Rogue Processes
Malware authors generally pick one of two strategies for obscuring their malicious processes: hide in plain sight or attempt to appear legitimate, or use code injection and/or stealth methods to hide from the view of normal analysis tools. See below for more on code injection and rootkits.

When searching for malware attempting to hide in plain sight, look for process names that appear legitimate but originate from the wrong directory path or with the wrong parent process ID. Look for maliciously named svchost.exe or lsa.exe and check for unusual command-line arguments. See the opposite side of this poster for legitimate Windows process details.

Besides processes, look for suspicious DLLs executed through exported APIs that expose hidden processes. Checking for signed code can help reveal suspicious executables. While there have been and continue to be endeavors to signature malware, you can typically rely on code signed by a company you trust using a certificate from a trusted CA. For example, on a default installation of Windows 7 Enterprise, processes executed by svchost.exe are signed by Microsoft for live response memory analysis. Windows' malware defenses will check on-disk signatures for running code. For offline analysis, DFRWin Advanced Analysis Tools' System'sightcheck.exe provide a tremendous amount of information about a file's digital signature.

Code Injection and Rootkit Behavior
Code injection and rootkits provide malware by hiding it from normal analysis techniques. Fortunately, memory analysis provides an effective mechanism for detecting both of these behaviors.

Typical code injection techniques provide an effective way to hide code without requiring any low-level programming knowledge, thus making it very popular among malware authors. Code injection is almost never legitimate, as the use of such code obfuscation, and evidence of code injection on a standard system is almost always worth looking into further.

A rootkit is a broad term for describing ways of subverting the operating system with the intent to hide activity and data. There are a number of techniques for doing this, but the end result is usually malware that is often undetectable by security tools running on the system. That said, there are a few rootkit detection tools available, such as GHIK and Rootkit Revealer, that can compare the state of the system as determined by the OS versus the state determined by the tool. When there are differences, it is often an indication of rootkit behavior.

The easiest effective technique for detecting rootkits is via memory forensics since offline memory analysis does not rely on the compromised OS. For example, memory forensics can identify running processes even if they are unlinked by a malicious rootkit. Evidence of code injection and rootkits is almost always worth looking into further.

Memory Analysis tools like Mandiant Redline and Volatility provide robust techniques for finding code injection and rootkit behaviors.

Unusual Windows Behavior:

Rogue Processes
Unknown Services
Code Injection and Rootkit Behavior
Unusual OS Artifacts
Suspicous Network Activity
Evidence of Persistence

Unusual Services
Windows services are designed to run applications in the background without user interaction. Many services are required on system boot, including the DRPD client, Windows Event Log, Server, and Workstation services. These services provide critical functionality for the OS and must be started immediately without requiring user input.

Services can be exploited as standalone maliciously loaded DLLs. In order to consume less resources, every service DLL is grouped together and run under a separate shared executable instance. svchost.exe is a Windows generic service host process, and it is typical to see several running instances of svchost.exe (5 or more is common).

Service configurations, as well as device driver configurations, are stored in the registry under HKLM\SYSTEM\CurrentControlSet\Services. This key holds the parameters for each service, including the service name, display name, path to the service's executable image file, the start type, required privileges, dependencies, and more. Each service is configured to start at boot, by manual intervention, or on trigger events such as obtaining an IP address or hardware device connection. Windows services provide great flexibility to developers, and remember to evaluate services, for automatically running code on a Windows host.

For offline analysis, investigate service configurations within the registry. On live or remote systems, use the built-in ‘net.exe’ command to query installed services. Try parameters such as ‘queryres’, ‘qq’, ‘querysvc’, and ‘querytype’ to get detailed information on service configurations.

Unusual OS Artifacts
Malware does not need to be present on a system for it to be compromised. We need to also look for unusual OS-based artifacts that would not exist on a typical operational system. When looking for process execution, focus on suspect, anomalous, suspicious registry keys, and event logs. Many of these artifacts can result from an adversary using your system but not implanting malware. Look for evidence showing odd behavior such as new processes that are not normal or exhibiting normal user activity.

