Charger Active Defense v1.0 Team 2 - Group 12

Noah Sickels, Adam Brannon, William Lochte

**Background / Abstract**

Modern attack tools are efficient, allowing them to attack quickly. To mitigate this risk, it is important to have a way to slow or stop them. Our project plans to accomplish this by finding network responses that cause attack tools to crash or hang, and then providing those network responses whenever the attack tool makes a request. To find such network responses, we plan on performing research on each tool for known CVEs. In addition, we plan on fuzz-testing each tool’s network protocols using well-known fuzz-testing tools such as AFLnet.

**Current Project Status, Issues, & Short-Term Activities & Goals**

The first portion of this project consisted of screening, static analysis, and selection of desired attack tools out of our five candidates, background research into their known vulnerabilities and CVEs, and working with the first fuzz testing tool against Medusa and Masscan. For this reporting period, we successfully used the tool AFLnet, a module add-on to a fork of AFLplusplus specifically designed to fuzz network protocols. Ultimately, we met all of our goals for this reporting period.

Short-term goals met for this period:

* Complete background research on all tools.
* Selection of attack tools to determine which tools to begin testing on.
* Ensure the ability to communicate with the target VM from the host VM on the specific ports or protocols.
* Capture clean network traffic of Medusa and Masscan in use against the target.
* Begin testing the viability of the first fuzz testing tool (AFLnet/AFLplusplus) against Medusa and Masscan.

During this period, we encountered a few issues:

* After background research for any known vulnerabilities or CVEs for all of our five attack tool candidates, the only information we could find was the initial CVEs from their initial release. For example, Yersinia's only relevant CVE was from its debut in 2005 (CVE-2005-4440). We also reviewed their GitHub issues that appeared most relevant by title or with specific labels ("bug," "vuln," etc.) but found nothing of note.
* AFLnet requires you to provide server information ("tcp://192.168.1.100/22", etc.) and the protocol you wish to test against. However, one of our tools - Masscan - scans a subnet through ICMP echo probes through the ICMP protocol and does not use a specific port to do so. As ICMP is a separate protocol and does not designate a port number, providing the correct server information to AFLnet is problematic.
* Another requirement of AFLnet is that the targeted binary needs to be instrumented. Instrumenting the binary involves recompiling the binary from source files using the custom afl-gcc compiler. If the binary isn't recompiled, AFLplusplus throws an error and prevents further testing in some cases. When trying to instrument the binary for Medusa and Masscan, it successfully recompiled Masscan without any issues. However, Medusa had a dependency problem, as it required specific OpenSSL header files from a deprecated version of the OpenSSL library that was not available on the Ubuntu or Kali Linux version we are using.
* A significant complication that surfaced and we will address in the future is that Masscan has its own custom ad-hoc TCP/IP protocol stack, so anything other than simple port scans on a target can cause conflicts with the local TCP/IP stack. While AFLnet worked on Masscan, it was incredibly slow, as the translation between the custom TCP/IP stack to the AFLnet wrapper on the host caused the program to throttle.
* AFLnet is limited in the protocols that it supports. During static analysis against Medusa, we noticed that the memory leaks varied immensely depending on the Medusa module you used. The main example is Medusa's SSH module with 3,256 bytes of memory lost, against Medusa's PostgreSQL module with 38,342 bytes of memory lost. From the results, we wanted to use the PostgreSQL module as it seemed most favorable, but AFLnet lacks direct support for the PostgreSQL protocol.

