Adventures in Shellcode Obfuscation! Part 6: Two Array Method

redsiege.com/blog/2024/07/adventures-in-shellcode-obfuscation-part-6-two-array-method

by Mike Saunders, Principal Security Consultant

Watch Video At: https://youtu.be/50r93mxjFMk

This blog is the sixth 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 another we can use to hide our shellcode.

The Idea

I first heard of this technique from my coworker, Justin Palk. He proposed taking an array of shellcode and splitting it into two arrays based on a byte's position in the array – even or odd. Let's take a look at what that would look like. First, we'll take a look at our original shellcode.

Note: this is just a contrived example, not functional shellcode.

char shellcode[10] = { 0xfc, 0x48, 0x83, 0xe4, 0xf0, 0xe8, 0xcc, 0x85, 0x93, 0x52 };

We'll start at index 0 in the array – `0xfc`. We'll consider position 0 to be even and insert this byte into a new array – `even`. The second byte, at position 1 – `0x48` – is in an odd position and goes into the `odd` array. We continue to iterate through our entire array and building out the new arrays until we've gone through all the shellcode. For our example, that would look like the following:

```
char even[5] = { 0xfc, 0x83, 0xf9, 0xcc, 0x93 };

char odd[5] = { 0x48, 0xe4, 0xe8, 0x85, 0x52 };
```

The above arrays are what is stored in the executable. We have taken alternating bytes of our shellcode and put them in two separate arrays; 1st, 3rd, 5th, … in one, and 2nd, 4th, 6th, … in another.

To reconstruct the shellcode, we'll use a `while` loop. First, we declare a new array to store our reconstructed shellcode. Next, we'll declare two integers. `idx` will be used to keep track of our position in the shellcode array. `twoArrIdx` will keep track of where we're at in our `evens` and `odds` arrays.

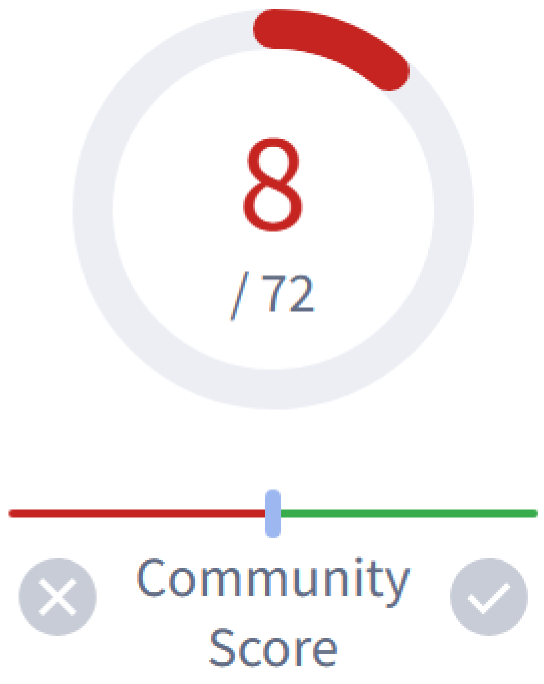
We'll use a while loop to reconstruct our payload. While `idx` is less than the length of our reconstructed shellcode, we'll read from the `evens` array and insert that value into the shellcode array at position `idx`.

If the size of our original shellcode is an odd value, then the odds array will be one less than the evens array. We'll do a test to see if our current `twoArrIdx` value is equal to that of the size of the `evens` array. If it is, it means we've read all of the values from the `odds` array and there's only one remaining `evens` value, so we'll do nothing. If it's not, we'll grab the next value from our `odds` array and insert it into the shellcode array at `idx + 1`. We have to add one to `idx`, otherwise we'd overwrite the value we just inserted from the `evens` array.

Finally, we increment `idx` by two. We do this because we've just inserted two values into `shellcode`. We need to shift two positions over, otherwise we'd overwrite the last byte we just inserted.

```
char shellcode[PAYLOAD_SIZE] = { 0x00 };

int twoArrIdx = 0;

int idx = 0;

while (idx < PAYLOAD_SIZE)

{

// read from the even array

shellcode[idx] = evens[twoArrIdx];

// odds will be one byte less than evens if PAYLOAD_SIZE is odd

if ( twoArrIdx == (int)sizeof(odds) )
{

// do nothing, otherwise we'll read past the end of our array

}

else

{

// read from odd array

shellcode[idx+1] = odds[twoArrIdx];

// increment twoArrIdx to move to the next position in the evens and odds arrays

twoArrIdx++;

}

// we've just added two bytes, so we need to shift two positions instead of one

idx = idx + 2;

}
```

Uploading the compiled test program to VirusTotal showed 8 engines alerted on the payload. Most notably, Microsoft Defender identified this as `Trojan:Win64/Meterpreter/D`, so it definitely wasn't fooled. However, this might be a useful technique on targets that aren't running Defender.



Virus Total Score

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

Advenutures in Shellcode Obfuscation! Part 5: Base64

By Red Siege | July 18, 2024

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

Learn More
Advenutures in Shellcode Obfuscation! Part 5: Base64