Smashing the Stack For Fun and Profit (Today)

18 Apr 2016

Tags: security (/tag/security/), tutorial (/tag/tutorial/)

The article Smashing the Stack for Fun and Profit (http://phrack.org/issues/49/14.html) by Aleph One is the seminal work in bringing the method of stack-based buffer overflows to the masses. However, a problem with *Smashing the Stack* is that it was published in 1996—modern defenses (which are enabled by default) frustrate would be hackers who try to follow the tutorial, only to find that the examples do not work.

As the 20th anniversary of *Smashing the Stack* approaches, this blog post tries to show how the sample code must be compiled so that the resulting executables are 32-bit and vulnerable as described.

This guide assumes you are running *Ubuntu 14.04* or later. The directions would be similar if you are running a Debian or Ubuntu derivative (https://wiki.debian.org/Derivatives). If you are not running a Linux-based system already, you can install an Ubuntu 14.04 in a virtual machine with Virtualbox (https://www.virtualbox.org/wiki/Downloads).

Install necessary packages

If you are running 64-bit (amd64) Ubuntu 14.04 or later, then you need add the necessary 32-bit libraries:

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 \
    g++-multilib build-essential gdb
```

If you are running 32-bit (x86) Ubuntu 14.04 or later:

```
sudo apt-get update
sudo apt-get install build-essential gdb
```

Compiling the example code

In order to compile the example, code you need to disable the security features that have been enabled since *Smashing the Stack* has been written. Here's an example of how to compile

example1.c:

```
gcc -m32 -fno-stack-protector -z execstack -D_FORTIFY_SOURCE=0 \
    -o example1 example1.c
```

This is what the different compile switches do:

- `-m32` : compile for 32-bit
- `-fno-stack-protector` : disable stack canaries (https://en.wikipedia.org/wiki/Buffer_overflow_protection#Canaries)
- `-z execstack` : ensure the stack is executable (disable NX bit protection (https://en.wikipedia.org/wiki/NX_bit))
- `-D_FORTIFY_SOURCE=0` : disable FORTIFY_SOURCE (https://securityblog.redhat.com/2014/03/26/fortify-and-you/)

Running programs

Before running programs you need to disable ASLR (https://en.wikipedia.org/wiki/Address_space_layout_randomization) for your system so that addresses are predictable.

To disable ASLR:

```
sudo sysctl -w kernel.randomize_va_space=0
```

To re-enable ASLR:

```
sudo sysctl -w kernel.randomize_va_space=2
```

Reading Smashing the Stack

You are now ready to read *Smashing the Stack*. Because of some minor mistakes in the original article, I recommend you read a revised version in the links below.

Links

- Smashing the Stack for Fun & Profit : Revived (https://avicoder.me/2016/02/01/smashsatck-revived/), formatted/revised by avicoder (https://twitter.com/avicoder)
- Smashing The Stack For Fun And Profit (Revised PDF version) (https://www.eecs.umich.edu/courses/eecs588/static/stack_smashing.pdf), based on HTML version by Prabhaker Mateti
- Smashing The Stack For Fun And Profit (http://phrack.org/issues/49/14.html) (Original Phrack article)

Tags: security (/tag/security/), tutorial (/tag/tutorial/)