Part 2: Linux Format String Exploitation

Welcome to the first part in our Linux Exploitation Tutorial Series. Again I would like to thank kyuzo for taking the time to share his knowledge with us! In this part we will be looking at Format String Exploitation. Format string vulnerabilities usually occur when a programmer wants to print out a user controlled function but does not sanitize the user input allowing the malicious attacker to inject their own format specifiers. This in turn allows the malicious attacker to read and write arbitrary memory.

In my own attempt to educate myself in the fundamentals of format string exploitation I read two most excellent resources: (1) Exploiting Format String Vulnerabilities [scut / Team Teso – 2001] and (2) Advances in format string exploitation [gera & riq / Phrack – 2002]. I have included links to both resources below and I can highly recommend them for background reading.

Before we get to the good stuff I just want to mention that by default gdb unassembles opcode in AT&T syntax, for those of us who have done allot of Windows exploit development this is a bit confusing. Fortunately you can easily change the disassembly flavor in gdb with the following commands..

set disassembly-flavor intel
set disassembly-flavor att

Resources:
Exploiting Format String Vulnerabilities (by scut): Link
Advances in format string exploitation (by gera & riq): Link

Introduction

A few months ago b33f and myself put together a workshop on software exploitation to be presented in a university environment. The workshop had two main ideas: (1) deal with non-buffer overflow exploitation vulnerabilities and (2) talk about both Windows and Linux. So I picked up a list of less glamorous and more esoteric(!) vulnerabilities; among these we decided to include format strings, as they have been quite important in the last few years as far as Linux is concerned. Just to mention one, at the beginning of 2012 sudo was released with a format string flaw in the sudo_debug function and that was shipped with main-stream distributions like Fedora and OpenSUSE.

After the workshop, I had promised b33f I would contribute some material to FuzzySecurity and, a few months later, I finally made up my mind and decided to make a couple of videos about format string exploitation.

Sample code used in the first part of the video tutorial:

```
/* example.c
 *
 * $ gcc -o example example.c
 * $ execstack -s example # make stack executable
 */
#include <stdio.h>

int main() {
    int a = -5;
    float b = 5.5;
    char *c = "My String";

    printf("A = %d, B = %f, C = %s\n", a, b, c);
}
```

Sample code used in the first and second part of the video tutorial:

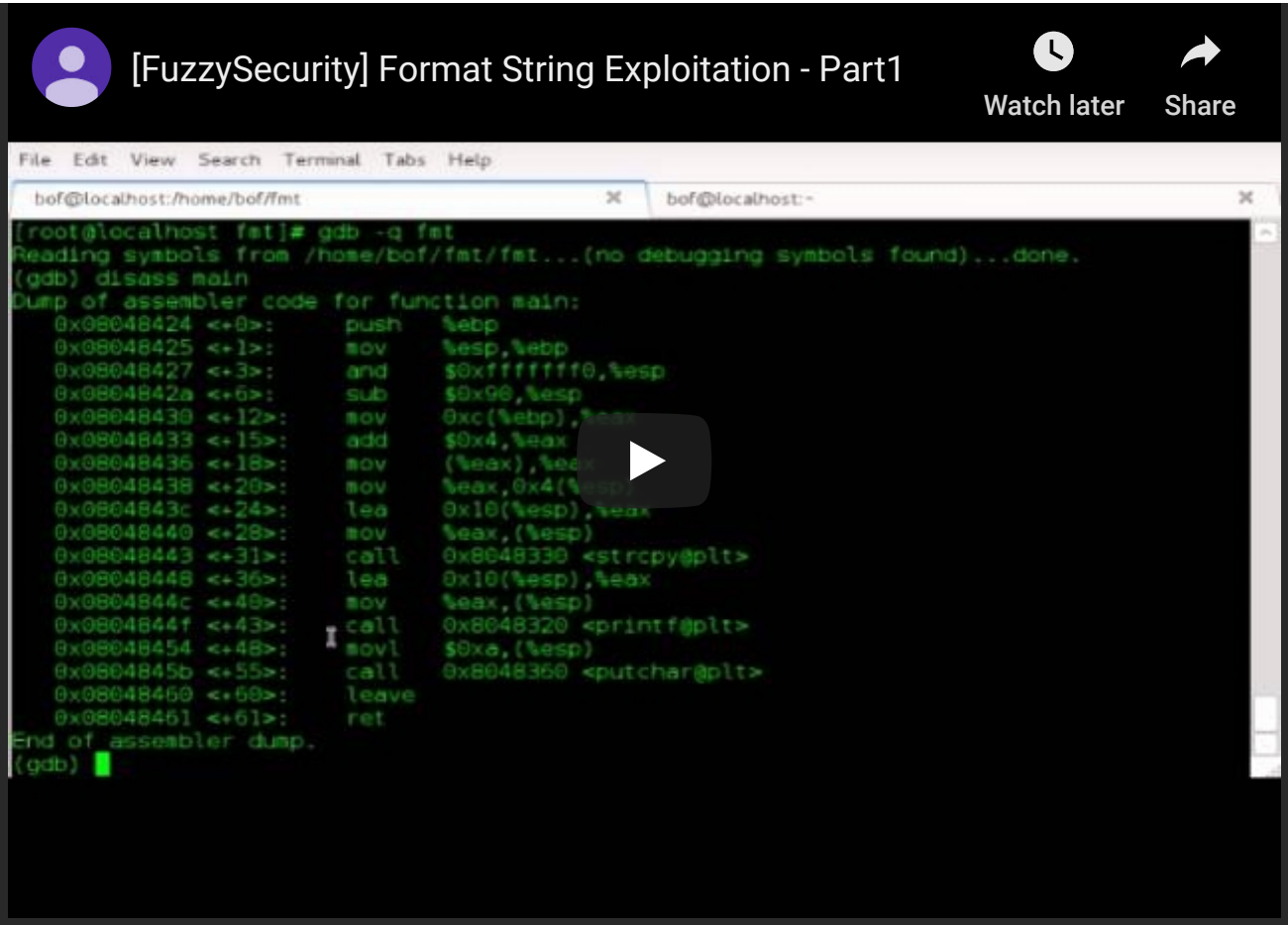
```
/* fmt.c - sample program vulnerable to format string exploitation
 *
 * $ gcc -o fmt fmt.c
 * $ execstack -s fmt # make stack executable
 */
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    char b[128];
    strcpy(b, argv[1]);
    printf(b);
    printf("\n");
```

```
}
```

