accounts. Specify * to scan all user profiles. This option requires administrative rights.

- **–z systemroot userprofile** Scans an offline Windows system, specifying the file-system paths to the target system's Windows directory and to the target user-profile directory.

### File information

- **–h** Shows file hashes
- **–s** Verifies digital signatures
- **–u** If VirusTotal check is enabled, –u shows only files that are unknown by VirusTotal or that have non-zero detection. If VirusTotal check is not enabled, –u shows only unsigned files.

- **–v[rs]** Queries VirusTotal for malware based on file hashes. With “r” added, it opens the web browser to VirusTotal report for files with non-zero detection. With “s” added, it uploads files that report as “unknown”—that is, not previously scanned by VirusTotal. (Also, note the meaning of –u when used with the –v[rs] option.)

- **–vt** Accepts the VirusTotal terms of service (TOS) without opening the TOS webpage.

### Output format

- **–c** Prints output as comma-separated values (CSV)
- **–ct** Prints output as tab-delimited values
- **–x** Prints output as XML
- **–m** Hides Microsoft entries. If used with –s, hides signed Microsoft entries.

- **–t** Shows timestamps in normalized UTC: YYYYMMDD-hhmmss. Alphabetically sorting normalized UTC also produces a chronological sort.
 Autoruns and malware

One of the goals of most malware is to remain active on an infected system indefinitely. Malware has therefore always used ASEPs. Years ago, it usually just targeted simple locations such as the Run key under HKLM. As malware has become more sophisticated and difficult to identify, its use of ASEPs has become more sophisticated as well. Malware has been implemented as Winsock providers and as print monitors. Not only are such ASEPs locations more obscure, but the malware doesn’t show up in a process list because it loads as a DLL in an existing, legitimate process. Malware has also become more adept at infecting and running without requiring administrative privileges, because there are increasing numbers of users who only ever have standard user privileges.

In addition, malware often leverages rootkits, which subvert the integrity of the operating system. Rootkits intercept and modify system calls, lying to software that uses documented system interfaces about the state of the system. Rootkits can hide the presence of registry keys and values, files and directories, processes, sockets, user accounts, and more, or they can make software believe something exists when it doesn’t. In short, a computer on which malware has run with administrative privileges cannot be trusted to report its own state accurately. Therefore, Autoruns cannot always be expected to identify malicious autostart entries on a system.

That said, not all malware is that sophisticated, and there are still some telltale signs that can point to malware:

- Entries with a well-known publisher such as Microsoft that fail signature verification. (Unfortunately, not all software published by Microsoft is signed.)
- Entries with an image path pointing to a DLL or EXE file that is missing Description or Publisher information (unless the target file is not found).
- A common Windows component that is launched from an unusual or nonstandard location—for example, svchost.exe or another service launching from C:\Windows or C:\Windows\SysWOW64 (instead of from System32) or from C:\System Volume Information.
- Entries with names that can be mistaken for common Windows components, such as those with slight misspellings—for example, “Isass.exe” with a capital “I” instead of a lower-case “L”, “scvhost.exe” instead of “svchost.exe,” or “iexplorer.exe” with the extra “r” at the end.
- Entries for which the file date and time of the launched program correspond to when problems were first noticed or a breach is discovered to have occurred.
- Disabling or deleting an entry, pressing F5 to refresh the display, and finding the entry still present and enabled. Malware will often monitor its ASEPs and put them back if they get removed.

Malware and antimalware remains a moving target. Today’s “best practices” will seem naïve and insufficient tomorrow.
There are some entries you might come across that seem suspicious but are innocuous:

- A default installation of Windows Vista might have a small number of “File not found” entries on the Drivers tab for NetWare IPX drivers and for “IP in IP Tunnel Driver.”

- Default installations of Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 might have a WMI entry named “BVTConsumer”. This code is inoperative and can be safely ignored.

- A default installation of Windows 7 might have a small number of entries on the Scheduled Tasks tab under “\Microsoft\Windows” that show an entry name but no further information.

- As explained in the “KnownDLLs” section earlier in this chapter, on 64-bit Windows Autoruns reports “File not found” for WOW64 support DLLs in the SysWOW64 directory. These known DLLs exist only in the System32 directory.
Index

A
/accepteula switch, 13–14
access checks, 90
ACCESS DENIED events, 476–477, 482–483, 500
access rights, 320–321
access tokens, 16–17, 19–20
AccessChk utility, 5, 314–322, 409
access, viewing, 28
access rights, searching for, 320–321
command-line options, 316
effective permissions, reporting, 593
filtering features, 592–595
object integrity labels, viewing, 22
object types, 317–320
output options, 321–322
access-control model, Windows, 16–17
AccessEnum utility, 5, 337–339
account rights, 314–322
Active Directory
ADMIN_LIMIT_EXCEEDED errors, 493
database snapshots, 351–352, 358–360
forest functional level, failed raises, 492–494
Rights Management Services, 491
Active Directory management utilities, 5
AdExplorer, 351–360
AdInsight, 360–370
AdRestore, 371
active memory, 439
ActiveX controls, 473–474
address space fragmentation, 272–273
Address Windowing Extensions (AWE), 438n1, 440
AdExplorer utility, 5, 351–360
configuration settings, 360
display window, 352–353
domains, connecting, 351–352
object attributes, viewing, 355–357
objects, viewing, 354–355
search functionality, 357–358
snapshots, saving and comparing, 358–360
AdInsight utility, 5, 360–370
command-line options, 370
data capture, 361–364
Details pane, 363–364
display options, 364–365
event errors, finding, 368
Event pane, 362–363
exporting captured data, 369–370
filtering results, 368–369
forest functional level raise operations, tracing, 493–494
Go To Next Event Error button, 493
highlighting events, 366–367
saving captured data, 369
searching text, 365–366
time display options, 364–365
administrative rights, 16–18
for Autoruns, 117–118
dependencies, 526–528
malicious elevation, 549. See also Stuxnet virus
for ProcDump, 198
for Procexp, 44–45
for Procmon, 146
for PsTools utilities, 223
run-once bugs, 526–528
Administrator accounts
access, 16–17
password setting utility, 245
Write permissions, 409
ADMIN_LIMIT_EXCEEDED errors, 493
AdRestore utility, 5, 371
Advanced Windows Debugging (Hewardt and Pravat), 498
AeDebug debugger, ProcDump as, 201–202
alternate data streams (ADSs), 8–9, 391–392
!analyze -v command, 506
annotation utility, 383–387
anti-malware utilities, 571. See also malware
antivirus software
on file servers, 537–538
installation problems, 522–523
App Containers, 23–28, 200
access checks, 27–28
components, 23
directory hierarchies, 25–26
Object Manager namespace, 25–27
registry hives, 25–26
App Installer Recorder script, 617–625
app model, Windows, 23–28
AppData directory, 534
AppInit DLLs, 132–133
AppInit_DLLs ASEP, 603
Application Information (Appinfo) service, 16–17
application installation error messages

