Final Project Requirements and Timeline

Charger Active Defense – G12

Noah Sickels, Adam Brannon, William Lochte

## Marketing Requirements

|  |  |
| --- | --- |
| # | Description |
| M1 | The ChAD fuzzing workflow must conduct network-based fuzzing to identify network responses, also known as Active Defense Responses, that can crash or hang adversarial attack tools. |
| M2 | The existing vulnerabilities of six well-known attack tools must be documented |
| M3 | Down select to a single well-known attack tool for testing. |
| M4 | Must use two different AI/LLM models to generate additional attack tools. |
| M5 | Use each AI/LLM to generate three attack tools for fuzz testing. |
| M6 | Software fuzzing tools capable of testing the attack tools must be identified. |
| M7 | Demonstrate a fuzz testing workflow for Masscan and AI-generated attack tools. |
| M8 | The ChAD program must monitor incoming network traffic to the host machine. |
| M9 | All transaction history of incoming and outgoing response packets must be logged and recorded. |
| M10 | The ChAD program must provide an active defense response. |
| M11 | Evaluate the effectiveness of any active defense responses found. |
| M12 | Findings must be documented in an IEEE/ACM-style paper. |

## Engineering Requirements

|  |  |
| --- | --- |
| Marketing Requirements | Engineering Requirements |
| M3 | E1: LDRA static analysis and Valgrind memory leak analysis must be used on both selected attack tools for present vulnerabilities or more favorable testing targets. |
| M6 | E2: Develop a fuzzing workflow using three fuzzing tools and approaches and rank them based on the probability of success with selected well-known attack tools. |
| M4, M5 | E3: Must use GitHub Copilot and Phind models to generate three types of attack tools each - a banner-grabber, password brute-forcer, and a simplistic multi-threaded banner-grabber. |
| M7 | E4: Must demonstrate fuzzing workflow on selected/generated attack tools |
| M1, M7, M10 | E5: The workflow must be capable of finding vulnerabilities (crashes or hangs) within attack tools, if any are found. |
| M11 | E6: We must document whether each network response crashes or hangs the attacking application and calculate and record the average hang time if it hangs the application. |
| M12 | E7: The rationale must be compiled for the selection of existing attack tools, fuzzing tools, compatibility results, and analysis for testing into an IEEE conference paper using proper formatting with font type, size, headers, and two columns per page. |
| M1, M2 | E8: Search MITRE CVE and Exploit-DB databases and compile known vulnerabilities for all six possible attack tools. |
| M8, M10 | E9: The ChAD service must send active defense responses within one minute of detection of an incoming attack. |

## Timeline

All course-defined due dates (briefings, proposal, interim design review, final presentation, demo, documentation, etc.) are highlighted in yellow. All milestones are highlighted purple for Noah S, pink for William L, and green for Adam B.



