Charger Active Shield v1.0 - User Guide

Noah Sickels, Adam Brannon, William Lochte – G12

Text

Description automatically generated with medium confidence

 [](https://www.python.org/downloads)  

## Table of Contents

* [Project Overview](#project-overview)
  + [Project Directory Structure](#project-directory-structure)
* [Prerequisites](#prerequisites)
* [Testbed Configuration](#testbed-configuration)
* [Usage & Installation](#usage--installation)
  + [Bash Script (Recommended)](#bash-script-recommended)
  + [Dockerfile *(WIP)*](#dockerfile-wip)
  + [Manual Installation (Recommended)](#manual-installation-recommended)
* [References](#references)

## Project Overview

Our senior design group is the second team working on the Charger Active Defense project. This project aims to develop a fuzzing workflow that effectively tests the networking aspects of the selected target applications, Medusa and Masscan. We strive to identify any hangs or crashes that may occur, which can then be sent back to the host machine to potentially disrupt or halt the adversary’s tool.

This project is divided into two main phases - the fuzzing workflow and the active defense tool. The fuzzing workflow phase consists of the selection of fuzzing tools, two attack tools to fuzz, and the development of a fuzzing workflow. The active defense tool phase consists of developing a tool that can detect and respond to attacks on the network and send the fuzzed responses back to the adversary’s tool.

You can find the sponsor’s project proposal slide below.

Text

Description automatically generated

### Project Directory Structure

#### **Directory Tree**

Charger Active Defense v1.0 - Senior Design Project  
.  
├── README.md  
├── config  
│ ├── Metasploitable2\_Running\_Services.txt  
│ └── Testbed\_Config.md  
├── deliverables  
│ ├── Conference-template-A4.doc  
│ ├── G12\_attack\_tool\_selection\_report.docx  
│ ├── G12\_fuzz\_tool\_selection\_report.docx  
│ ├── G12\_fuzzing\_results\_analysis.docx  
│ ├── G12\_updated\_milestones.docx  
│ ├── G12\_updated\_timeline.png  
│ ├── Project\_Timeline\_v2.gan  
│ ├── briefings  
│ │ ├── brief\_1  
│ │ │ ├── G12\_briefing\_1\_progress\_report.docx  
│ │ ├── brief\_2  
│ │ │ ├── G12\_briefing\_2\_progress\_report.docx  
│ │ │ └── fuzzowski\_medusa\_telnet.pcap  
│ │ └── brief\_3  
│ │ └── G12\_briefing\_3\_progress\_report.docx  
│ ├── design\_review  
│ │ ├── 495\_488\_design\_review\_template.pptx  
│ │ ├── Behavioral\_Decomposition.vsdx  
│ │ ├── Functional\_Decomposition.vsdx  
│ │ ├── G12\_design\_review\_presentation.pptx  
│ │ ├── G12\_level\_of\_effort.docx  
│ │ ├── G12\_marketing\_requirements.docx  
│ │ ├── Updated\_Behavioral\_Decomposition.png  
│ │ ├── Updated\_Functional\_Decomposition.png  
│ │ └── individual\_level\_of\_effort.md  
│ ├── final\_report  
│ ├── proposal  
│ │ └── Project-Proposal-Submission.pdf  
│ └── timeline\_and\_milestones  
│ └── initial  
│ ├── Project\_Timeline\_Proposal.gan  
│ └── milestone\_analysis.md  
├── fuzzing  
│ ├── afl-qemu-trace  
│ ├── fuzzowski.medusa.ftp  
│ │ └── ftp.py  
│ ├── fuzzshark  
│ │ └── ~src  
| ├── icmp.masscan  
│ │ ├── fuzz\_ping.sh  
│ │ ├── grammer.bnf  
│ │ ├── internet\_checksum.py  
│ │ ├── requirements.txt  
│ │ └── send\_icmp.py   
│ ├── medusa.postgresql.afl\_1  
│ │ ├── cmdline  
│ │ ├── fuzz\_bitmap  
│ │ ├── fuzzer\_setup  
│ │ ├── fuzzer\_stats  
│ │ ├── ~hangs  
│ │ ├── init\_attempt  
│ │ │ ├── medusa\_config.txt  
│ │ │ ├── wrapper.c  
│ │ │ └── wrapper.sh  
│ │ ├── ~plot\_data  
│ │ └── ~queue  
│ ├── peach\_fuzz  
│ │ ├── network\_fuzzing.xml  
│ │ └── peachfuzzer.dockerfile  
│ ├── radamsa  
│ │ ├── Radamsa\_Instructions.md  
│ │ └── ~img  
│ ├── randbytes  
│ │ ├── ftp\_server.py  
│ │ └── pcap\_parsing.py  
│ ├── randpkt  
│ │ └── ~src  
│ └── scapy.radamsa  
│ └── radamsa\_scapy\_pcap\_fuzzing.py  
├── misc  
│ ├── Attack\_Tool\_Commands.md  
│ ├── Attack\_Tool\_Info.md  
│ ├── password\_list.txt  
│ └── repeat\_medusa.sh  
├── pcaps  
│ ├── baseline  
│ │ ├── masscan.pcap  
│ │ ├── medusa\_ftp.pcap  
│ │ ├── medusa\_postgresql.pcap  
│ │ └── medusa\_ssh.pcap  
│ └── scapy  
│ ├── ftp\_login\_packets.pcap  
│ ├── fuzz\_test\_1.pcap  
│ ├── medusa\_ftp\_brute\_force.pcap  
│ ├── medusa\_ftp\_fail.pcap  
│ └── nmap\_ftp\_scan.pcap  
├── project\_overview.png  
├── research  
│ ├── CVEs.md  
│ ├── Fuzzing\_Tools.md  
│ └── cmiller-csw-2010.pdf  
└── tests  
 ├── ldra  
 │ ├── aircrack-ng  
 │ │ └── aircrack-ng.mts.htm  
 │ ├── masscan  
 │ │ └── masscan.mts.htm  
 │ ├── medusa  
 │ │ └── medusa.mts.htm  
 │ ├── netdiscover  
 │ │ └── netdiscover.mts.htm  
 │ ├── reaver  
 │ │ └── reaver.mts.htm  
 │ └── yersinia  
 │ └── yersinia.mts.htm  
 └── valgrind  
 ├── commands.txt  
 ├── masscan.txt  
 ├── medusa\_ftp.txt  
 ├── medusa\_postgres.txt  
 ├── medusa\_ssh.txt  
 └── netdiscover.txt