application installation error messages, 477–482
application isolation in Windows, 22–29
applications
  API calls, 21
  brokers, 25
  Capability SID, 23–25
  hangs, 510–511
  identification of, 23
  manifests, 311
  prefetch files, 489–490
  security, 23–28
  virtual desktops, running on, 382–383
AppLocker feature, 529
AppX packages, 23
ASCII strings, searching files for, 389–390
ASEPs (Autostart Extensibility Points), 113
  creation context, viewing, 586
  file system, 116–117, 122
  hiding entries, 120–121
  image hijacks, 131–132
  malware related to, 141, 602–605
  offline instances, viewing, 123
  viewing, 115
attachments, saving, 470
Authentic User Gesture (AUG), 25
authentication, 343
Autologon utility, 5, 342–343
Autoplay action, 480–481
Autorun action, 480
autorun.exe, 477
Autorun.inf file, 477–483
Autoruns utility, 4, 113–142
  administrative permissions, 117–118
  Analyze Offline System feature, 495–499
  ApplInit tab, 132–133
  AutorunsC, 138–140
  autostart categories, 122–136
  autostarts, disabling and deleting, 117
  Boot Execute tab, 130–131
  code signatures, verifying, 118
  Codecs tab, 130
  comparing results, 138
  crashes at startup or logon, 497–498
  Description column, 116
  Drivers tab, 129–130
  Explorer tab, 126–127
  filtering entries, 121
  font, changing, 123
  hiding entries, 120–121
  Image Hijacks tab, 131–132
  Image Path column, 115
  information about entries, 122
  Internet Explorer tab, 127–128
  KnownDLLs tab, 133
  Logon tab, 124–125
  LSA Providers tab, 135
  main window, 116
  malware detection and removal features, 141–142, 548, 571
  Network Providers tab, 135
  Office tab, 136
  offline analysis, viewing, 123
  overview, 115–123
  Print Monitors tab, 135
  Publisher column, 116
  saving results, 137–138
  scanning the system, 117
  scareware monitoring and analysis, 577–586
  Scheduled Tasks tab, 128
  searching, 116–117
  Services tab, 129
  Sidebar Gadgets tab, 136
  in Stuxnet virus investigation, 550
  suspicious images, 116
  suspicious processes, suspending and deleting, 576
  virus scanning autostarts, 119–120
  VirusTotal column, 116
  Winlogon tab, 133–134
  Winsock tab, 134
  WMI tab, 136
AutorunsC utility, 138–140
Autostart Extensibility Points. See ASEPs (Autostart Extensibility Points)
autostarts, 124–136
  codecs, 130
  configuration location, opening, 122
  defined, 113
  disabling and deleting, 117
  drivers, 129–130
  executables, 130–131
  gadgets, 136
  Internet Explorer entries, 127–128
  KnownDLLs, 133
  at logon, 124–125
  LSA, 135
  network providers, 135
  of other users, 122–123
  Task Scheduler entries, 128
  timestamps, 115–116
  viewing, 113–115. See also Autoruns utility
  virus scanning, 119–120
  Windows Explorer/File Explorer entries, 126–127
  Windows services, 129
  of Winlogon process, 133–134
  WMI event consumers, 136

B
backing files, 179
bad memory, 439
“Bad Network Path” error, 525
bandwidth availability, 423–424
basic disks, 419
Bcdedit utility, 404
BgInfo utility, 5, 373–381
  appearance options, 377–379
  configuration settings, saving, 379
  display data, configuring, 374–377
  output options, 379–381
  updating other desktops, 381
“Blue Screen of Death” (BSOD) crash simulator, 463–464
blue-screen crashes, 130
Bluescreen Screen Saver, 6, 463–464
Blu-ray drives, stuttering, 518–521
boot configuration database (BCD) disk signatures, 403–405
boot device data in registry, 405
boot logging, 175–176, 488–490, 602–603
hangs, troubleshooting, 511
during logon sequence, 522–523
booting in debug mode, 285
broker processes, 25
browser hijackers, 575
“buddy system” malware, 54, 572
buffer overflows, 150
bugs, 193, 529. See also debugging

C
cached memory, 439
caches, processors mapped to, 452–453
call stacks, 30–31, 489
displaying, 98, 156–158
file-write operations, suspicious, 558–559
finding, 510
inspecting, 469
monitoring summary, 187–188
return address display conventions, 31
Capability SIDs, 23–25
Caps Lock keystrokes, converting into Control keystrokes, 464
Carnegie Mellon University Computer Emergency Response Team (CERT) Autorun disabling trick, 481
certificate stores, dumping contents, 302
certificates, verifying, 302. See also signature verification
Chen, Raymond, 132n5
Citrix ICA client, sharing violations, 486–490
client-side APIs, 360
client-side extensions (CSEs), 133–134
ClockRes utility, 6, 459
clones of processes, 210–211
CloseHandle API, 22
clusters, volume, graphical representation, 412
code paths
displaying, 30
process access, 437–438
code signatures, verifying with Autoruns, 118
codecs, autostarts, 130
Cogswell, Bryce, 3, 41
ColdFusion DLLs, calling registry enumeration API, 514–516
COM components, run-once bugs, 526–528
command line
/accepteula switch, 13–14
AccessChk options, 316
AdInsight options, 370
Disk2Vhd options, 403
LiveKd examples, 291
ProcDump syntax, 195–197, 204–207
process, 620
ProcExp options, 110
Procmon options, 177, 180–182
PsExec options, 227–232
PsTools options, 254–256
SigCheck syntax, 304
syntax, displaying, 220
VMMap options, 274
Command Processor Autorun, 131
commands
!analyze -v, 498, 506
!critlist, 540
fsutil hardlink, 393
fsutil hardlink list filename, 395
fsutil reparsepoint, 393
!locks, 540
mklink, 393
NET FILE, 232
PsService, 246–251
Run As, 340–341
Run As Administrator, 17, 341
Runas.exe, 16–17
commit charge
displaying, 70, 103–104
dumps, triggering, 205
communication utilities, 6
PsPing, 423–432
TCPView, 433–434
Whois, 434–435
community support forum, 3
compatibility bugs, 529
compressed files
fragmentation of, 416
secure deletion, 348
Conficker, 480–481
Config utility
contiguous files, creating, 417–418
defragmenting files, 414–416
free space, analyzing fragmentation, 416–417
configuration information, displaying as desktop wallpaper, 373–381
Configuration Manager object types, 21
console output redirection, 225–226
console sessions, 36
console utilities
EULA, 226
file, 389–399
remote enabling, 224
context switches, 44
Contig utility
contiguous files, creating, 417–418
free space, analyzing fragmentation, 416–417
continuous monitoring, 588
control, returning after exceptions, 496
cookies for client variables, 516
core dumps, 193
CoreInfo utility, 6, 449–454
Cottingham, Greg, 600–605
CPU sockets, processors mapped to, 454
CPU usage
measuring, 43–44
runaway threads and, 510–511, 514
spikes, analyzing, 541–543
systemwide, 70, 103
crash dumps, 193
analyzing, 498
capturing, 504–505
kernel targets, 287–288
crashes, 468
  !analyze -v command, 498
analyzing, 506
on registry access, 502–503
on registry permissions, 501–502
of SearchFilterHost.exe, 505–507
of SearchProtocolHost.exe, 505–507
at startup or logon, 497–498
triggers, 496
troubleshooting, 495–507
unbootable computers, 498–499
Create Symbolic Links privilege, 393
CreateFile events, 525
Credential Provider, 133
critical section locks, 540
!critlist command, 540
cross-process memory functions, 20
cryptographic operations, 530–532
Ctrl2Cap, 6, 464
custom debug output, 213
cybersecurity, 588, 611. See also malware; security

