Security Guide: How to Evade Anti-Virus Detection

How do you test that your defenses are actually working? Penetration testing can give you a clear view of the vulnerabilities that can easily be exploited within your environment; however, organizations need to be able to understand and test their users' behavior without anti-virus programs stopping these tests in their tracks. A great way to explore that user behavior is by deploying social engineering programs during a pen test.

In this guide, penetration testers will learn how to evade anti-virus detection on target machines for your Metasploit pen tests. This guide will be most useful to readers who already have some penetration testing experience and are familiar with Metasploit Pro.

Key Concepts: Exploits and Payloads

Attacks keep growing in complexity, and as a result, your defenses need to become stronger as well. In order to keep up with the latest attack methods, you need to know anti-virus evasion techniques in order to test user behavior.

Exploits are used to deliver a payload onto a victim's computer. There are two types of payloads: a single payload in which the exploit delivers everything at once; and a staged payload that loads the stager into memory and then connects back to Metasploit and pulls down the next stage of the payload. Metasploit created the idea of separating an exploit from its payload, and these staged payloads are most common because exploits typically need more space than available with a single payload.

Payloads consist of the actual shell code which executes instructions on the victim's machine, and the executable template. If the shell code isn't being executed directly from main memory, it is embedded in an executable (an .exe file in the case of Windows) that loads that shell code into memory and executes it. Most times, this executable template is what is actually being detected by an anti-virus program, not the actual payload.

Therefore, if the .exe file is what's being detected by the anti-virus, that's where we need to focus for evasion.

First, let's cover the obvious ways to avoid anti-virus:

- Turn off anti-virus on the victim machine before payload executes
- Suspend the anti-virus processes
- Migrate code directly into a Windows system process
- Do not write executables to disk
    - Use memory corruption exploits when you can
    - Use alternative methods such as Power Shell

If you've taken care of the obvious ways to avoid anti-virus, let's then dig deeper and discuss what you should do when you have to drop an executable on the file system. For example, a PS Exec module is used to execute remote code via SMV when you have the credentials. Anti-virus picks up PS Exec all the time, but it is picking up the executable template, not the payload. Can you then avoid using recognizable executable templates?

RAPID7

Basic Methods to Avoid Anti-Virus Detection

There are a number of different ways we can try to avoid anti-virus.

- The first and most obvious way is to just *avoid using an executable at all*. Try and execute your payload from directly in-memory through a corruption technique or use something like Power Shell. These scripting languages typically aren't flagged by any anti-virus provider.

- You can also try to *avoid signature detection*. Signature-based detection is the most common method of identifying computer viruses. A **signature** is an algorithm or hash (a number derived from a string of text) that uniquely identifies a specific virus. Signature detection can be effective against malware where samples have already been obtained and signatures have been created. However, with over 200,000 unique malware[1] created every day, it is impossible for the signature detection technology to keep up.

  In addition, signature matching is based on the executable template only, not the payload. So, when an executable payload is generated in Metasploit—even if there is nothing actually malicious in the payload—many anti-virus programs will pick it up. We believe signature detection is lazy. Basically it says "If you see an .exe that looks like this, it's a Metasploit payload and we want to kill that." That said, it's incredibly easy to beat.

  Heuristic analysis is more sophisticated than signature detection in the way that it tries to detect viruses. With heuristics, the anti-virus engine monitors the application and determines if they behavior is malicious by matching specific behavior rather than exact patterns in the code. A good heuristic-based engine teaches itself to become smarter as it learns more, but these can also be beat.

- One option is to use a *custom executable template*, preferably something that is known to be a valid .exe, such as the Windows Calculator Program. Certain anti-virus programs will take a look at that template say: "Oh, this is Calculator. I know what this is. I really don't have to worry about this." And with that, you have evaded the anti-virus program.