For the next reporting period, our short-term goals are:

|  |  |
| --- | --- |
| Task | Responsibility |
| Begin viability/compatibility testing on either Fuzzowski or Scapy with Medusa and Masscan. | Noah |
| Research other networking-based protocol fuzzing applications and select a third fuzzing target for the next reporting period | William |
| Continue fuzz testing with AFL++ and try to apply it with the chosen targets | Adam |

**Milestone Status Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Milestone / Task | Projected Due Date | Completion Date | Status | Notes |
| Background Research for CVEs & Exploits | 9/24 | 9/23 | Completed | Found all relevant CVEs/Exploits from Exploit-db & NVD for all attack tool candidates. |
| Apply LDRA static analysis & Valgrind on All Attack Tools | 9/24 | 9/30 | Completed | Fully screened all attack tool candidates with both LDRA and Valgrind. |
| Attack Tool Selection | 9/30 | 9/30 | Completed | Testing will focus on Medusa and Masscan as the two primary attack tools. |
| Identify Fuzz Testing Tools | 9/30 | 10/7 | Partial/Incomplete | First tool selected was AFLnet, but the second and third tools were still |
| Network Packet Capture on Medusa & Masscan | 10/1 | 10/1 | Completed | Captured .pcap files for regular use of Medusa and Masscan on target VM. |
| Test Fuzzing Tool #1 | 10/7 | 10/7 | Completed | Successfully configured AFLnet to properly test both Medusa and Masscan. |

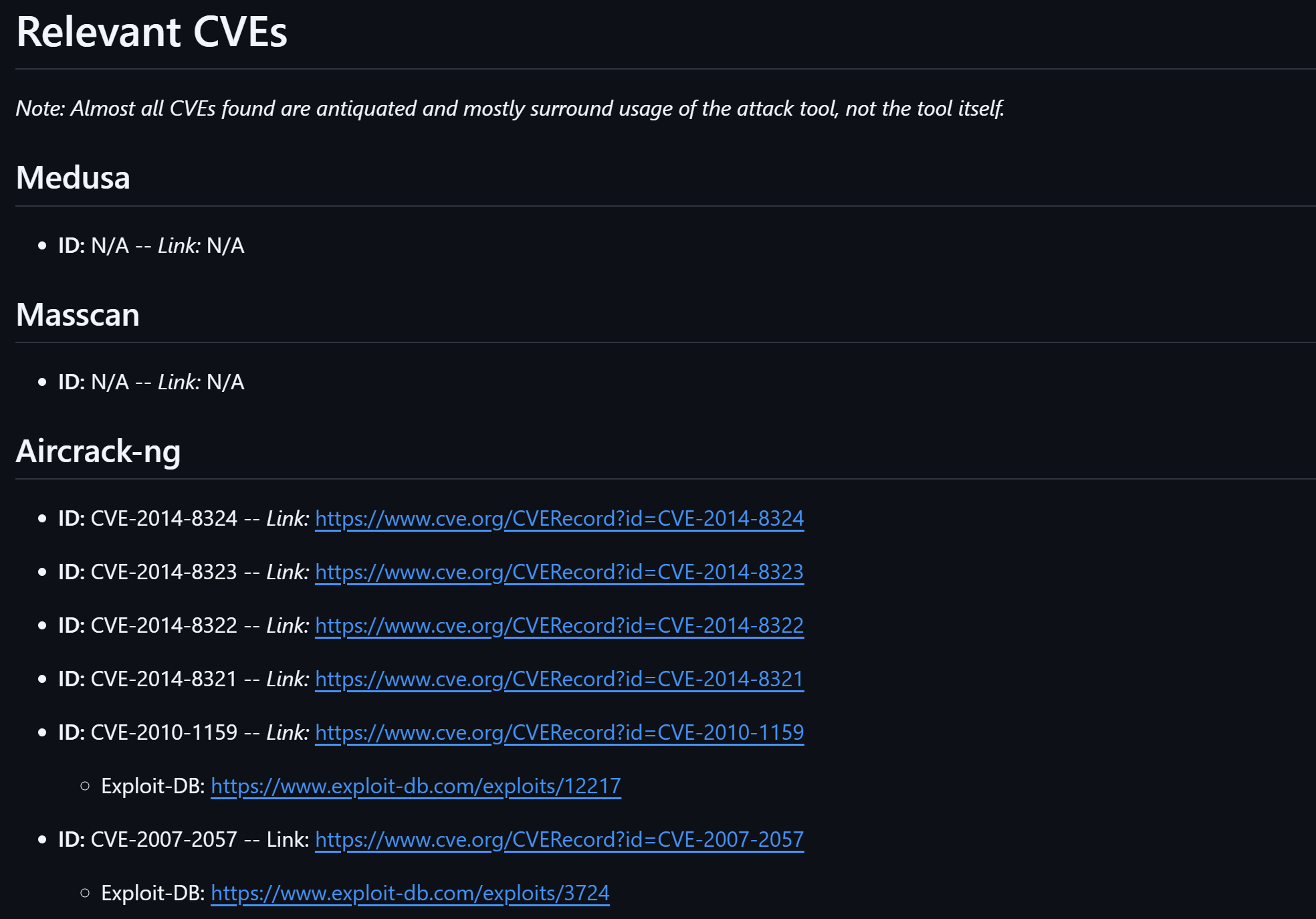
**Level of Effort / Individual Responsibility Record**