D
data, process access, 437–438
data caches, processors mapped to, 452–453
database queries, sluggish, 543
Dbghelp.dll, 33
DcomLaunch service, 620
Debug Programs privilege, 28
debuggers
  exceptions and, 497
  launching, 55
  ProcDump as, 506–507
debugging, 275
  Dbghelp.dll, 33
debug mode, booting in, 285
  of Hyper-V guest virtual machines, 285, 290
disk caching, 25–26
kernel debuggers, 285–292
  kernel-mode output, 278–279
  output events, injecting in Procmon traces, 190–191
  Rights Management Services, 491
  symbols, 31, 34
  user-mode output, 277–278
Debugging Tools for Windows, 506, 539
Dbghelp.dll, 33
installing, 34
  symbol files, downloading, 32–33
DebugView utility, 4, 259, 275–285
debug output, 275, 277–278
  DebugView agent, 284–285
display window, 275–277
  kernel-mode debug output, 278–279
logging output, 282–283
  monitoring traces with, 491
  saving output, 281
searching, filtering, highlighting, and limiting output, 279–281
decimal numbers, converting from hexadecimal, 462
default procedure calls (DPCs), 51
defragmenting
defragmentation API, 348
files, 413–418
solid state drives, 414
Windows, 412
deleted (tombstoned) objects, restoring, 371
deleton
delayed, 398
secure, 346–349
dependencies
  admin-rights, 526–528
  DLL, 54, 77, 148
  load order and, 457
  on malware services, 129
  missing, 475
  of Windows services and drivers, 249
Dependency Walker (Depends.exe) utility, 54, 77, 475
Desktop Gadgets autostarts.exe, 136
desktop utilities, 5
  BgInfo, 373–381
  Desktops, 382–383
  ZoomIt, 383–387
desktop wallpaper, computer-configuration information as, 373–381
Desktop Window Manager (DWM), 52, 71, 510
desktops, 37–38
  virtual, 382–383
Windows, 37–38
Desktops utility, 5, 38, 382–383
developer troubleshooting, 631–636
devices, viewing information about, 455
diagnostic utilities, 4–5
  DebugView utility, 275–285
  LiveKd utility, 285–293
  VMMAP utility, 259–274
digital signatures
  mismatched version and signature information, 554
  verification, 99, 302, 306–308, 551. See also SigCheck utility
directories, 469n
deleting, 399
  hierarchies, 25–26
  in-use, 296–300
  permissions, 317
  permissions, misconfigured, 337–339
  security-related functions on, 302
  size of, 395–396
directory servers, connecting to, 351–352
disk cache, flushing to physical disk, 408–410
disk cloning, 403
disk extents, 418–419
disk I/O, 67
disk management utilities, 5–6, 401–422
  Contig, 413–418
  Disk2Vhd, 401–408
  DiskExt, 418–419
  DiskView, 410–413
  LDMDump, 419–421
  Sync, 408–410
  VolumeID, 421–422
Disk Management (Diskmgmt.msc) utility, 406–407
disk signatures, 404, 406–408
Disk Usage (DU) utility, 395–398
Disk2Vhd utility, 5, 401–408
command-line options, 403
disk-signature collisions, 403–408
Prepare For Use In Virtual PC option, 402
DiskExt utility, 6, 418–419, 608
disks
attaching, 403
basic, 419
dynamic, 419
Master Boot Record, 404
offline mode, 406
physical, 401–410
physical-to-virtual conversion, 401–408
raw access events, 330
signature collisions, 403–408
signatures, 403
VHD images of, 401–408
DiskView utility, 6, 410–413
Cluster Properties dialog box, 412
dump format, 413
File Errors dialog box, 411
files, cluster view, 412–413
Volume Properties dialog box, 413
Dissmeyer, Joe, 522–523
DLLs, 31
dependencies, 54, 77, 148
digital signatures, 78
export tables, 31
image and memory strings, 78
image signatures, 99
listing, 293–296
process, 72–79
Properties dialog box, 77–78
relocated, 76
searching for, 473–474
VirusTotal.com results, 78
DoesNotExist in path names, 478
domain accounts, 223
domain controllers
NTLM use, 625, 627–628
SID definitions on, 234
domain registration lookups, 434
domain user account password-setting utility, 245
domains, connecting to, 351–352
downloaded files, alternate data streams and, 391
downloading Sysinternals utilities, 7–9
drive letters, mysterious, 607–610
drivers
autostarts, 129–130
definitions of, 609–610
load events, 326
load order, 457–458
locked memory, 440
viewing information about, 455
drives. See also disk management utilities; disks
flushing, 408–410
ID number, changing, 422
DU utility, 5
dump files
capturing, 539
contents, specifying, 209–211
criteria, specifying, 204–207
of Exchange process, 538–539
kernel dumps, 288–290
memory, 193
reason comments, 216
viewing in debugger, 216–217
dynamic disks, 419
E
echo request packets, 423
effective permissions, 314, 593. See also permissions
elevated command prompt, 17. See also User Account
Control (UAC)
email
attachments, saving, 470
delayed, 523
embedded strings, searching files for, 389–390
encrypted files, secure deletion, 348
Encrypting File System (EFS), 346
End User License Agreement (EULA), 13–14, 226
endpoints, listing, 433–434
Enhanced Mitigation Experience Toolkit (EMET), 502–503
timeout monitoring, 336
errors, 468
ACCESS DENIED, 476–477, 500
ActiveX control failed registration, 473–474
application installation, 477–482
“Bad Network Path,” 525
crashes, 496
File In Use, 471–472
folder associations, missing, 483–486
forest functional level, failed raises, 492–494
locked folders, 469–470
NAME NOT FOUND, 485
with Network Location Awareness (NLA) service, 501–502
reproducing and tracing, 468
resource access conflicts, 468
with Rights Management Services, 491
sharing violations, 487–489
text files, unreadable, 482–483
troubleshooting, 468–494
unknown, 472–473
User Environment, 486
user profile load, 486–490
EulaAccepted registry value, 14
event IDs, 241
event logs, 323. See also logging
displaying records, 241–244
permissions, viewing, 318
Event Properties dialog box, 153–158
Event Viewer, 241, 325, 336, 520, 626
events
counting occurrences, 189
details, viewing, 154–155
driver loaded, 326
events (continued)
file creation time changes, 327–328
file system, 148
filtering in Procmon, 474–475
image loaded, 326–327
long gaps between, 528–533
monitoring. See Process Monitor (Procmon)
network, 148
network connection detected, 328–329
ProcDump-generated, 159
process, 148
process creation, 324–325
process details, viewing, 155–156
process termination, 326
profiling, 148, 158–159
raw disk and volume access, 330
registry, 148
Sysmon error reports, 331
Sysmon events recorded, 324–331
Sysmon service state changes, 330
thread call stack details, viewing, 156–158
thread creation, 329–330
viewing information about, 455
Excel, PsPing histogram data charts, 432
exception handlers, 496
exceptions
filtering with ProcDump, 504–505
first-chance, 497, 504–505
hardware, 496
monitoring, 208–209
return of control, 496
second-chance, 497
software, 496–497
sources, 208
unhandled, 496
Exchange Server
high-item-count folders, 543
Store.exe usage spikes, 541–543
troubleshooting, 538–543
executables, 19, 31
autostarts, 130–131
image signatures, verifying, 99
strings in, 581–582
symbol files, 31
Executing, 21
exefile redirections, 131
exit codes, 225
export tables, 31
extents, disk, 418–419
F
FAT drive ID number, changing, 422
File Explorer autostart entries, 126–127
File In Use errors, 471–472
file servers, antivirus software on, 537–538
file share security settings, viewing, 339–340
file system
ASEPs, 116–117, 122
directories, 34
drivers, 67
file events, 148
modifications, processing, 622
objects, 81–82, 469n
offline analysis, 123, 140
Fileinfo file system minifilter driver, 490
file_or_directory parameter, 389–392
files
antivirus analysis, 100–101
cached and noncached reads, 535–536
clusters, 412–413, 415–416
contiguous, 417–418
creation timestamp changes, 327–328
defragmenting, 413–418
delayed delete, 398
delayed operations, 399
deletion, post-reboot, 398–399
events summary, 184–186
fragment boundaries, 412
fragmentation, analyzing, 415–416
handles, displaying information about, 296–300
hard links to, 392
hashes, 302
information about, displaying, 302, 310–312
in-use, identifying, 296–300
jumping to, 160
management utilities, 5, 389–399
mapping, 20, 416, 444
modifications, capturing, 618
moving, post-reboot, 399
open errors, 533–538
permissions, misconfigured, 337–339
post-reboot utilities, 398–399
Properties dialog box, 122
Read-Only permissions, 500
remotely opened, listing, 232–233
renaming, post-reboot, 398–399
searching for online, 122
secure deletion, 346–349
security-related functions on, 302. See also SigCheck utility
sharing, 487–489, 602
signature verification, 302, 306–308
symbolic links, 393
virus scanning, 119. See also VirusTotal analysis
file-write operations, suspicious, 558–559
filter manager callbacks, 537
FindLinks utility, 5, 394–395
findstr, 389
firewall rules, 427–428
first-chance exceptions, 497, 504–505
folders, 469n
associations, missing, 483–486
redirection, 534
forest functional level, failed raises, 492–494
fragmentation
address space, 272–273
analyzing, 415–416
free space, 348, 416–417, 439
fsutil hardlink command, 393
fsutil hardlink list filename command, 395
fsutil reparsepoint command, 393