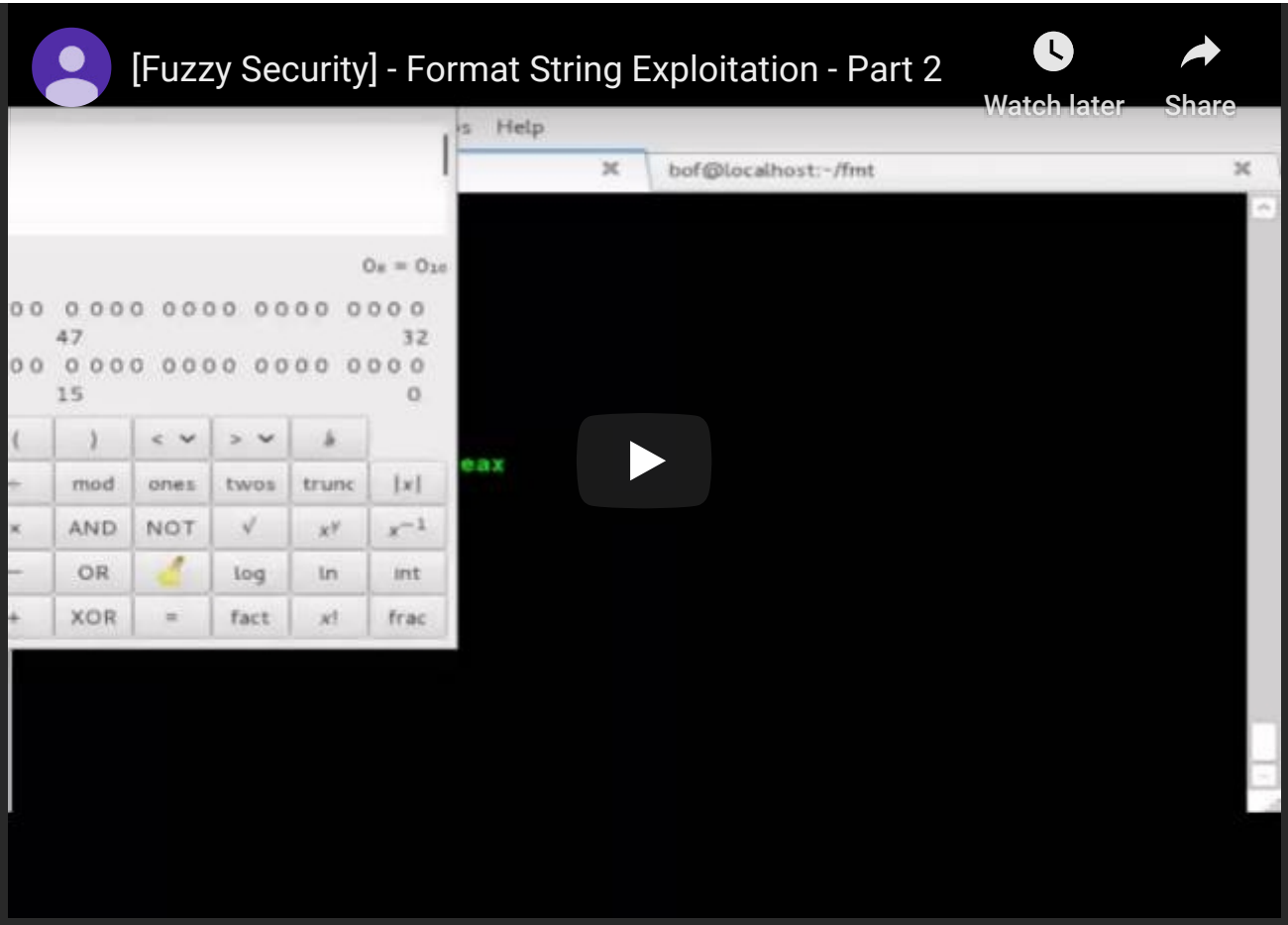
Format String Tutorial

The first video offers an introduction to what format strings are and how they can lead to information leakages (some of the topics include: conversion specifiers usage and direct parameter access). The second part moves things a step forward and shows how to own a program leveraging arcane format string features like the evil %n conversion specifier!

Part 1



Part 2



Comments

There are no comments posted yet. Be the first one!

Post a new comment

Enter text right here!

Name

Displayed next to your comments.

Email

Not displayed publicly.

Subscribe to None

Submit Comment