|  |  |
| --- | --- |
| Member | Hours |
| Noah Sickels | 56 |
| Adam Brannon | 16 |
| William Lochte | 14 |

|  |  |
| --- | --- |
| Member | Individual Accomplishments |
| Noah Sickels | Static analysis against all attack tool candidates.  Background research of known vulnerabilities and CVEs.  Capture of unaltered network traffic of Medusa and Masscan against target.  AFLnet configuration and testing against Medusa and Masscan. |
| Adam Brannon | Research on fuzz testing using simple and smart fuzzers  Setup AFL++ fuzzing tool  Testing AFL++ fuzzer on simple C/C++ programs |
| William Lochte | Research on existing projects  Research on potential fuzz testing tools  Working on slides for presentation |

**Milestone Completion & Analysis**

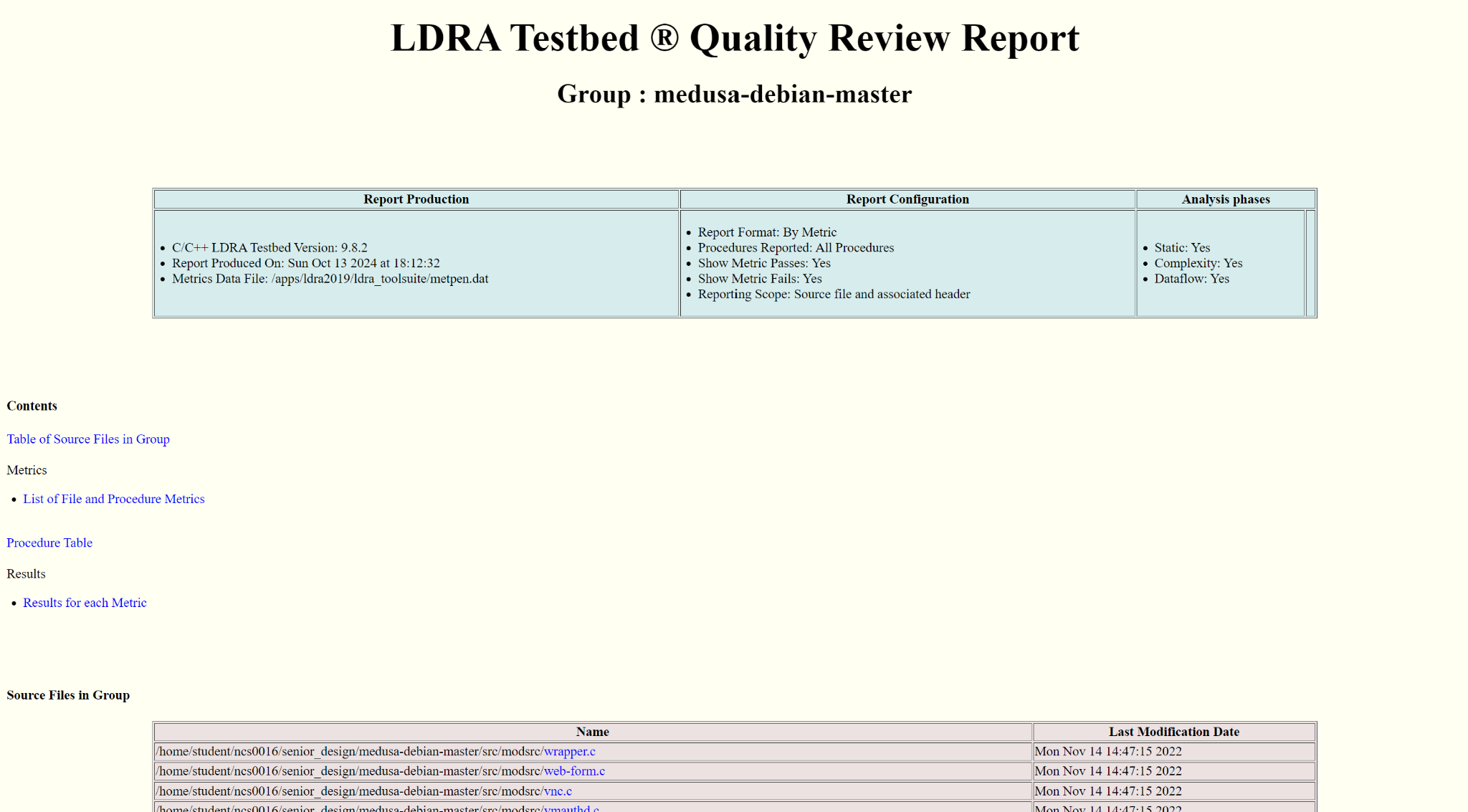
* Written Proposal slides completed
* Background research on the attack tools to find relevant vulnerabilities and CVEs.