642
FTP connections, malware-related, 588–592
full dump files, 210
functions, calling sequences, 30

G

gadgets, autostarts, 136
Garnier, Thomas, 323n5
gears, viewing information about, 455
GetPrivateProfileString, 479–480
ghost windows, 510–511
global object namespace names, 608
Graphics Processing Unit (GPU)
performance, displaying, 70
per-process attributes, displaying, 68
runaway, 587
systemwide metrics, 106–107
groups
Deny flag, 91
rights, displaying, 314–322
GUI threads, 39

H

Handle utility, 5, 259, 296–300
closing handles, 300
handle counts, 299–300
search feature, 297–299
handles, 21–22
access rights and, 80
attributes, displaying, 81–82
closing, 22, 82, 300
counts, 299–300, 456
information about, displaying, 79–83, 296–300
list, 19
open, searching, 470–471
Properties dialog box, 82–83
security descriptor, 83
unnamed objects, 82
hangs, 193, 211, 226
Outlook, 538–543
run-once bugs, 526–528
troubleshooting, 510–511
hard links, 392–395
hardware attributes, displaying as desktop wallpaper, 375
hardware exceptions, 496
hashes, comparing, 310–311
Hewardt, Mario, 498
Hex2Dec, 6, 462
hexadecimal numbers, converting to decimal, 462
hotfix information, displaying, 236
htmlfile redirections, 131
hung windows, 510–511
Hyper-V guest virtual machines (VMs), 285, 290, 401
load events, 326–327
paths, viewing, 115–116
signatures, verifying, 99
suspicious, 116
version information, 540
virus scanning, 55, 100–101
immersive processes, 47
impersonation, 91, 227
index-checking bug, 566
infinite loops, 511–512
.ini files, 480–482
.ini-file APIs, 480
IniFileMapping, 480–483

Inside WindowsDebugging(Soulamí), 498
integrity labels of object security descriptors, 22
integrity levels (ILs), 22, 40
interactive sessions, 36, 231
Internet
content, unblocking, 8–9
SysInternals, running from, 10
Internet Control Message Protocol (ICMP), 423–424
Internet Explorer
austostart entries, 127–128
Protected Mode, 18
SysInternals, running from, 10
Interrupts pseudo-process, 51, 74
intruders, tracking, 323–337. See also malware; security
in-use files and directories, identifying, 296–300
I/O
displaying, 42, 70, 88
per-process, displaying, 64–67
systemwide metrics, 102, 105
I/O Manager, 21
Ionescu, Alex, 15, 29
IP (Internet Protocol), 424n2
IP address lookups, 433–434
IPv4 endpoints, listing, 433–434
IPv6 endpoints, listing, 433–434
IPv6-ICMP, 424

J

Jackson, Chris, 529
Java updater, fake, 574–576
job objects (jobs), 19
displaying information on, 95–96
nested, 96
in Procexp, 47
Junction utility, 5, 393–394
junctions, 393–394
just-in-time debugger, ProcDump as, 506–507
K

Kerberos, 625, 628, 631–632
kernel
memory, 440
memory dump symbol files, 292
processes, 20–21
stack memory, 440
targets, 287–288
kernel debuggers, 285–292
kernel mode, 20–21
kernel objects, 72
kernel services function, 21
KeyedEvents, 455
killing processes, 237–238
KnownDLL autostarts, 133

L
large page memory, 440
latency testing, 423–424
LDAP API calls, 360–370
LDMDump utility, 6, 419–421
Leznek, Jason, 529
licensing SysInternals utilities, 13–14
Lichtel, Marty, 518
link utilities, 392–395
links, 393–395
ListDLLs utility, 5, 259, 293–296
loaded DLLs, listing, 602
malware detection and removal features, 548
live kernel targets, 287
LiveKd utility, 5, 259, 285–293
dump contents, 289–290
example command lines, 291
Hyper-V guest debugging, 290
kernel-debugger targets, 287–288
online kernel memory dump, 292
output to debugger or snapshot, 288–289
running modes, 286–287
symbol loading debugging, 291
system requirements, 286
Load Image events, 148, 501–502
LoadOrder (Loadord.exe), 6, 457–458
local kernel targets, 288
Local Security Authority (LSA)
active logon sessions, enumerating, 343–346
autostart entries, 135
logon sessions and RDS sessions, 35
Lsass.exe processes, rogue, 551–552
local Service account, running processes under, 230
local user account password-setting utility, 245
locked folders, troubleshooting, 469–470
locked memory, 440
locks, 539–540
locks debugger command, 540
logged-on users, listing, 240
logging
boot, 175–176, 488–490, 511, 522–523, 602–603
debug output, 282–283
events, 241–244, 318, 323
post-logoff, 177
profile, 487
Logical Disk Manager (LDM) database contents, displaying, 419–421
Logical Prefetcher, and sharing violations, 489–490
logical processors. See processors
logoff, post-logoff logging, 177
logon sessions, 343–346
Logon SID group, 91
logons
AppDataRoaming folder synchronization, 534
attributes, displaying as desktop wallpaper, 375
automatic, configuring, 342–343
autostarts, 124–125
delayed, 522–523
/ForceInstall option, 522–523
information, viewing, 240
processes, displaying, 51–52
user profile load errors, 486–490
LogonSessions utility, 5, 343–346
administrative privileges, 344
logon sessions, enumerating, 16
LUA Buglight, 526

M
machine SIDs, 233
Magnotti, David, 323n5
malicious activity, tracking, 323–337
Malicious Software Removal Tool (MSRT), 571
malware, 113, 545–605
ASEP-related, 141, 602–605
“buddy system,” 54
“buddy system” defense, 572
buffer overflows, 150
characteristics, 547, 570
cleaning steps, 547
in codecs, 130
Conficker, 480–481
fake Java updater, 574–576
fake system components, 600–601
FTP connections, unexplained, 588–592
GPU, runaway, 587
known-good systems and, 123
packed images, 47
process-killing, 598–599
reboot-related, 569–573
registry key write times and, 446
removing, 398
rootkits, 141
signatures, 303
signs in Autoruns, 141–142
Stuxnet, 549–569
submitting to Microsoft, 549n4
Sysinternals detection and removal features, 548–550
Sysinternals-blocking, 596–598
troubleshooting, 546–548
Windows services, misconfigured, 592–595
Windows services-related, 129
Winwebsec scareware, 577–586
Malware Protection Center portal, 549n4
Mandatory Integrity Control (MIC), 22
mapped files, 439
Properties dialog box, 77–78
RAM usage, 444–445
Margosis, Aaron, 13, 592–595, 612–618, 631–632
Marioforever virus, 602–605
Mark’s blog, 12–13
Mark’s webcasts, 13
Master Boot Record (MBR), 404
memory. See also physical memory; virtual memory

- address ranges, 443
- allocations, 262–265
- leaks, troubleshooting, 623–636
- metafile, 440
- modified, 439
- NUMA node access performance, 453
- object reuse protection, 346
- overwriting unallocated space, 347
- pages, displaying, 442–443
- processes associated with pages, 440–441
- purging, 445
- snapshots, saving and loading, 437, 446
- systemwide usage metrics, 103–105
- unknown stack addresses with write and execution permissions, 559
- usage, 60–63, 70, 259–274, 437–446
- memory dumps, 55, 193
- Memory Manager, 21
- memory pressure, 441
- messages, Windows, 39
- metadata files, defragmenting, 415
- metafile memory, 440
- Microsoft Excel, PsPing histogram data charts, 432
- Microsoft Hyper-V, 285, 290, 401
- Microsoft Malicious Software Removal Tool (MSRT), 571
- Microsoft Office, 136, 432
- Microsoft Security Essentials (MSE), 499, 597–598
- corrupted installation, 569–573
- Microsoft symbol server, 32
- Microsoft TechNet Sysinternals home, 6
- Microsoft Windows. See Windows operating system
- minidump files, 210
- Miniplus dump files, 210, 212–213
- mklink command, 393
- modified memory, 439
- motherboard CPU sockets, processors mapped to, 454
- MoveFile utility, 5, 399
- MoveFileEx API, 398
- Mrxnet.sys driver file, 549, 551
- Multiple Provider Notification Application (Mnotify.exe), 487
- mutexes, viewing information about, 455