##### ***Explanation***

* **README.md:** This file.
* **project\_overview.png:** Image of the project overview.
* **config:** Contains configuration files.
  + **Testbed\_Config.md:** Configuration details for the testbed.
* **deliverables**: Contains project deliverables, including the tool reports, proposal presentation slides, briefings, design review, and final report.
  + **G12\_attack\_tool\_selection\_report.docx:** Attack tool selection report.
  + **G12\_fuzz\_tool\_selection\_report.docx:** Fuzz tool selection report.
  + **G12\_fuzzing\_results\_analysis.docx:** Fuzzing results analysis.
  + **G12\_updated\_milestones.docx:** Updated milestones.
  + **G12\_updated\_timeline.png:** Updated timeline.
  + **Project\_Timeline\_v2.gan:** Gantt chart file for the project timeline.
  + **briefings:** Contains briefing files.
    - **brief\_1:** Briefing 1 files.
      * **G12\_briefing\_1\_progress\_report.docx:** Briefing 1 progress report.
    - **brief\_2:** Briefing 2 files.
      * **G12\_briefing\_2\_progress\_report.docx:** Briefing 2 progress report.
      * **fuzzowski\_medusa\_telnet.pcap:** Fuzzowski Medusa Telnet PCAP file.
    - **brief\_3:** Briefing 3 files.
      * **G12\_briefing\_3\_progress\_report.docx:** Briefing 3 progress report.
  + **design\_review:** Contains design review files.
    - **495\_488\_design\_review\_template.pptx:** Design review template.
    - **Behavioral\_Decomposition.vsdx:** Behavioral decomposition Visio diagram.
    - **Functional\_Decomposition.vsdx:** Functional decomposition Visio diagram.
    - **G12\_design\_review\_presentation.pptx:** Design review presentation.
    - **G12\_level\_of\_effort.docx:** Level of effort document.
    - **G12\_marketing\_requirements.docx:** Marketing requirements document.
    - **Updated\_Behavioral\_Decomposition.png:** Updated behavioral decomposition diagram image.
    - **Updated\_Functional\_Decomposition.png:** Updated functional decomposition diagram image.
    - **individual\_level\_of\_effort.md:** Individual level of effort document.
  + **final\_report:** Final report files.
  + **proposal:** Proposal files.
    - **Project-Proposal-Submission.pdf:** Project proposal presentation slides.
  + **timeline\_and\_milestones:** Contains timeline and milestones files.
    - **initial:** Initial timeline and milestones.
      * **Project\_Timeline\_Proposal.gan:** Initial project timeline proposal.
      * **milestone\_analysis.md:** Milestone analysis.
* **fuzzing:** Contains fuzzing-related files.
  + **afl-qemu-trace:** AFL QEMU trace binary.
  + **fuzzowski.medusa.ftp:** Fuzzowski Medusa FTP files.
    - **ftp.py:** FTP file for Fuzzowski Medusa.
  + **fuzzshark:** Fuzzshark files.
  + **medusa.postgresql.afl\_1:** Medusa PostgreSQL AFL files.
    - **init\_attempt:** Initial attempts with AFLnet.
      * **medusa\_config.txt:** Medusa configuration file for wrapper.
      * **wrapper.c:** Custom wrapper source file.
      * **wrapper.sh:** Custom wrapper script.
  + **peach\_fuzz:** Peach Fuzz files.
    - **network\_fuzzing.xml:** Network fuzzing XML model file.
    - **peachfuzzer.dockerfile:** Peach Fuzzer Dockerfile.
  + **radamsa:** Radamsa files.
