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. Our goal is to slow or stop attack tools by sending these tools invalid responses over the network. Some of the major steps that we have made so far to achieve this goal include:

* Researched CVEs for each of our preliminary tools
* Performed various forms of static and dynamic analysis on each tool
* Explored and tested a variety of fuzzing tools

In the future, we plan to:

* Explore more fuzzing tools
* Perform extensive fuzz testing
* Create a fuzz-testing workflow for attack tools

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

This reporting period primarily focused on compatibility testing our second selected fuzz testing tool against Medusa and Masscan attack tools. We also performed research to discover a third fuzzing tool and an alternative fuzzing target.

Short-term goals met for this period:

* Test the viability of the first fuzz testing tool (Fuzzowski/BooFuzz) against Medusa and Masscan.
* Discover a third fuzzing tool (AFL++) and an alternative fuzzing target (reaver).

During this period, we encountered a few issues:

* Fuzzowski is limited in the protocols that it supports and does not natively have support for SSH or PostgreSQL modules as previous testing focused on, but it does have support for Telnet and TFTP. However, since Fuzzowski is a fork of Sully – a fuzz testing framework – it has a lot of custom Python libraries available to simplify writing custom fuzz testing modules for use with other protocols. We wrote a test file for Medusa's FTP module as a proof-of-concept implementation, but more thorough testing is still needed. We wanted to focus on using either the SSH or PostgreSQL modules instead, but due to time limitations, we could not verify before the end of the second reporting period.
* One ongoing challenge we will need to work with more is the difficulty surrounding network protocol fuzz testing with the added factor of the attack tools. While we've discussed several possibilities with our sponsor, including fuzz testing the network-specific source files or using custom network sockets to mediate the communication between the target and the attack tool, we will need to explore these options more.
* Expanding upon the previous issue, while testing Fuzzowski with Medusa's FTP module, verifying that the fuzz testing is working correctly is challenging as it focuses more on fuzzing the port and protocol itself, as provided. For this reason, we decided to script Medusa to continually perform the brute-forcing while Fuzzowski was running against the specific port and protocol. After Medusa finished, it would repeat the command until the script was terminated. This method allowed Medusa to send traffic back and forth to the target and simulate fuzzing the responses replying to the host.

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

|  |  |
| --- | --- |
| Task | Responsibility |
| Begin viability/compatibility testing on the Medusa & Masscan with Peach Fuzzer. | Noah |
| Begin drafting the Design Review report. | William |
| Begin viability/compatibility testing on the Medusa & Masscan with Scapy or Randpkt. | Adam |

**Milestone Status Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Milestone / Task | Projected Due Date | Completion Date | Status | Notes |
| Test fuzzing tool #2 (Fuzzowski) | 10/14 | 10/14 | Completed | Verified viability with Medusa and Masscan. |
| Write Tool Selection Report | N/A | TBD | Partial / Incomplete | During briefing 1, we were informed that we should have a tool selection report as a separate deliverable. We have started writing it but are still working on it. |
| Identify Fuzz Testing Tools #2 & #3 | 9/30 | 10/14 | Completed | The second tool we selected was Fuzzowski for tool #2, and for we are dividing the group workload between Peach Fuzzer or Scapy for tool #3. |

**Milestones for Next Period**

|  |  |  |  |
| --- | --- | --- | --- |
| Milestone / Task | Start Date | Projected End Date | Notes |
| Test Fuzzing Tool #3 | 10/15 | 10/21 | We will begin testing with our final fuzz testing candidates of Peach Fuzzer. |
| Analyze Fuzz Testing Results | 10/21 | 10/21 | This will focus on examining each of the fuzzing tool’s ease of use, compatibility, and features available to use with Medusa and Masscan. |
| Rank Fuzz Tools Based on Probability of Success | 10/22 | 10/22 | From the above analysis, decide the order of preference between the tested fuzzing tools to use for the workflow. |

**Level of Effort / Individual Responsibility Record**

|  |  |
| --- | --- |
| Member | Hours |
| Noah Sickels | 22 |
| Adam Brannon | 10 |
| William Lochte | 13 |

|  |  |
| --- | --- |
| Member | Individual Accomplishments |
| Noah Sickels | Fuzzowski configuration and testing against Medusa and Masscan - including Medusa repetition script and POC FTP module.  Updated general configuration information on attack tool(s), kernel, and gcc compiler versions for reporting and documentation. |
| Adam Brannon | Start a document to document the reasons why Masscan and Medusa were chosen as primary fuzz targets.  Continue setting up AFL++. |
| William Lochte | Decided that our final attack tool should be reaver, and explored options for fuzz testing it. I decided that AFL++ would work for fuzz testing Reaver. |

**Milestone Completion & Analysis**

* Fuzz testing with Fuzzowski against Medusa and Masscan for viability.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Mitigation Plan**

* Milestone: Test Fuzzing Tool #3
  + If issues arise with getting Peach Fuzzer to work with Medusa or Masscan, we have two other possible solutions with Scapy or with Randpkt.

**Contingency Plans**

* Milestone: If we can’t get testing to work with Masscan or Medusa, Reaver is an alternative attack tool we can target.