### Activities, Primary & Secondary Responsibilities, & Dependencies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Activity | Primary | Secondary | Dependencies |
| 1.1 | Research Implementation of TSan into GCC compilers. | Will | Noah | - |
| 1.2 | Add integration hook into existing fuzzing workflow script. | Adam | Will | 1.1 |
| 1.3 | Begin fuzz testing on Masscan with AFLnet & Tsan hook. | Noah | Will | 1.2 |
| 2.1 | Develop Python service interactive UI module with Prompt\_Toolkit library. | Noah | Adam | - |
| 2.2 | Develop Python service CLI argument module with Argparse library. | Noah | Will | - |
| 2.3 | Develop Python service error logging module with Logging library wrapper. | Will | Noah | 2.1 |
| 2.4 | Develop Python service network traffic logging module with Logging library wrapper. | Adam | Noah | 2.3 |
| 3.1 | Prompt Phind model for banner-grabbing Attack Tool (1st) | Adam | Noah | - |
| 3.2 | Debug Phind banner-grabbing tool until it compiles with afl-gcc. (1st) | Adam | Will | 3.1 |
| 3.3 | Prompt Copilot model for banner-grabbing Attack Tool (1st) | Will | Noah | - |
| 3.4 | Debug Copilot banner-grabbing tool until it compiles with afl-gcc. (1st) | Will | Adam | 3.3 |
| 4.1 | Research & determine preferable method for storing active defense responses for the Python service. | Noah | Will | - |
| 4.2 | Write implementation of database handler based on selected storage method. | Noah | Adam | 4.1 |
| 4.3 | Write skeleton for network connector with database handler. | Noah | Adam | 4.2 |
| 5.1 | Integrate 1st Copilot & Phind attack tools into fuzzing workflow script. | Noah | Will | 3.2, 3.4 |
| 5.2 | Debug AFLnet integration with 1st attack tools from both models. | Noah | Adam | 5.1 |
| 5.3 | Begin fuzz testing on 1st Gen-AI attack tool with fuzzing workflow. | Noah | Will | 5.2 |
| 6.1 | Perform LDRA static analysis on 1st set of Gen-AI attack tools using TestBed | Adam | Noah | 3.1, 3.3 |
| 6.2 | Perform memory leak analysis on 1st set of Gen-AI attack tools using Valgrind. | Will | Noah | 3.1, 3.3 |
| 7.1 | Prompt Copilot model for password brute-forcing Attack Tool (2nd) | Will | Noah | - |
| 7.2 | Debug Copilot password brute-forcing attack tool until it compiles with afl-gcc. (2nd) | Will | Noah | 7.1 |
| 7.3 | Prompt Phind model for password brute-forcing Attack Tool (2nd) | Adam | Noah | - |
| 7.4 | Debug Phind password brute-forcing attack tool until it compiles with afl-gcc. (2nd) | Adam | Noah | 7.3 |
| 8.1 | Compare code structure & fuzzing results of 1st Gen-AI Attack tools. | Noah | Will | 6.1, 6.2 |
| 8.2 | Compile analysis results into report. | Noah | Adam | 8.1 |
| 9.1 | Integrate 2nd Copilot & Phind attack tools into fuzzing workflow script. | Noah | Will | 7.2, 7.4 |
| 9.2 | Debug AFLnet integration with 2nd attack tools from both models. | Noah | Adam | 9.1 |
| 9.3 | Begin fuzz testing on 2nd Gen-AI attack tool with fuzzing workflow. | Noah | Will | 9.2 |
| 10.1 | Perform LDRA static analysis on 2nd set of Gen-AI attack tools using TestBed | Will | Noah | 7.1, 7.3 |
| 10.2 | Perform memory leak analysis on 2nd set of Gen-AI attack tools using Valgrind. | Adam | Noah | 7.1, 7.3 |
| 11.1 | Prompt Phind model for multi-threaded banner-grabbing Attack Tool (3rd) | Will | Noah | 7.2, 7.4 |
| 11.2 | Debug Phind multi-threaded banner-grabbing attack tool until it compiles with afl-gcc. (3rd) | Will | Noah | 11.1 |
| 11.3 | Prompt Copilot model for multi-threaded banner-grabbing Attack Tool (3rd) | Adam | Noah | 7.2, 7.4 |
| 11.4 | Debug Copilot multi-threaded banner-grabbing attack tool until it compiles with afl-gcc. (3rd) | Adam | Noah | 11.3 |
| 12.1 | Compare code structure & fuzzing results of 2nd Gen-AI Attack tools. | Noah | Will | 11.1, 11.3 |
| 12.2 | Compile analysis & add results into existing report. | Noah | Adam | 12.1 |
| 13.1 | Integrate 3rd Copilot & Phind attack tools into fuzzing workflow script. | Noah | Adam | 11.2, 11.4 |
| 13.2 | Debug AFLnet integration with 3rd attack tools from both models. | Noah | Will | 13.1 |
| 13.3 | Begin fuzz testing on 3rd Gen-AI attack tool with fuzzing workflow. | Noah | Adam | 13.2 |
| 14.1 | Perform LDRA static analysis on 3rd set of Gen-AI attack tools using TestBed | Adam | Noah | 11.2, 11.4 |
| 14.2 | Perform memory leak analysis on 3rd set of Gen-AI attack tools using Valgrind. | Will | Noah | 11.2, 11.4 |
| 15.1 | Compare code structure & fuzzing results of 3rd Gen-AI Attack tools. | Noah | Adam | 11.2, 11.4 |
| 15.2 | Compile analysis & add results into existing report. | Noah | Will | 15.1 |
| 16.1 | Develop network traffic receiver with Scapy or Pyshark libraries. | Noah | Adam | 2.2, 2.4, 4.1 |
| 16.2 | Develop network traffic sending module with Scapy or Pyshark libraries. | Will | Noah | 2.2, 2.4, 4.1 |
| 17.1 | Compile and analyze all LDRA & Valgrind results. | Adam | Noah | 6.1, 6.2, 10.1, 10.2, 14.1, 14.2 |
| 17.2 | Document results and add into comparison report. | Adam | Will | 17.1 |
| 17.3 | Finalize comparison report. | Adam | Noah | 17.2 |
| 18.1 | Unit tests on accuracy of network module of Python service. | Adam | Noah | 4.3, 16.1, 16.2 |
| 18.2 | Unit tests on Python service error handling capability. | Will | Noah | 2.4 |
| 18.3 | Unit tests on Python service logging module functionality. | Adam | Noah | 2.4 |
| 18.4 | Unit tests on Fuzzing workflow application to Masscan. | Noah | Adam | 1.3 |
| 18.5 | Unit tests on fuzzing workflow application to Gen-AI attack tools. | Noah | Will | 5.1, 9.1, 13.1 |
| 19.1 | Compile and analyze Masscan Tsan fuzzing results | Noah | Adam | 1.3, 17.2 |
| 19.2 | Write results report and add to existing Gen-AI comparison report. | Noah | Will | 17.2, 19.1 |
| 20.1 | Integration tests on fuzzing workflow application to attack tools for generating active defense responses. | Will | Noah | 1.3, 5.3, 18.5 |
| 20.2 | Integration tests for Python service retrieving, handling, and formatting responses from fuzz testing. | Adam | Noah | 18.1, 18.2, 18.3 |
| 20.3 | Integration tests on Python service network send and receive functionality to and from the attacking machine. | Will | Noah | 18.1, 18.2, 18.3 |
| 20.4 | Integration tests on capability of Python service sending active defense responses to attacking machine. | Will | Noah | 20.3 |
| 21.1 | Acceptance tests with results compiled into IEEE paper | Noah | Adam | 20.4 |
| 21.2 | Acceptance tests that fuzzing workflow applies to multiple attack tools. | Adam | Will | 20.1 |
| 21.3 | Acceptance tests that Python service detects incoming network traffic from attacking machine to the victim. | Noah | Will | 20.4 |
| 21.4 | Acceptance tests that Python service sends active defense responses to counter the attacking tool. | Will | Adam | 20.4 |
| 21.5 | Acceptance test that offensive tool crashes or hangs from active defense response, if a vulnerability is found. | Adam | Noah | 20.4 |