    - **Radamsa\_Instructions.md:** Radamsa testing instructions.
  + **randbytes:** Randbytes files.
    - **ftp\_server.py:** FTP server file.
    - **pcap\_parsing.py:** PCAP parsing file with Scapy.
  + **randpkt:** Randpkt files.
  + **scapy.radamsa:** Scapy Radamsa files.
    - **radamsa\_scapy\_pcap\_fuzzing.py:** Radamsa & Scapy PCAP fuzzing Python script.
* **misc:** Miscellaneous files.
  + **Attack\_Tool\_Commands.md:** Commands for attack tools used during compatibility testing.
  + **Attack\_Tool\_Info.md:** Information about attack tools.
  + **password\_list.txt:** Password list used for testing.
  + **repeat\_medusa.sh:** Script to repeatedly run Medusa.
* **pcaps:** Contains PCAP files.
  + **baseline:** Baseline PCAP files.
  + **scapy:** Scapy PCAP files.
* **tests:** Contains test-related files.
  + **ldra:** LDRA test files.
    - **aircrack-ng/aircrack-ng.mts.htm:** Aircrack-ng LDRA test files.
    - **masscan/masscan.mts.htm:** Masscan LDRA test files.
    - **medusa/medusa.mts.htm** Medusa LDRA test files.
    - **netdiscover/netdiscover.mts.htm:** Netdiscover LDRA test report.
      * **netdiscover.mts.htm:** HTML report for Netdiscover.
    - **reaver/reaver.mts.htm:** Reaver LDRA test report.
    - **yersinia/yersinia.mts.htm:** Yersinia LDRA test report.
  + **valgrind:** Valgrind test results for each attack tool candidate.
    - **commands.txt:** Commands used for running the Valgrind tests.
    - **masscan.txt:** Masscan Valgrind test results file.
    - **medusa\_ftp.txt:** Medusa FTP Valgrind test results file.
    - **medusa\_postgres.txt:** Medusa PostgreSQL Valgrind test results file.
    - **medusa\_ssh.txt:** Medusa SSH Valgrind test results file.
    - **netdiscover.txt:** Netdiscover Valgrind test results file.
* **research:** Contains research-related files.
  + **CVEs.md:** List of CVEs from all attack tool candidates.
  + **Fuzzing\_Tools.md:** Background research on possible fuzzing tools.
  + **cmiller-csw-2010.pdf:** Research paper on general fuzzing and fuzzing tools.

## Prerequisites

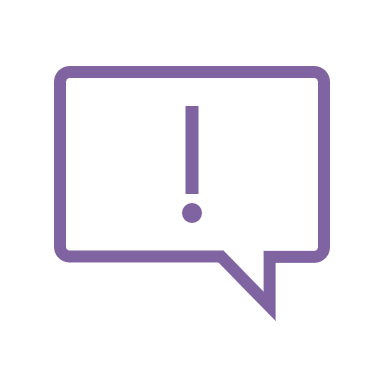
* VirtualBox 7.1.0 (or later)
* Kali Linux 2023.4 (or later) or Ubuntu 20.04 (or later)
* Packages: clang, graphviz-dev, libcap-dev, git, make, gcc, autoconf, automake, libssl-dev, wget, curl

## Testbed Configuration

|  |  |  |
| --- | --- | --- |
| Component | Configuration |  |
| Hypervisor | VirtualBox | 7.1.0 |
| Virtual Machine 1 (Host) | OS | Kali Linux 2024.3 |
| Kernel Version | Linux kali 6.10.11-amd64 |
| GCC Version | 14.2.0 (Debian 14.2.0-3) |
| Network Adapter 1 | NAT |
| Network Adapter 2 | Internal Network *(intent)* |
| IP Address | 192.168.1.99 /24 |
| Miscellaneous | 16,384 MB RAM, 2 CPU cores, 80 GB HDD |
| Virtual Machine 2 (Target) | OS | Metasploitable2 |
| Kernel Version | Linux Metasploitable 2.6.24-16-server |
| GCC Version | 4.2.4 (Ubuntu 4.2.4-lubuntu4) |
| Network Adapter 1 | Internal Network *(intent)* |
| Network Adapter 2 | *Optional* |
| IP Address | 192.168.1.100 /24 |
| Miscellaneous | 2,048 MB RAM, 1 CPU core, 8 GB HDD |

**Important**

“*Virtual Machine 1 (Host*)”refers to the attacking virtual machine running Kali, which runs Medusa and Masscan against the target VM.  
  