• cmd.exe execution – Provides command-line access
• cscript.exe or wscript.exe execution – Used for privilege escalation and persistence
• psExec.exe execution – Provides a mechanism to execute command-line access
• PowerShell execution – Provides a mechanism to execute command-line access
• net.exe execution – Used for mapping drives for lateral movement and enumerating groups like “Domain Admin”
• net.exe execution – Used for mapping drives for lateral movement and enumerating groups like “Domain Admin”
• MountPoint.exe registry key – Records shares on remote systems such as CIFS, WebDAV, etc.
• Job file in C:\Windows\Tasks – Related to add application Persistence

Suspicous Network Activity
Many suspicious processes in Windows utilize the network, including svchost.exe, lsa.exe, and even the svchost.exe process. Since you can’t rule out the possibility of legitimate network activity stemming from these processes, you need an effective way to identify legitimate network activity. Windows networking tools can go through mining and even residual connections and sockets established by the system. When you are getting started to try to identify unusual network behavior, keep an eye out for the following:

• Any process communicating over port 80, 443, or 8000 that is not a browser
• Any browser not communicating over port 80, 443, or 8000
• Connections to unassigned external or internal IP addresses. For example, why did a process have a TCP connection to a system in Moldova?
• Web requests directed to an IP address rather than a domain name
• RDP connections (port 3389), particularly if originating from an odd IP address. External RDP connections are typically routed through a VPN concentrator.
• DNS requests for unusual domain names

Evidence of Persistence
Malware commonly accomplishes persistence using a variety of techniques. The most often used capability to achieve persistence with elevated rights is through scheduled tasks using the “at” command. With elevated rights, an adversary can create a service to automatically load malware or replace an existing service with malware. Finally, malware can achieve persistence with elevated rights is through scheduled tasks using the “at” command. Malware can be used to run scripts at logon/logoff. Finally, malware can achieve persistence with elevated rights through scheduled tasks using the “at” command.

• More Advanced – PowerShell background job, Local Group Policy, MS Word Add-in, or BIOS Flashing

Poster References
• Windows Internals, 6th Edition, Parts 1 & 2
• Tool IT (Windows Internals, 6th Edition, Parts 1 & 2)
• Windows Internals Administrator’s Reference
• Windows Internals, 6th Edition, Parts 1 & 2
• More Advanced – PowerShell background job, Local Group Policy, MS Word Add-in, or BIOS Flashing

Evidence of Persistence
Malware commonly accomplishes persistence using a variety of techniques. The most often used capability to achieve persistence with elevated rights is through scheduled tasks using the “at” command. With elevated rights, an adversary can create a service to automatically load malware or replace an existing service with a new malicious executable. The next most common malware persistence mechanism is using the registry. Malware can add a new registry module and auto-start services run at boot to allow persistence. Finally, malware can also be installed as a Microsoft Office Add-In when MS Word starts, the malware is executed.

• Scheduled Tasks
• Run keys
• Service Replacement
• Service Creation
• More Advanced – PowerShell background job, Local Group Policy, MS Word Add-in, or BIOS Flashing
Know Normal...Find Evil

Knowing what’s normal on a Windows host helps cut through the noise to quickly locate potential malware.

Use the information below as a reference to know what’s normal in Windows and to focus your attention on the outliers.

---

**System**

Hacker View Tools Users Help

<table>
<thead>
<tr>
<th>Processes</th>
<th>Services</th>
<th>Network</th>
<th>Disk</th>
</tr>
</thead>
</table>

**Process Hacker**

When searching for malicious processes, look for any of these anomalous characteristics:

- Started with the wrong parent process
- Image executable is located in the wrong path
- Misprocessed packages
- Processes that are running under the wrong account (incorrect SID)
- Processes with unusual start times (i.e., starts minutes or hours after boot when it should be within seconds of boot)
- Unusual command-line arguments
- Packets executable

---

**Process Listing from Windows 7 Enterprise**

Image Path: %SystemRoot%\System32\lsass.exe

Parent Process: wininit.exe

Number of Instances: One

User Account: Local System

Start Time: Typically when user logs in

Description: The generic host process for Windows Services. It is used for running the Services.exe service, which is responsible for starting and stopping services and scheduled tasks. Services.exe also implements the Service Control Manager (SCM), which handles the loading of services and services denied by network connections. In addition, a user having a limited access account, it is the LSASS executable that handles all local security access control (LIML). A list of local services can be found in the %SystemRoot%\etc\services file.