O

- object handles, 296–300. See also handles
- Object Manager, 21
  - directory permissions, viewing, 317, 319
  - Named Objects container for apps, 25–27
  - namespace, viewing, 454–457
- objects
  - attributes, viewing and editing, 351, 354–357
  - creating, 355
  - deleted (tombstoned), restoring, 371
  - deleting, 392–393
  - favorites, defining, 353
  - handles, 21–22, 456. See also handles
  - information about, 455
  - Object Manager structure, 454
  - permissions information, 456
  - properties, 351, 354–357, 456
  - quota charges, 456
  - reference counts, 456
  - reuse protection, 346
  - security descriptors, 22
  - security descriptors, displaying, 321–322
  - types, 21
- Office
  - add-ins and plug-ins, autostarts, 136
  - Outlook, 470, 523–526, 538–543
  - offline systems, analyzing, 498–499
  - Omniture, 525
  - on-access virus detection, 537
  - open handles, searching, 470–471

N

NAME NOT FOUND errors, 485
name resolution with no symbol files, 483
Named Objects container for apps, 25–27
named pipes
  - effective permissions, reporting, 315
  - listing, 232, 458–459
named streams, 391
NET FILE command, 232
.NET Framework
  - assembly digital signatures checking, disabling, 533
  - exceptions, 208–209
  - process behaviors, displaying, 63–64
  - processes, 47
  - network attributes, displaying as desktop wallpaper, 375
  - network connections, unexplained, 611–612
  - network diagnostic utilities, 6
    - PsPing, 423–432
    - TCPView, 433–434
    - Whois, 434–435
  - network events, 148, 188, 328–329
  - Network Location Awareness (NLA) service errors, 501–502
  - network loopback, 223
  - network monitoring with Procmon, 588–592
  - network provider autostart entries, 135
  - Network Service account, running processes under, 230
  - network shares, 533–538
  - nodes, memory-access performance, 453
  - nonpage pool, 440
  - NOS Microsystems Ltd., 513
  - notification area, adding Procexp icons, 70
  - NPFS.sys, 458
  - NTFS
    - drive ID numbers, changing, 422
    - file mapping, 416
    - link utilities, 392–395
  - NTLM communications, 625–629
  - NtReadFile kernel function, 21
  - null characters, 463
  - NUMA nodes, 453
  - NX (No eXecute) fault, 506

- object handles, 296–300. See also handles
- Object Manager, 21
  - directory permissions, viewing, 317, 319
  - Named Objects container for apps, 25–27
  - namespace, viewing, 454–457
- objects
  - attributes, viewing and editing, 351, 354–357
  - creating, 355
  - deleted (tombstoned), restoring, 371
  - deleting, 392–393
  - favorites, defining, 353
  - handles, 21–22, 456. See also handles
  - information about, 455
  - Object Manager structure, 454
  - permissions information, 456
  - properties, 351, 354–357, 456
  - quota charges, 456
  - reference counts, 456
  - reuse protection, 346
  - security descriptors, 22
  - security descriptors, displaying, 321–322
  - types, 21
- Office
  - add-ins and plug-ins, autostarts, 136
  - Outlook, 470, 523–526, 538–543
  - offline systems, analyzing, 498–499
  - Omniture, 525
  - on-access virus detection, 537
  - open handles, searching, 470–471
operating system. See also Windows operating system
attributes, displaying as desktop wallpaper, 375
rollout images, 522–523
optical drives, sluggish performance, 518–521
outages during dump capture, 210–211
Outlook
  attachments, saving, 470
e-mail delays, 523–526
  hangs, 538–543
  picture-download blocking, 525–526
  remote hosts, connecting, 525
  Store.exe usage spikes, 541–543
own processes, 52

P
packed images, 47
page memory, 438n1
page pool, 440
page table, 440
page table entries (PTEs), 440
paging lists, purging, 445
partitioning, 418–419
"pass the hash" attacks, 28, 223
password-setting utility, 245
path name DoesNotExist, 478
path of execution, displaying, 30
PayPal emails, delayed, 523–526
PendMoves utility, 5, 398–399
Perform Volume Maintenance Tasks privilege, 418
performance
cached and noncached file reads, 535–536
sluggish, troubleshooting, 510–511
performance counters, triggering dumps, 205
permissions
displaying, 314–322
effective, 314
misconfigured, identifying, 337–339
on objects, 456
in registry keys, 502
on services, 476–477
volume, 409–410
physical disks, 401–410
physical memory. See also memory
  address ranges, 443
  analysis, 259–274
  pages, displaying, 442–443
  processes associated with pages, 440–441
  purging, 445
  usage, 70, 437–446
physical processors, mapping to processors, 450
Ping utility, 423
PipeList utility, 6, 458–459
Play To feature errors, 476–477
PNF files, 562–566
post-reboot file operation utilities, 398–399
Pravat, Daniel, 498
prefetch files, 489–490
presentation utility, 383–387
print monitor autostart entries, 135
print spooler, troubleshooting, 135
private symbol files, 32
private virtual address space, 19
privileges
  Debug Programs, 28
disabled, 91
  reporting by AccessChk, 319–320
 ProcDump, 4, 193–217, 538–543
    administrative rights, 198
    attaching to processes, 198–202
    auto-enabling, 201–202
call stack, finding and inspecting, 510
copying data, 49
  CPU Cycles Delta column, 44, 97
color rows and heatmaps, 45–48, 511
columns, 49, 55–69
colors, 49, 110
digital signature verification, 551
display options, 48, 108
displayed data, saving, 69
  DLL tab, 43, 72–79
  Environment tab, 91
  GPU Graph tab, 88–89
  GPU information, 587
  GPU tab, 106–107
graphs on toolbars, 70
handle search feature, 469–472
Handle View, 38, 43, 72–74, 79–83
highlighting, configuring, 47–48
highlighting, configuring, 47–48
highlighting, configuring, 47–48
hypothesis, establishing, 517
image signatures, verifying, 99
image signatures, verifying, 99
image signatures, verifying, 99
image signatures, verifying, 99
image signatures, verifying, 99
integrity level, viewing, 22
I/O tab, 105
Job tab, 95–96
keyboard shortcuts, 111
main window, 42–43, 45–72
malware detection and removal features, 548
Memory tab, 103–105
miscellaneous features, 110
.NET tab, 63–64, 94–95
overview, 41–45
Performance Graph tab, 87–88
Performance tab, 86–87
process actions, 53–55
process details, 83–96
Process Disk tab, 67
Process GPU tab, 68
Process I/O tab, 64–66
process list, 45–55
Process Memory tab, 60–63
Process Network tab, 66–67
Process Performance tab, 58–60
process tree, 50
processes, creating, 109
Properties dialog box, 83, 521
protected processes, viewing, 29
Remote Control option, 109–110
scareware monitoring and analysis, 577–586
Search dialog box, 73–74, 468
search online option, 55
Security tab, 90–91
Service column, 97
Services tab, 93
shutdown options, 110
Start Address, 97
startup and logon processes, 51–52
status bar, 71–72
strings in executable files, displaying, 581
Strings tab, 92
Stuxnet virus investigation, 550–551
system information, 102–107
system processes, 51
Task Manager, as replacement for, 109–110
TCP/IP tab, 89
text strings, comparing, 604
thread details, 96–99
thread stacks, 511–513
Threads tab, 89
TID, 96
toolbars, 69–71
tooltips, 50
user processes, 52–53
VirusTotal analysis, 55, 100–101, 575
window owners, identifying, 71
process handle table, 79–81
process IDs (PIDs), 19
mismatched, 559
reuse of, 613
Process Manager object types, 21
Process Monitor (Procmon), 4, 145–192
administrative rights, 146
advanced output, 165–166
backing files, 179
bookmarking events, 165
boot logging, 175–176, 488–490, 511, 522–523, 602–603
call stack, finding, 510
call stack symbol file information, 32–33
column display, 149, 151–153, 524
command-line options, 180–182
configuration settings, importing and exporting, 180
copying event data, 160
Count Occurrences feature, 189, 484, 502–503, 527–528, 611
Cross Reference Summary, 189
custom debug output, 190–191
display options, 147
DLL, searching for, 473–474
event class toggle filters, 484–485
Event Properties dialog box, 153–158
events, 148–160, 528–533
Exclude Events Before, 474–475
File Summary dialog box, 184–186, 533–538
filters, saving, 166–167
highlighting, 164–165, 478, 525
Include Process, 474, 478
install app, recording, 618–625
installation-related processes, filtering on, 619–620
log size, controlling, 178–179
logon operations, recording, 488
malware detection and removal features, 548
network monitoring, 588–592
Network Summary, 188
/noconnect option, 484
NTLM auditing, 625, 628–629
optical drive performance, tracing, 518–521
overview, 146–147
post-logoff logging, 177
ProcDump diagnostic data, 213–215
Process Activity Summary, 183–184, 515, 613
process command lines, 50, 620
Process Exit events, 614–616
Process Profiling events, 614–616
Process Tree, 168–169, 474, 559, 588–592, 612–613
processes with malware characteristics, displaying, 570–571
profiling events, displaying, 158–159
program versions, comparing, 631–632
quick filters, setting, 530
registry or file location, jumping to, 160
Registry Summary, 186–187
result codes, 150–151
root-cause analysis, 518–521
run-once bugs, 526–528
scareware monitoring and analysis, 577–586
searching online, 160
Stack Summary, 187–188
Stack Dump analysis, 489
in Stuxnet virus investigation, 550–551
suspend functionality, 572
symbols, configuring, 158
System process activity, 517–518
/terminate command, 39
thread stacks, 514–516
ThreadID (TID) column, 484
toolbar reference, 191–192
Process Monitor (Procmon)