Recommendation: Create Your Own Executable Templates

We recommend using different executable templates or creating your own templates. To use different executable templates, select the PS Exec module in Metasploit and hit "Show Advanced" to get to the .exe template option. Alternatively, you can create your own executable template. There are several different techniques you can use in your template to help try and hide what you are doing from an anti-virus solution. Anti-virus products use a technique called sandboxing in a lot of cases. Sandboxing creates a protected part of the operating system and runs the executable in for a limited period of time to make sure that it is not doing anything malicious.

In those cases, all we have to do is beat the clock. We can continue to do harmless operations over and over again for a minute or two, usually less. The anti-virus sandbox will eventually say: "Well, there's nothing going on here, I am just going to let this program run." Therefore, we can beat out the clock, continue on, and execute our payload.

[1] http://www.bitdefender.com/blog/Anti-virus-Signature-Detection-Is-Not-Effective-Why-Would-You-Still-Use-It-61.html

Busting the Encoder Myth

There is a myth among pen testers that encoders are meant to evade anti-virus. To be clear: encoders are not meant to evade anti-virus, but are designed to handle bad characters—they are only involved with anti-virus on an accidental basis.

If there's a null byte anywhere in the string, that will break the code. Encoders have rules for altering the payload shell code to get rid of any bad characters; but they were not designed to evade anti-virus.

With memory corruption exploits, some characters may break your exploit, such as space, tab, carriage return, and line feed. Advanced encoders use a rolling XOR encoding to create polymorphic code. The prepended decoder stub decodes it, in other words, the payload is not recognizable until decoded. As a result, this has unintentionally evaded the anti-virus program, but that was not its intention.

Therefore, encoders are not going to get you past anti-virus most of the time; messing with the executable templates will get you much further.

Why and How to Use the Dynamic Executable Template Generator

The purpose of the *dynamic executable generator* is to avoid anti-virus detection by generating the C code for the executable template. It includes the payload directly, instead of injecting it after the template has been generated. The C code is written in a random, on-the-fly fashion and compiled with Metasm, using several techniques to try to evade anti-virus programs.

Metasploit Pro offers the option to dynamically generate executable templates when using the PS Exec module. Select from "Advanced Options" when running the PS Exec module, or from the "Payload Settings" in the Bruteforcer when selecting SMB.

Advantages of the dynamic executable generator include the fact that the code is different every time, it appears harmless, the payload is hidden, and the code can detect debuggers. Since the code is different every time it is generated, the execution flow doesn't fall into any predictable patterns. This makes it virtually impossible for anti-virus vendors to create a static signature to detect these payloads. The randomly generated C functions fill the call tree with harmless operations, which is important for sandboxing. Since sandboxes only run for a limited time, they will eventually release the payload.

In addition, Metasploit Pro randomizes its payload in memory. The payload is read into memory completely scrambled, and unscrambled only just before it is executed. A lot of anti-virus solutions can be picked up the same way you can pick up a debugger. So, if you detect whether a debugger is attached to the executing program, you just don't ever go to the routines that actually execute our payload. Instead, you just continue to do harmless things and anti-virus will never flag it.

Dynamic run-time linking is another strategy to evade anti-virus. Because anti-virus programs often look at the imports table of the executable, Metasploit Pro evades detection through dynamic runtime linking of all required functions, keeping the imports table almost empty. This is another layer of obfuscation and makes life for anti-virus vendors very difficult.

Conclusion

Anti-virus evasion is never guaranteed. It is an ongoing challenge, especially as anti-virus vendors continue to catch up with exploits—but it is still possible to evade anti-virus. Evading anti-virus remains an important part of penetration testing practices, as you cannot be complacent in security practices and think that anti-virus programs will stop every malicious threat that comes your users' way. Testing user behavior and training users to handle potential threats correctly is a critical part of a robust security program.

Additional Resources

Webinar

http://information.rapid7.com/evading-anti-virus-detection-with-metasploit-video-page.html

Whiteboard Wednesday video

http://www.rapid7.com/resources/videos/evading-anti-virus-detection-with-metasploit.jsp