Binary Code Modification
[Patching Vulnerabilities]

Hellcode Research

Celil ÜNÜVER

celilunuver[n0sp4m]gmail.com

http://tcc.hellcode.net/musashi

Introduction

Vulnerabilities which are published everyday in Bugtraq can be software that we use daily. The most famous softwares can have vulnerabilities too. If you look at the bugtraq, you can see the security advisories for big vendor's products.

Recently, web vulnerabilities have been famous; nevertheless, software vulnerabilities like Buffer overflow are still the most dangerous programming errors. Also they cause the most professional attacks. Fundamentally these dangerous programming mistakes are emerging as a result of negligence of programmers.

In this paper, I will explain to patch a software which have a vulnerability.

Patching ??

This technique also known as **“hotpatching” or “runtime patching”** . Patching is a method to modify a software's binary for an aim with the help of disassembler , debugger , hex editor etc.

This is a very common technique for reversing. It can be used for api hooking , cracking, code injecting etc. But in my paper , I will explain to use this technique for fixing security holes.

Tool Bag

The softwares which are well-known and used by all reverse engineers (debuggers, disassemblers etc.) will be helpful for our work. But a disassembler/debugger which have inline assembler and binary edit features like IDA Pro, Ollydbg can make it easy.

Unfortunately , inline assembler and binary edit features are only available for “x86” executables in IDA or Ollydbg. For example , if you want to patch an arm, xbox executable , you should look at opcode (instruction encoding) topics in processor references.

Practice

We will do “hotpatching” for the program which is below. As you see , it has a buffer overflow vulnerability. Now compile it and forget the source code , it is our closed source software anymore....

```
#include <stdio.h>

int main()
  {
  char buf[16];
  printf("\nString giriniz:");
  scanf("%s", &buf);
  return 0;
  }
```

*I suppose you know what is buffer overflow etc..

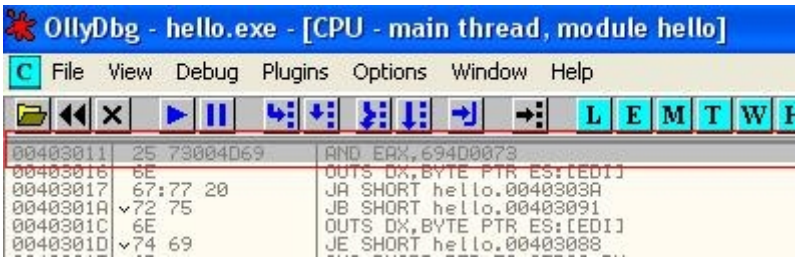
```
mov     [ebp+var_1C], eax
mov     eax, [ebp+var_1C]
call    sub_401AC0
call    sub_401770
mov     [esp+38h+var_38], offset aStringGiriniz ; "\nString giriniz:"
call    printf
lea     eax, [ebp+var_18]
mov     [esp+38h+var_34], eax
mov     [esp+38h+var_38], offset aS ; "%s"
call    scanf
```

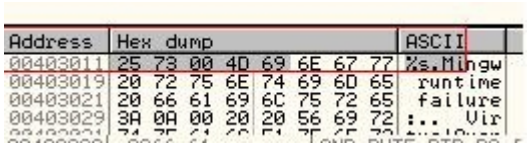
As you see in source code and assembly codes , scanf doesnt check the size of string. (%s)
As you know the functions like scanf and sprintf can check the size of strings. Just we need to put an integer in front of the format character. (e.g %15s or %.15s)
Lets try to fix this issue. I will prefer to use Ollydbg for patching. It is easy for patching. I prefer IDA for only analysing.

Opening the vulnerable program via Ollydbg;

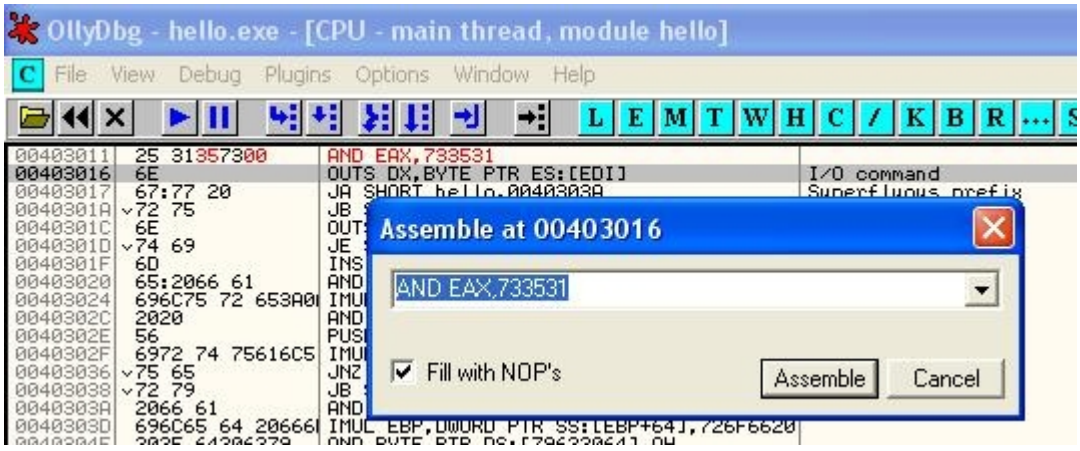
As you see in Disassembler , it call “00403011” offset for moving the string to the buffer.

Lets go to “00403011” address via CTRL+G shortcut.





I think everything is clear in the pictures :]
So we will assemble/modify this line --> “AND EAX, 694D0073” .



We are writing “and eax, 733531” instead of “and eax , 694D0073” code. (you know 313573 codes are the hex type of “%15s”. We wrote it inversely because Last In First Out!)

Yeah , thats all! We patched the vulnerable part of our software easily. For saving patched software , right click on the patched line and select “Copy to Executable > Selection” . A new window will be opened , while closing it , it will ask a question for saving or not. After saving it , you can try to overflow it :)

Lets look at our patched program via IDA PRO ;

```
mov     [esp+38h+var_34], eax
mov     [esp+38h+var_38], offset a15s ; "%15s"
call    scanf
mov     eax, 0
leave
retn
```

```
a15s db '%15s',0
```

Last words:

I hope this paper will be helpful to understand basics of patching. I am planning to write about another patching tricks in my next paper.

Acknowledgments:

I would like to thank my brother, my gf , my family, my friends [murderkey, AhmetBSD aka L4M3R , BoB (ulaş), kurti]

Links:

http://tcc.hellcode.net
http://hellcoderesearch.wordpress.com