# Fuzzing

## Summary

Fuzzing is a term for functional boundary testing of software to reveal faults. In this exercise, we will learn about fuzzing in Ubuntu Linux. The goal of this laboratory exercise is to obtain hands-on practice with a Fuzzer framework. In this exercise, we’ll be fuzzing a vulnerable server application Vulnserver (runs on Ubuntu) to identify some of its vulnerabilities. We will separately analyze the original source.



Figure -https://www.peach.tech/resources/peachcommunity

## Prerequisites

* Setup an Ubuntu VM as outlined in the VM setup instructions on Blackboard
* Install the dependencies

$ sudo apt-get install mono-complete

$ cd ~/Downloads && wget https://sourceforge.net/projects/peachfuzz/files/Peach/3.1/peach-3.1.124-linux-x86\_64-release.zip

$ sudo mkdir /opt/peach-3.1.124 && cd /opt/peach-3.1.124

$ sudo unzip ~/Downloads/peach-3.1.124-linux-x86\_64-release.zip

* Add the following alias to your ~/.bashrc

alias peach="/opt/peach-3.1.124/peach"

* Source the new bashrc

$ source ~/.bashrc

* Disable the firewall

$ sudo ufw disable

* Clone the repository into your sandbox

$ git clone https://gitlab.com/underpantsgnomes/softwaresecurity/fuzzerlab.git

* Import the project into eclipse and build vulnserver as we have done for previous exercises

## Background

* See the slides

## Details

### Overall sequence

1. Running vulnserver on the Ubuntu VM
2. HTER command fuzzing sequence setup and execution (it takes a while!)
3. Static analysis of the original source (Can you spot any suspicious areas?)
4. Inspection of the HTER fuzzing results (What are we looking for? Did this correlate with anything identified in step 3?)
5. Implement additional peach pit files to fuzz these other commands
6. Results comparison to the original source (How good were the fuzzer results?)

### vulnserver

Vulnserver is a http server program that is a port of Stephen Bradshaw’s version of the same for Windows OS. It was originally a Windows based threaded TCP server application that was designed to be exploited. We have ported it to Ubuntu and additionally used tinyhttpd as a reference for this port.

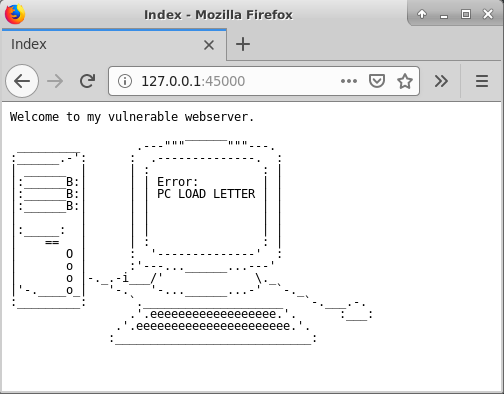
* Run the server in a terminal session

$ Debug/vulnserver

* + You should see the following output

|  |
| --- |
| vulnserver httpd version 1.00 running on port 45000  This is vulnerable software!  Do not allow access from untrusted systems or networks!  Called essential function so version 1.00 |

* From a browser, connect to the server at 127.0.0.1:45000
  + You should see the following in your browser



* From another terminal session, connect to the server using netcat

$ netcat 127.0.0.1 45000

* Type HELP and press enter and you should see the following output

|  |
| --- |
| Valid Commands:  HELP  STATS [stat\_value]  RTIME [rtime\_value]  LTIME [ltime\_value]  SRUN [srun\_value]  TRUN [trun\_value]  GMON [gmon\_value]  GDOG [gdog\_value]  KSTET [kstet\_value]  GTER [gter\_value]  HTER [hter\_value]  LTER [lter\_value]  KSTAN [lstan\_value]  EXIT |

### Example session fuzzing

In this session, we will fuzz the vulnserver target’s HTER command

* Open the hterl.xml file for editing line 23 and insert your VM’s ip address in the location field where VmIpAddressHere is below

<Agent name="LocalAgent" location="tcp://VmIpAddressHere:9001">

* Start the monitor (agent)

$ peach -a tcp

* + You should see the following output

|  |
| --- |
| [[ Peach v3.1.124.0  [[ Copyright (c) Michael Eddington  [\*] Starting agent server  -- Press ENTER to quit agent --  [Thread debugging using libthread\_db enabled]  Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".  vulnserver httpd version 1.00 running on port 45000  This is vulnerable software!  Do not allow access from untrusted systems or networks!  Called essential function so version 1.00 |

* From another terminal, start the fuzzer (publisher)

$ peach hterl.xml TestHTER

* + You should see the following output

|  |
| --- |
| [[ Peach v3.1.124.0  [[ Copyright (c) Michael Eddington  [\*] Test 'TestHTER' starting with random seed 42146.  [R1,-,-] Performing iteration  [1,-,-] Performing iteration  [\*] Fuzzing: DataHTER.DataElement\_1  [\*] Mutator: StringMutator  [2,-,-] Performing iteration  [\*] Fuzzing: DataHTER.DataElement\_1  [\*] Mutator: DataElementSwapNearNodesMutator  … |

* Let the fuzzing sequence continue to execute while we perform the next step

### Static analysis of the original source

* Inspect the “vulnserver.c” source file for vulnerabilities
* Write down suspicious functions and any commands that an external attacker could potentially call them with
  + For any commands of interest, reverse engineer the parameter format
  + Have a look at the HTER command, this is what we’re fuzzing first

### Inspection of the results from fuzzing HTER

* Stop the fuzzing sequence by pressing “ctrl-c” to halt peach running in the Linux VM
* Peach creates a folder pitfilename.xml\_TestName\_DateTimestamp
  + Peach creates a /Faults folder within that contains a set of folders with interesting test cases
    - Exploitable test cases will be in general folders named /Faults/EXPLOITABLE\_\*
      * Inside these will be specific /Faults/EXPLOITABLE\_\*/testNumber instances of test cases that generated the condition
        + see /Faults/EXPLOITABLE\_\*/testNumber/2.Initial.Action\_1.bin for the input condition that generated the crash
        + see /Faults/EXPLOITABLE\_\*/testNumber/RemoteAgent.Monitor.WindowsDebugEngine.StackTrace.txt for the stack trace details following the crash
* You can reproduce by remotely connecting to vulnserver using netcat and entering the contents from this file as the command below

|  |
| --- |
| HTER AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA |

### Implement additional peach pit files to fuzz these other commands

* Augment the pit file (add StateModels, DataModels, and Tests) to enable fuzzing of the following commands
  + KSTET
  + TRUN
  + GMON
  + GTER
  + LTER

### Evaluate the fuzzer results

* How did the fuzzer results compare with your results from static analysis?
* How long did it take the fuzzer to realize a serious flaw?

## References

* Rahul Nair, course assistant for JHU-ISI EN.650.660, 2016
* <https://gitlab.com/underpantsgnomes/softwaresecurity/fuzzerlab.git>
* <http://www.thegreycorner.com/p/vulnserver.html>
* <http://www.thegreycorner.com/2010/12/introducing-vulnserver.html>
* <https://sourceforge.net/projects/tinyhttpd>
* <https://www.peach.tech/resources/peachcommunity>