---

**explorer.exe**

Image Path: %ProgramFiles%\Internet\Explorer\explorer.exe

Parent Process: explorer.exe

Number of Instances: One or more

User Account: Administrator

Start Time: Starts with Windows 7/8 and later

Description: Used to start Internet Explorer. It typically starts when a user logs in. Depending on the user's profile and settings, Internet Explorer may be started in the background or in the foreground.

---

**smss.exe**

Image Path: %SystemRoot%\System32\smss.exe

Parent Process: explorer.exe

Number of Instances: One

User Account: Administrator

Start Time: Starts when user logs in

Description: The Session Manager process is responsible for creating new sessions. When a user logs in, the Session Manager starts a new instance of the explorer.exe process. This process is responsible for creating and managing all sessions on the system, including the explorer.exe process, which is responsible for rendering the desktop and providing user interface functionality.

---

**explorer.exe**

Image Path: %ProgramFiles%\Internet\Explorer\explorer.exe

Parent Process: explorer.exe

Number of Instances: One or more

User Account: Administrator

Start Time: Starts when a user logs in

Description: Explorer.exe is the primary user interface for managing files and folders on a Windows system. It is typically started when a user logs in and is responsible for rendering the desktop and providing user interface functionality.

---

**csrss.exe**

Image Path: %SystemRoot%\System32\csrss.exe

Parent Process: explorer.exe

Number of Instances: One

User Account: Administrator

Start Time: Starts when a user logs in

Description: The Session Manager process is responsible for creating new sessions. When a user logs in, the Session Manager starts a new instance of the explorer.exe process. This process is responsible for creating and managing all sessions on the system, including the explorer.exe process, which is responsible for rendering the desktop and providing user interface functionality.

---

**services.exe**

Image Path: %SystemRoot%\System32\services.exe

Parent Process: explorer.exe

Number of Instances: One

User Account: Administrator

Start Time: Starts when user logs in

Description: The Service Control Manager (SCM) is responsible for controlling all of the system's services. It is typically started when a user logs in and is responsible for starting and stopping services, managing service dependencies, and providing other services to the operating system.

---

**wininit.exe**

Image Path: %SystemRoot%\System32\wininit.exe

Parent Process: explorer.exe

Number of Instances: One

User Account: Administrator

Start Time: Starts when user logs in

Description: wininit.exe is a critical process responsible for initializing the Windows environment. It is typically started when a user logs in and is responsible for loading the kernel and launching the explorer.exe process to render the desktop.

---

**explorer.exe**

Image Path: %ProgramFiles%\Internet\Explorer\explorer.exe

Parent Process: explorer.exe

Number of Instances: One or more

User Account: Administrator

Start Time: Starts when a user logs in

Description: Explorer.exe is the primary user interface for managing files and folders on a Windows system. It is typically started when a user logs in and is responsible for rendering the desktop and providing user interface functionality.

---

**lsass.exe**

Image Path: %SystemRoot%\System32\lsass.exe

Parent Process: explorer.exe

Number of Instances: One

User Account: Administrator

Start Time: Starts when user logs in

Description: The Local Security Authority (LSA) is responsible for managing security information on a Windows system. It is typically started when a user logs in and is responsible for managing user accounts, security policies, and security credentials.

---

**explorer.exe**

Image Path: %ProgramFiles%\Internet\Explorer\explorer.exe

Parent Process: explorer.exe

Number of Instances: One or more

User Account: Administrator

Start Time: Starts when a user logs in

Description: Explorer.exe is the primary user interface for managing files and folders on a Windows system. It is typically started when a user logs in and is responsible for rendering the desktop and providing user interface functionality.

---

**explorer.exe**

Image Path: %ProgramFiles%\Internet\Explorer\explorer.exe

Parent Process: explorer.exe

Number of Instances: One or more

User Account: Administrator

Start Time: Starts when a user logs in

Description: Explorer.exe is the primary user interface for managing files and folders on a Windows system. It is typically started when a user logs in and is responsible for rendering the desktop and providing user interface functionality.