traces, 169–175, 468–469, 483–486
unknown error explanations, 472–473
<unknown> module marker, 558–560
/WaitForIdle command, 39
Windows Event Log, 520
Write Category, 527
XML files, saving traces as, 613–614, 621
XML schema, 171–174

Process Properties dialog box, 23, 43, 55, 83–96, 122
Environment tab, 91
GPU Graph tab, 88–89
Image tab, 84–86
Job tabs, 95–96
.NET tabs, 94–95
Performance Graph tab, 87–88
Performance tab, 86–87
Security tab, 90–91
Services tab, 93
Strings tab, 92
TCP/IP tab, 89
Threads tab, 89
process reflection feature, 210–211
process tokens, 90–91
Process Tree dialog box, 168–169, 474, 559, 588–592, 612–613
process utilities, 4–5
DebugView, 275–285
Handle, 296–300
ListDLLs, 293–296
LiveKd, 285–293
VMMap, 259–274
processes, 19
activity summary, 183–184, 515
address space, 20
AppDomains and assemblies, 94–95
attributes, 56–58, 60–66, 84
broker, 25
calling WMI to access CD-ROM drives, 521
clones, 210–211
command line, 50, 620
commenting, 86
components of, 19
crashed, 46
creation events, 324–326
debugging, 55
defense-in-depth mitigations status, 84
defined, 19, 45
disk I/O attributes, 67
DLL dependencies, 54
DLL view, 72–79
dump files, 193–194. See also ProcDump
evironment variables, 91
executable data regions, suspicious, 554
executable image path, 50
executing at status change, 44
exit codes, 225
functions, 30
GPU attributes, 68, 88–89
groups, 95. See also job objects (jobs)
groups of, 19
handles, 79–83, 296–300
image load events, 326–327
image signatures, verifying, 99
integrity levels, 22
I/O attributes, 64–66
job information, 95–96
kernel objects, 72
killing, 54, 85
listing information about, 238–240
locks, 539–540
long-running, 524–526
malware characteristics, 570
malware that kills, 598–599
memory dumps, 55
memory usage attributes, 60–63
.NET behaviors, 63–64
.NET Framework performance, 94
network connection events, 328–329
with open files, 602
open handles, searching, 470–471
own, 46
parent-child relationships, 50, 168
paths cross-reference summary, 189
performance, 58–60, 86–88
permissions, 318
physical memory pages, 440–441
priority, 54
private memory, 439
ProcessIndex numbers, 614–615
processor access modes, 20–21
processor affinity, 53
profiling events, 158–159
Properties dialog box. See Process Properties dialog box
protected, 28–29
RAM usage, 438
remote thread creation events, 329–330
resuming, 54
runaway, 514–516
runtime environment options, 229–232
security context, 90–91
services, 93
short-lived, 612–617
signature verification, 85
startup, 54, 474
static attributes, 84
strings in image file, 91
suspending, 46, 54, 254
TCP operations, 66–67
TCP/IP information, 89
terminating, 237–238, 326
threads, 19–20, 89, 96–99. See also threads
token details, 314–322
tooltip information, 50
trustworthiness, 22
UAC elevation, 17
user-defined comments, 50
virtual and physical memory analysis, 259–274
VirusTotal results for image file, 84–85
window ownership, 85
windows station, association with, 37
working set, 438–439
processor affinity, 53
processors
access modes, 20–21
cache information, 452–453
features information, 450–452
group information, 452
identification information, 450–451
information about, 449–454
mapping to NUMA nodes, 453
mapping to physical processors, 450
mapping to sockets, 454
virtualization-related features, 454
Profile APIs, 482–483
profile logging, 487
profiling events, 148
debug output events, 190
displaying, 158–159
program associations, 484–486
programs
defined, 19
running as different user, 340–342
start failures, troubleshooting, 148
Project, file open errors, 533–538
Protected Administrator accounts, 223
protected processes, 28–29
  DLL view, 74
  light types, 29
  in Procexp, 47
Windows protection types, 29
PsExec utility, 4, 224–232
  alternate credentials, 222, 227
  command-line options, 227–228
logoff, monitoring, 177
ProcDump, running as System, 215–216
process performance options, 228–229
redirected console output, 225–226
remote connectivity options, 229
remote process exit, 225
runtime environment options, 225
PsFile utility, 4, 232–233
PsGetSid utility, 4, 233–235
PsInfo utility, 4, 235–237
PsKill utility, 4, 237–238
PsList utility, 4, 238–240
PsLoggedOn utility, 4, 240
PsLogList utility, 4, 241–244
PsPasswd utility, 4, 245
PsPing utility, 4, 246–247
Rammap utility, 4, 248–250
  mapped file usage statistics, 444–445
  page lists, 438–439
  purging, 437
  usage analysis, 437–446
PsService utility, 4, 245–251
config command, 248–249
depend command, 249
find command, 250
query command, 246–248
security command, 249–250
setconfig command, 251
stop, start, restart, pause, cont commands, 251
PsShutdown utility, 4, 251–254
PsSuspend utility, 4, 254
PsTools suite, 4, 219–257
  alternate credentials, 222, 227
  command-line syntax, 254–256
  common features, 220–223
  overview, 219–220
PsExec, 224–232
PsFile, 232–233
PsGetSid, 233–235
PsInfo, 235–237
PsKill, 237–238
PsList, 238–240
PsLoggedOn, 240
PsLogList, 241–244
PsPasswd, 245
PsPing, 423–432
PsService, 245–251
PsShutdown, 251–254
PsSuspend, 254
remote connections, troubleshooting, 222–223
remote operations, 220–222
system requirements, 257
public symbol files, 32
Pyle, Ned, 625
Q
Q: drive, 607–610
quota charges, 456
R
RAM. See also memory
  allocation type, 438
  mapped file usage statistics, 444–445
  page lists, 438–439
  purging, 437
  usage analysis, 437–446
Rammap utility, 6, 437–446
  File Details tab, 444–445
  File Summary tab, 444
  Physical Pages tab, 442–443
  Physical Ranges tab, 443
  Priority Summary tab, 441
  Processes tab, 440–441
  purging physical memory, 445
  snapshots, saving and loading, 446
  Use Counts tab, 438–440
reachability testing, 423
reading to alternate data streams, 391
Read-Only permissions, 500
reads, cached and noncached, 535–536
ReadyBoost driver, 517–518
rebooting, 285
  malware-related, 569–573
  in Safe Mode, 582–583
redirection
  console output, 225–226
  image hijacks, 131–132
reference counts, 456
RegDelNull, 6, 463
Regedit
  registry paths, navigating, 461–462
  running as System, 230
RegEnumKey events, 515
registry
  BCD structure, 405
  jumping to, 160
  NAME NOT FOUND errors, 485
  program associations, 484–486
  redirecting access to, 480
  user settings, 486
  Windows Boot Manager, 407–408
registry events, 148, 186–187
registry hives, 25–26
registry keys
  failed open attempts, 478–479
  misconfigured permissions, identifying, 337–339
  modifications, capturing, 618
  modifications, processing, 622–623
  with null characters, deleting, 463
  permissions, analyzing, 502
  permissions, viewing, 317
  registry usage, 446–449
  write times, 446
registry paths, 461–462
Registry Usage (RU) utility, 446–449
  CSV output, 448
  hive analysis, 448–449
  registry keys, specifying, 447
  subkeys usage, 447
RegJump utility, 6, 461–462
remote computers
  alternate credentials, 222
  debug output, monitoring, 283–284
  event logs, displaying, 241–244
  executing arbitrary processes on, 224–232
  open files, listing, 232–233
  processes, suspending, 254
  shut down, reboot, and hibernate utility, 251–254