* Static analysis of attack tools.

A screenshot of a computer program

Description automatically generated



* Attack tool selection – Medusa (1) and Masscan (2).
* Network packet captures of standard attacks with Medusa and Masscan against the target VM.

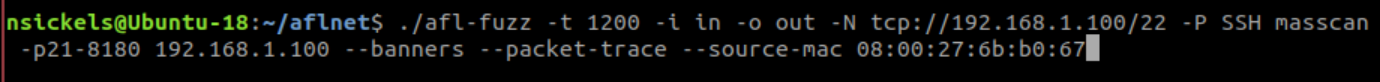
A screenshot of a computer

Description automatically generated

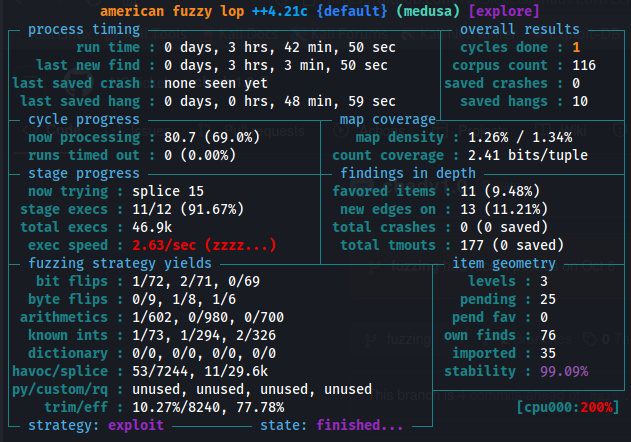
A screenshot of a computer

Description automatically generated

* Fuzz testing with AFLplusplus AFLnet module against Medusa and Masscan for viability.



A screenshot of a computer screen

Description automatically generated

**Mitigation Plan**

* Milestone: Identify Fuzz Testing Tools
  + We will focus on identifying a second possible fuzzing tool at the start of the next briefing period so that we may test it. Throughout the period, we will also research a 3rd possible fuzzing tool.

**Contingency Plans**

* Milestone: Testing Fuzzing Tool #2
  + If we have issues using this tool by the next briefing, then we will just continue with testing the next fuzzing tool. If it takes more than a week to figure out then it is probably not going to work well in our workflow.