“*Virtual Machine 2 (Target)*”refers to the virtual machine running Metasploitable2, which has vulnerable services active.



## Usage & Installation

There are three ways to install and use the tools necessary for the Chadv1.0 workflow: using the Bash script, the Dockerfile, or manually.

### Bash Script (Recommended)

To install the attack tools and fuzzing tools, you can use the provided Bash script as shown below (***requires root privileges***):

|  |
| --- |
| # Download the workflow script through curl or manually from the repository curl -O https://raw.githubusercontent.com/NCSickels/chadv1.0/main/workflow.sh  # Make the script executable chmod u+x workflow.sh  # Using the workflow script sudo ./workflow.sh --help  # To install all tools (attack and fuzzing) sudo ./workflow.sh install  # To build all tools (attack and fuzzing) sudo ./workflow.sh build |

**Note**

If you encounter the error: *-bash: ./workflow.sh: /bin/bash^M: bad interpreter: No such file or directory*, it is most likely due to the script being in DOS format for a UNIX system. To fix this, you can use the *dos2unix* command to convert the script to UNIX format. You can install it through Apt package manager using the command *sudo apt install dos2unix*.



### 

### Dockerfile *(WIP)*

**Warning**

*Requires Docker and Make to be installed on the host machine. Docker Desktop is available* [*here.*](https://www.docker.com/get-started/)



The Chad workflow Docker implementation utilizes a Makefile and the make utility to build and run the Dockerfile image in a streamlined manner.

* Build the Chadv1.0 Workflow Docker image: make build
* Run the Chadv1.0 Workflow Docker container: make run

Optionally, you can build the Docker image and run the container manually using the commands below.

**Build the Docker Image**

|  |
| --- |
| # Build the Docker image docker build -t workflow . |

**Run the Docker Container**

|  |
| --- |
| # Run the Docker container docker run --rm -it --name workflow -v . workflow /bin/bash |

### Manual Installation (Recommended)

**Clone the Repositories**

|  |
| --- |
| # Clone the attack tool repositories git clone https://salsa.debian.org/pkg-security-team/medusa.git  git clone https://github.com/robertdavidgraham/masscan.git  # Clone the fuzzing tool repositories git clone https://github.com/aflnet/aflnet.git git clone https://gitlab.com/akihe/radamsa.git |

**Install Necessary Dependencies**

|  |
| --- |
| sudo apt install -y clang graphviz-dev libcap-dev git make \  gcc autoconf automake libssl-dev wget curl |

**Build the Attack Tools**

|  |
| --- |
| # Build Medusa cd medusa ./configure make make install cd ..  # Build Masscan cd masscan # Optionally, can run `make -j` for faster compilation make  make install cd .. |

**Build the Fuzzing Tools**

|  |
| --- |
| # Build AFLnet cd aflnet make clean all  cd llvm\_mode make # If this command does not work, it most likely means that llvm-config is not in your PATH. If so, you can add it manually as shown below. # It should be named something like llvm-config-6.0 in /usr/bin/ export LLVM\_CONFIG=$(ls /usr/bin/llvm-config-\* 2>/dev/null | head -n 1) cd ../.. export AFLNET=$(pwd)/aflnet export WORKDIR=$(pwd) export PATH=$PATH:$AFLNET export AFL\_PATH=$AFLNET cd ..  # Build Radamsa cd radamsa make sudo make install cd .. |

This will install the necessary tools for the Chadv1.0 fuzzing workflow, including AFLnet, Radamsa, Medusa, and Masscan.

## References

### Attack Tools

* [Medusa](https://salsa.debian.org/pkg-security-team/medusa)
* [Masscan](https://github.com/robertdavidgraham/masscan.git)

### Fuzzing Tools

* [AFLnet](https://github.com/aflnet/aflnet)
* [Radamsa](https://gitlab.com/akihe/radamsa.git)

### Other

* [VirtualBox](https://www.virtualbox.org/wiki/Downloads)
* [Docker Desktop](https://www.docker.com/get-started/)
* [Kali Linux 2023.4 Pre-built VMs](https://www.kali.org/get-kali/#kali-virtual-machines)
* [Ubuntu 20.04 LTS ISO](https://releases.ubuntu.com/focal/)
* [Metasploitable2 VM](https://www.rapid7.com/products/metasploit/metasploitable/)
* [dos2unix](https://dos2unix.sourceforge.io)
* [MD to DOCX Converter (Used to create User Guide from README)](https://cloudconvert.com/md-to-docx)