  specifying, 221
  system information, displaying, 236
remote connections
  alternate credentials, 222, 227
  impersonation, 227
  PsExec options, 229
remote desktop services (RDS) sessions, 35–36
remote operations, PsTools for, 220–223
remote process exit codes, 225
Remote Registry service, 223
remote thread creation events, 329–330
remotely opened files, listing, 232–233
removable drives
  Autorun.inf file, 480–481
  dismounting, 409
  Write permissions, 409
resources
  access conflicts, 468
  brokered access, 25
  creating or opening, 21
  reverse DNS lookups, 434
Richards, Andrew, 538, 632
Rights Management Services (RMS) debug tracing, 491
rogue security software, 546
Winwebsec, 577–586
root cause analysis, 490
root objects, properties, 354
RootkitRevealer, 596
rootkits, 123, 141, 549
round-trip latency, 423
RU, 6
Run As Administrator command, 17, 341
Run As command, 340–341
Runas.exe command, 16–17
runaway processes, 514–516
runaway threads, 510–512, 514
running processes
  defined, 45
  information about, listing, 238–240
  memory allocations, viewing, 261
  run-once bugs, 526–528
runtime environment, PsExec command-line options, 229–232
Russinovich, Mark, 3, 15, 549
  blog, 12–13
  webcasts, 13
S
Safe Mode, rebooting in, 582–583
Safe Removal applet, 409
“sandboxing” techniques, 22
scareware, 546
  Winwebsec, 577–586
Scheduled Tasks, 115n1
Schwartz, Jon, 341
screen saver
  autostart entries, 133–134
  Bluescreen Screen Saver, 6, 463–464
screen-magnification utility, 383–387
SDDelete utility, 5, 346–349
SearchFilterHost.exe crashes, 505–507
SearchProtocolHost.exe crashes, 505–507
searching online for process names of events, 160
SearchProtocolHost.exe crashes, 505–507
Second Level Address Translation (SLAT), 454
second-chance exceptions, 497
Section handles, 296–300
sections (Windows file-mapping objects), 20, 455
secure object permissions, reporting, 314–322
secure file deletion, 346–349
security. See also malware
  App Containers, 23–28
  “buddy system” malware, 54, 572
  continuous monitoring, 588
cybersecurity, 588, 611
object reuse protection, 346
packed images, 47
"pass the hash" attacks, 28, 223
shatter attacks, 39
social-engineering attacks, 574–576
squatting attacks, 26–27
utilities for. See AccessChk utility; AccessEnum utility; Autologon utility; LogonSessions utility; SDelete utility; ShareEnum utility; ShellRunAs utility; SigCheck utility; Sysmon utility
security catalog file dumps, 302, 313
security context, 19–20
security descriptors, 321–322, 594–595
security identifiers (SIDs), 233, 476n
App Container, 23
translating to names, 233–235
Security Reference Monitor, 21
security zone information, removing, 391
semaphores, 455
Service Control Manager (SCM), 592
services. See also Windows services
  Allow Service To Interact With Desktop option, 37
  load order, 457–458
  permissions, 476–477
  in processes, 46
  security identifiers, 476n
  session 0 isolation, 36–37
Session Manager (Smss.exe), 130, 398
sessions
  interactive, 36
  private memory, 440
  remote desktop services, 35–37
  windows stations, 35
shared memory sections, 20
ShareEnum utility, 5, 339–340
sharing violations, 487–489, 602
shatter attacks, 39
ShellRunAs utility, 5, 340–342
shims, 528
Sidebar Gadgets autostarts, 136
Sieext.dll Microsoft internal debugger extension, 540
SigCheck utility, 5, 118, 121, 302–313
  command-line syntax, 304
  driver images, verifying, 610
  files to scan, specifying, 305
  image type, verifying, 617
  MachineType line, 473
  malware detection and removal features, 548
  output format, 312
  signature verification, 306–308
  suspicious executable files, analyzing, 577
  VirusTotal analysis, 308–310
signature verification, 99, 302, 306–308, 531–532
sluggish performance, troubleshooting, 505–507, 510–511
SMB share permissions, viewing, 317–318
snapshots
  of Active Directory databases, 351–352, 358–360
  LiveKd output, 288–289
  saving and loading, 437, 446
  VMMap utility, 266–268, 273–274
social-engineering attacks, 574–576
sockets, processors mapped to, 454
soft links, 392
software. See also applications
  Autoplay installation option, 477
  autostarts, 113–142
software exceptions, 496–497
solid state drives
defragmentation, 414
  as ReadyBoost cache, 517–518
Solomon, David A., 15, 45, 145
Soukami, Tarik, 498
sparse files
  fragmentation of, 416
  secure deletion, 348
Spy++, 510–511
SQL Server databases, saving BgInfo data to, 380
squatting attacks, 26–27
Ssonsvr.exe startup, 486–490
stack traces, 187–188. See also call stacks
  for root cause analysis, 490
  third-party drivers, 536
standard user, 39
standby lists
  emptying, 445
  RAM on, 441
Start menu, launching SysInternals utilities from, 7–8
startup processes, displaying, 51–52. See also autostarts
state-sponsored cyber warfare, 549
stations, Windows, 37, 455
StockViewer, 529
Streams utility, 5, 9, 391–392
strings, 389
  in executable files, 581–582
  in memory regions, viewing, 268–269
  null characters, 463
  saving to text files, 79, 92
  searching for, 79, 92, 389–390
Strings utility, 5, 389–390
  malware behavior, revealing, 601
  prefetches, scanning, 489
  suspicious executable files, analyzing, 577
Stuxnet virus, 549–569
  disabling, 555
  elevation of privilege on Windows 7, 566–568
  filtering events, 555–558
  infection vector, 550
  .PNF files, 563–566
  system modifications, 558–563
  on Windows XP, 550–554
 Svchost.exe processes, 501–502
 Symantec Enterprise Vault, 507
symbol files, 31–34
  creation, 32
  for kernel memory dumps, 292
  none installed, 483
  troubleshooting loading issues, 291
symbol servers, 32
symbolic links, 392–393, 455
Sync utility, 6, 408–410
Sysinternals Live, 10
Sysinternals Site Discussion blog, 12
Sysinternals utilities, 7
administrative rights, 16
downloading, 7–9
going started, 3–14
launching, 7–8
license information, 13–14
malware blocking, 596–598
malware detection and removal features, 548–550
Mark's blog, 12–13
overview, 3–6
running from web, 10
single executable image, 11
32-bit and 64-bit support, 11
Windows SysInternals forums, 11–12
Windows SysInternals site blog, 12
Sysmon utility, 5, 323–337
eroort reports, 331
event data, extracting, 336–337
events recorded, 323–331
installing and configuring, 331–335
logging of events, 323
service state change events, 330
system behavior, understanding, 607–629
information, displaying, 235–237
processes, displaying, 51
PTE memory, 440
shut down, reboot, and hibernate utility, 251–254
System account, 224, 229–230
system clock, 459
System Configuration Utility (msconfig.exe), 113–114
System Idle Process, 51
DLL view, 74
processors, enumerating, 98
system information utilities, 6, 437–459
ClockRes, 459
CoreInfo, 449–454
LoadOrder, 6, 457–458
PipeList, 458–459
RAMMap utility, 437–446
Registry Usage (RU), 446–449
WinObj, 454–457
System process, 20, 51, 74, 517–518
systemwide metrics. See also system information utilities
commit charge, 103–104
CPU usage, 103
displaying, 71–72
GPU usage, 106–107
I/O, 105
memory usage, 103–105
in Procesx, 102–107
summary statistics, 102

