Adventures in Shellcode Obfuscation! Part 3: Encryption

redsiege.com/blog/2024/07/adventures-in-shellcode-obfuscation-part-3-encryption

By Red Siege | July 1, 2024

By Mike Saunders, Principal Security Consultant

Watch Video At: https://youtu.be/QhMUsMf2c6g

This blog is the third in a series of blogs on obfuscation techniques for hiding shellcode. You can find the rest of the series here. If you'd like to try these techniques out on your own, you can find the code we'll be using on the Red Siege GitHub. Let's look at some encryption methods we can use to hide our shellcode.

XOR Cipher

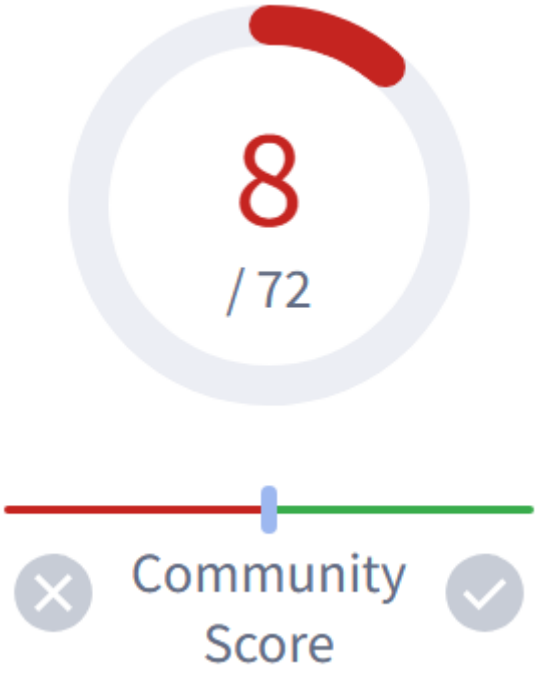
One of the most basic encryption methods we could use is a XOR cipher with a single byte key. The code is very simple:

```
for (int idx = 0; idx < sizeof(shellcode); idx++) {

shellcode[idx] = shellcode[idx] ^ xorkey;

}
```

XOR ciphers are fast but not very secure, but we're not trying to prevent our shellcode from being recovered. We're just trying to hide it from prying eyes. If we look at VirusTotal, we see that it was detected by 8 engines. Not bad, but could we do better?

A more effective method would be to use XOR with a multibyte key. That code is a little more complex.

```
void XOR(char * ciphertext, size_t
ciphertext_len, char * key, size_t key_len) {

int myByte = 0;

int k_minus_one = key_len - 1;

for (int idx = 0; idx < ciphertext_len; idx++) {

if (myByte == k_minus_one)

{

myByte = 0;

}

ciphertext[idx] = ciphertext[idx] ^
key[myByte];

myByte++;

}

}
```



VT Detection for XOR

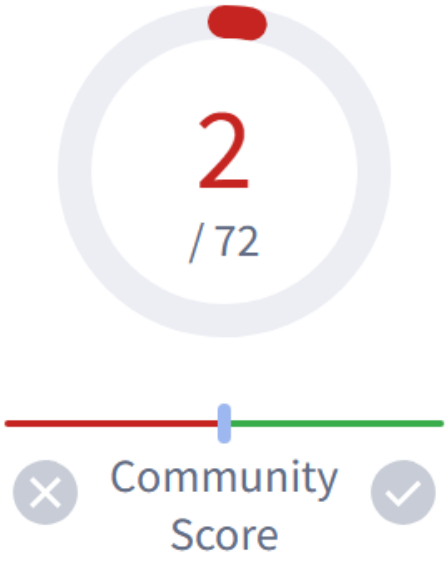
There's only one problem here – if you're target is using Defender, Defender has signatures for a XOR function with a multibyte key. Defender seems to think XOR with a multibyte key is a pretty good indication of trying to hide something. In the past, it was possible to get around this by disabling compiler optimization and not passing /Ox to cl.exe. Unfortunately, this stopped working recently. As a result, the XOR routine gets detected. Unless you can change the signature somehow. One way to do this would be to write to the null device at some point during the XOR routine:

```
// write something to the null device

FILE* outfile = fopen("nul", "w");

fputs("out", outfile);

fclose(outfile);
```

By making this small change to our XOR routine, we've changed the signature enough and Defender no longer detects our loader.

```
C:\Users\Mike\Desktop\ThreatCheck>ThreatCheck.exe -e defender
-f ..\obfuscation\XOR\xor-multibyte-key.exe
[+] No threat found!
```

Checking with VirusTotal, we see that only 2 vendors detected the program. Not too shabby!



VT Detection for XOR Multibyte Key

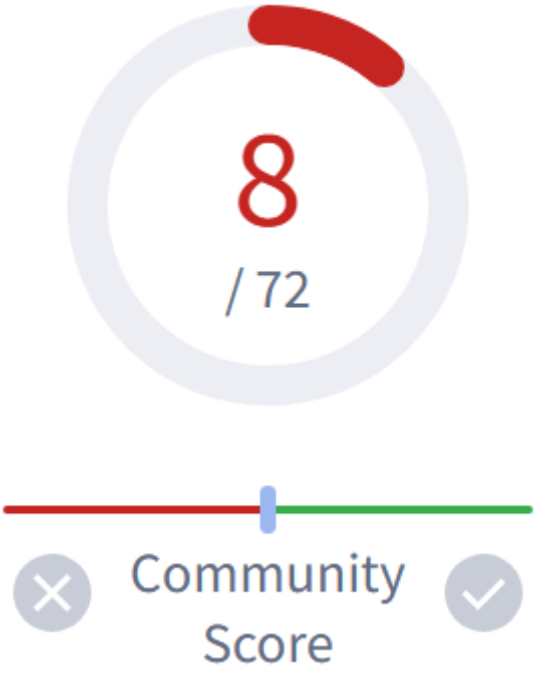
AES Encryption

Another common encryption scheme used to hide shellcode is AES. AES is a much more secure encryption algorithm than XOR, even XOR with a multibyte key. As such, we would expect that by encrypting our shellcode using AES, the detection rate should be lower. However, VirusTotal showed this version of the test program was detected by 8 engines!

Out of curiosity, I generated a file containing 32 null bytes, encrypted it, and replaced the encrypted shellcode and key in the test program. VirusTotal still showed 8 engines reported the program as malicious.

A possible reason is entropy. Entropy is the measure of randomness in something, like our encrypted shellcode program. Entropy increases when we use encryption. If you want to know more about entropy, how it can lead to detection, and how to get around it, read this blog.

The test program doesn't do much at all. It has about 600 bytes of encrypted shellcode, the decryption routine, and some code to print the decrypted shellcode out to screen. With this little actual code, the imported crypto libraries and functions, and the entropy, there are good reasons for engines to consider this potentially suspicious.



VT Detection for AES

Wrapping Up

XOR and AES certainly aren't the only means of encrypting our shellcode. Bcrypt and RC4 are other popular alternatives. If you do enough research, you'll find several other encryption routines commonly used by red teams and threat actors alike. In the next blog, we'll look at flipping things around to evade detection.

Try it Yourself

You can find the example code for this article as well as the other articles in this series at the Red Siege GitHub.

Stay Tuned

This blog is part of a larger series on obfuscation techniques. Stay tuned for our next installment!

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