### Milestones & Dates

|  |  |  |
| --- | --- | --- |
| Milestone | Start Date | End Date |
| Integration of Masscan with ThreadSanitizer Into Fuzzing Workflow | 1/20 | 1/24 |
| Develop Python Service User Interface | 1/24 | 1/31 |
| Develop Python Service Logger Module | 1/31 | 2/7 |
| Develop Gen-AI Attack Tools #1 | 2/7 | 2/14 |
| Develop Python Service Response Handling Module | 2/7 | 2/14 |
| Apply Fuzzing Workflow to Gen-AI Attack Tools #1 | 2/14 | 2/21 |
| Perform Static Analysis on Gen-AI Attack Tools #1 | 2/14 | 2/21 |
| Develop Gen-AI Attack Tools #2 | 2/21 | 2/28 |
| Compare Results of Analysis & Fuzzing on Gen-AI Attack Tools #1 | 2/21 | 2/28 |
| Apply Fuzzing Workflow to Gen-AI Attack Tools #2 | 2/28 | 3/7 |
| Perform Static Analysis on Gen-AI Attack Tools #2 | 2/28 | 3/7 |
| Develop Gen-AI Attack Tools #3 | 3/7 | 3/14 |
| Compare Results of Analysis & Fuzzing on Gen-AI Attack Tools #2 | 3/7 | 3/14 |
| Apply Fuzzing Workflow to Gen-AI Attack Tools #3 | 3/14 | 3/21 |
| Perform Static Analysis on Gen-AI Attack Tools #3 | 3/14 | 3/21 |
| Compare Results of Analysis & Fuzzing on Gen-AI Attack Tools #2 | 3/14 | 3/21 |
| Develop Python Service Network Receiver Module | 3/21 | 3/28 |
| Develop Python Service Network Sender Module | 3/21 | 3/28 |
| Unit Testing | 3/28 | 4/4 |
| Analyze Masscan ThreadSanitizer Fuzz Results | 3/28 | 4/4 |
| Integration Testing | 4/4 | 4/11 |
| Acceptance Testing | 4/11 | 4/18 |
| Write IEEE Conference Paper | 4/18 | 4/25 |
| Update Documentation & GitHub Repository | 4/25 | 5/2 |