TCP, 424n2
bandwidth testing, 423–424, 429–431
connections, closing, 433–434
endpoints, listing, 433–434
latency testing, 423–429
per-process operations, 66–67
TCP Ping, 425–427
TCPView, 6, 433–434
connection requests, listing, 611
Resolve Addresses option, 433
Show Unconnected Endpoints option, 433
terminal services (TS) sessions, 35–36, 57, 61, 153, 600
terminating processes, 237–238
text, extracting, 389–390. See also strings
text files, unreadable, 482–483
third-party drivers in stack trace, 536
thread IDs (TIDs), 20
thread-local storage (TLS), 20
threads, 19–20
access tokens, 20
active, identifying, 484
call stack dumps, 540
call stacks, 19, 98, 156–158, 511–516
components, 19–20
context switches, 44, 97
CPU cycles, 97
desktops, association with, 38
GUI, 39
IDs, 20, 96
information, displaying, 96–99
killing, 99
locks, 539–540
processor state information, 19
profiling events, displaying, 158–159
remote creation events, 329–330
runaway, 510
security context, 20
services associated with, 97
shared memory sections, 20
start address, 97
suspend counts, 254
suspending, 99
system service calls, 21
thread-local storage, 20
user-mode and kernel-mode execution, 21
thumb drives, malware on, 480–481
timers, viewing information about, 455
timestamps, displaying as desktop wallpaper, 375
tombstoned objects, restoring, 371
traces
analyzing with PowerShell script, 617–625
event. See Process Monitor (Procmon)
Trojan horses, 574–576
troubleshooting
ACCESS DENIED events, 482–483
crashes, 495–507
devlopers, 631–636
error messages, 468–494
errors. See errors
exceptions, 497
hangs, 510–511
locked folders, 469–470
malware, 546–548. See also malware
sluggish performance, 505–507, 510–511
unbootable computers, 498–499
unresponsiveness, 510–511

U
UDP (User Datagram Protocol), 424n2
bandwidth testing, 423–424, 429–431
equipment, listing, 433–434
latency testing, 423–424, 428–429
unbootable computers, troubleshooting, 498–499
unhandled exceptions, 202, 206, 497
Unicode strings, searching files for, 389–390
Universal Windows Platform (UWP)
dump files, 200–201
processes, 46
unknown error explanations, 472–473
unnamed file mappings, 77
unnamed objects handles, 82
unresponsiveness, troubleshooting, 510–511
unused memory, 440
uPNSuffixes values, 493–494
User Account Control (UAC), 16–17, 223
Administrators logon sessions, 346
disabling, 18
elevation modes, 17–18
triggering, 17
user accounts
password setting utility, 245
rights, 16
User Datagram Protocol. See UDP (User Datagram Protocol)
User Interface Privilege Isolation (UIPI), 39–40
user mode, 20–21
user processes, 20–21, 52–53
user profiles, 486
AppDataRoaming folder syncs, 534
load errors, 486–490
user rights, displaying, 314–322
User32.dll, infected, 603–604
users
administrative control, 16
logged-on, listing, 240
password-setting utility, 245
Write permissions, 409

V
Veghte, Bill, 528
VHD images, 401–408
.VHDX file format, 402
virtual addresses, displaying, 442
virtual cluster numbers (VCNs), 415–416
virtual desktops utility, 382–383
virtual hard disks (VHDs)
attaching to, 402
booting in Virtual PC, 402
creating, 401–402
disk signature, 403
using, 402
virtual machines (VMs)
debugging, 285, 290
physical disk representation to, 401–408
virtual memory. See also memory
access modes of processor, 20
addresses, 437–438
analysis, 259–274
Virtual PC, 401, 403
Virtual Server, 401
virtualization, processor, 454
virus scanners, 537
VirusTotal analysis
autostart files, 119–120
process image files, 55, 100–101
Procexp, 575
SigCheck utility, 308–310
VirusTotal.com web service, 55, 100–101, 119
VMMMap utility, 4, 259–274, 438
address space fragmentation, 272–273
command-line options, 274
defaults, restoring, 274
graphical analysis window, 262–264
instrumented processes, information from, 261–262, 269–272
launching applications from, 261–262
malware detection and removal features, 548
memory information, 265–266
memory types, 264–265
printable strings, 553
process address space usage display, 553
processes, choosing, 260–262
scareware monitoring and analysis, 577–586
snapshots, 266–268, 273–274
strings/text, 268–269
timeline feature, 267–268, 632, 634–635
unknown stack addresses with write and execution permissions, 559
writable and executable pages, 581
volume management utilities, 401–422
Contig, 413–418
Disk2Vhd, 401–408
DiskExt, 418–419
DiskView, 410–413
LDMDump, 419–421
Sync, 408–410
VolumeID, 421–422
Volume Serial Number, 421–422
Volume Shadow Copy Support (VSS), 401
VolumeID utility, 6, 421–422
volumes
disk statistics, 413
free space, 416–417
graphical map, 410–413
ID number, 421–422
multipartition, 419
partition locations, 418–419
permissions, 409–410
raw access events, 330
Windows Security Reference Monitor, 27
Windows services. See also services
autostarts, 129
configuration information, 248–249
dependencies information, 249
disabling and deleting, 129
dumps, capturing, 198
effective permissions, 593
information about, 245–251
instances, searching for, 250
misconfigured, as malware, 592–595
permissions, 93, 317, 592–593
in processes, 93
security descriptors, 594–595
security information, 249–250
start type, 251
starting, stopping, restarting, continuing, or pausing, 251
Windows Sockets (Winsock) providers, 134
windows stations, 35, 37
Windows SysInternals forums, 11–12
Windows SysInternals site blog, 12
Windows SysInternals website, 6–13
Winlogon, 602
autostart entries, 133–134
running processes on, 230–231
WinObj utility, 6, 454–457, 608–609
access rights, 454
directories, 456–457
interactive window stations, viewing, 37
Object Manager namespace, graphical view, 36
Winwebsec scareware, 577–586
Wireshark, 611
WMI event consumers, 136
WMI Provider Host process, 519
working sets, 445
worms, 480–481
Wow64, 220, 618
writing to alternate data streams, 391

XML schema of Procmon, 171–174

Zero Day (Russinovich), 549
zero page thread, 439
zero–day vulnerabilities, 549–550, 566
zeroed memory, 439
.zip files, unblocking, 8–9
Zone.Identifier stream, 391
Zoomit utility, 5, 383–387
break timer, 387
drawing mode, 385–386
LiveZoom, 387
typing mode, 386
zoom mode, 385