[The Art of 0-Day Vulnerabilities, Part2: Manually Fuzzing](https://null-byte.wonderhowto.com/how-to/art-0-day-vulnerabilities-part2-manually-fuzzing-0165705/)

* BY [**MR\_NAKUP3NDA**](https://creator.wonderhowto.com/q403_access_denied/)
* 10/26/2015 3:14 PM

welcome back my masters , teachers and classmates , in this second part of this 0-day series we will learn the basic of fuzzing, and i bought some gift for nullbyte(a 0 day vuln that will be released today exclusively on null-byte) .

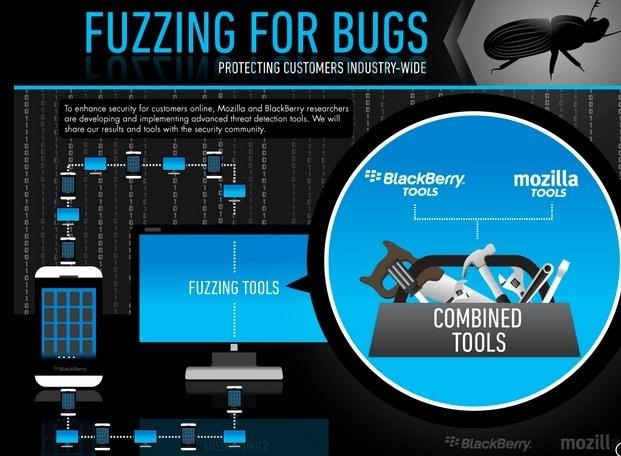
INTRODUCTION

[](https://img.wonderhowto.com/img/original/20/26/63581453641196/0/635814536411962026.jpg)

As i love wikipedia so much according to them Fuzz testing or fuzzing is a software testing technique, often automated or semi-automated, that involves providing invalid, unexpected, or random data to the inputs of a computer program. The program is then monitored for exceptions such as crashes, or failing built-in code assertions or for finding potential memory leaks. Fuzzing is commonly used to test for security problems in software or computer systems. It is a form of random testing which has been used for testing hardware or software.

Fuzzers work best for problems that can cause a program to crash, such as buffer overflow, cross-site scripting, denial of service attacks, format bugs and SQL injection. These schemes are often used by malicious hackers intent on wreaking the greatest possible amount of havoc in the least possible time.

In my opinion Fuzzing is the first step to exploit unknown vulnerabilities in software. The more you get familiar with Fuzzing, the more bugs you will find

[](https://img.wonderhowto.com/img/original/68/23/63581453213383/0/635814532133836823.jpg)

REMOTE AND LOCAL FUZZING

Depending on if the target is local or remote you will use different tools to fuzz. most of the time remote security bugs are very important, It means that to exploit the target machine we won't need user interaction, just sent the exploit to the remote machine and it will be attacked.

Local fuzzing deals with fuzzing applications locally or hosted on the target system. This can include, but isn't limited to:

Command Line Fuzzing - Fuzzing applications via the command line and/or environmental variables  
File Format Fuzzing - Fuzzing applications that read files in a specific format or format(s)  
Kernel Fuzzing - Fuzzing core kernel features, kernel modules, and  
system calls

Remote fuzzing deals with fuzzing a target remotely or a the network. This can include, but isn't limited to:

Network Protocol Fuzzing - Fuzzing applications or even a kernel that implements a specific protocol  
Database Fuzzing - Fuzzing database modules and/or database input sanitation policies  
Web Application Fuzzing - Fuzzing input vectors of web applications hosted on a web server

if a target has input, it can probably be fuzzed

LOCAL FUZZING EXAMPLE

There are a bunch of tools which you can use to fuzz locally depending on the target's environment  
you can use the command:  
now as example we will use a simple c program that i wrote that is vulnerable to buffer overflow here is the code:

#include <stdio.h>  
#include <string.h>

int main(void)  
{  
char buff15;  
int pass = 0;

printf("\n Enter the password : \n");  
gets(buff);

if(strcmp(buff, "mrnakupenda"))  
{  
printf ("\n Wrong Password \n");  
}  
else  
{  
printf ("\n Correct Password \n");  
pass = 1;  
}

if(pass)  
{  
printf ("\n welcome Mr*Nakup3nda you got Root privileges \n");  
}*

return 0;  
}  
save this code as buffer.c, compile and run it

RUNNING OUR EXAMPLE

to compile use the following command:  
cc buffer.c -o buffer  
then run it with command  
./buffer

[](https://img.wonderhowto.com/img/original/96/02/63581435204485/0/635814352044859602.jpg)

From the above screenshot i tried to enter the correct and wrong password and it works just fine, but when i run an input of length greater than what buffer can hold (fuzzing technique) and at a particular length of input the buffer overflow so took place that it overwrote the memory of integer 'pass'. So despite of a wrong password, the value of 'pass' became non zero and hence root privileges were granted to an attacker, so now that we know that our program is vulnerable save it and in the next tutorial we will test it with automated tools that will show us the same vulnerability we found by manually buzzing.

0 DAY GIFT

[](https://img.wonderhowto.com/img/original/48/46/63581453181727/0/635814531817274846.jpg)

today i was just hunting the web for bugs and i found that some websites made by a company suffer from an auth bypass vulnerability, im too lazy to contact them, plus i was lazy to find the perfect google dork for the other websites but i tested some of them and it works just perfectly. i guess websites running the 2009 footer are affected, again i did not have time to check all of them, so if you wanna know more about it let me know i will give you details about the vulnerability so you could practice by letting the company know about the bug then give them time to fix and we can later expose the vulnerability but do anything on your own, im not responsible for any damage you will make with this info.

the vulnerability can be exploited in this way

<http://www.website.com/admin/>  
username:ADMIN' OR 1=1#

password:anything