Charger Active Defense v1.0 Team 2 - Group 12

## Team Lead in the Reporting Period: Noah Sickels

Noah Sickels, Adam Brannon, William Lochte

## Background / Abstract

Modern attack tools are highly efficient, while the cybersecurity industry struggles with active defense capabilities. Current passive defenses primarily focus on mitigating threats through means that can disrupt business operations or can be circumvented. Additionally, with the rapid advancement of artificial intelligence and prompt engineering, attackers can instantly generate various attack tools to target organizations.

The Charger Active Defense project focuses on developing a network-based fuzzing workflow to effectively and comprehensively test these known and AI/LLM-generated attack tools. It aims to identify responses that may cause the attacking application to crash or hang. Responses generated could be saved and sent back to the adversary through a Python replay service during a detected attack. Due to the multi-threading nature of many attack tools like Masscan and Medusa, effectively fuzzing these tools is complicated. If a fuzzed response leads to a crash or hang, it typically occurs in a thread separate from the attacking application's instance. This crashed thread may either be reported as a false positive or not counted at all. Because of this discovery, at the customer's request, we are pivoting towards thoroughly fuzzing generated attack tools from large language models.

Since we have successfully applied the fuzzing workflow to Masscan, we will also pursue the integration of ThreadSanitizer (TSan) to determine whether current fuzz testing properly crashes threads. The end goal of this workflow is to assess how we can broaden this method to apply to other attack tools, providing organizations with a strategy to defend their systems against an ever-evolving threat landscape.

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

This reporting period focused on creating our second and third attack tools: a password brute forcer and a multi-threaded banner grabber sourced from GitHub Copilot and Phind, resulting in a total of four tools. We conducted static analysis using LDRA and FlawFinder, as well as dynamic analysis with Valgrind for each tool. After analysis, we applied our fuzzing workflow with each tool, ensuring they compiled correctly with AFLnet's afl-gcc and could successfully run with AFLnet's fuzzing tools. Additionally, we updated our User Guide and README with more information about the Python service and this semester's efforts to ease the documentation workload required by the semester's end.

Our short-term goals for this period were:

* Generate a password (FTP) brute forcing attack tool with GitHub Copilot and Phind AI models.
* Generate a simplistic, multi-threaded banner-grabbing attack tool with GitHub Copilot and Phind AI models.
* Perform LDRA & FlawFinder static analysis on the set of AI brute-force & multi-threaded banner-grabbing attack tools.
* Perform Valgrind dynamic memory leak testing on the set of AI brute-force & multi-threaded banner-grabbing attack tools.
* Apply fuzzing workflow to the set of AI brute-force & multi-threaded banner-grabbing attack tools.

During this period, we encountered a few issues:

* From our initial prompt for the password brute-forcing attack tool from the Phind model, it refused to provide any detailed implementation or source code due to ethical and legal implications, citing the Computer Fraud and Abuse Act (CFAA). We assumed the model flagged the phrase "brute force" and immediately blocked additional requests to generate a working program. Even claiming that we were a security researcher or that the program was just for a proof of concept on an isolated network where we had permission, it still refused to provide any implementation detail aside from the process overview of how FTP worked. We were eventually able to get it to generate a functional program by stating it was a "password checker" instead. Still, it provided a program configured to run as a server program that remote clients would connect to and check the password they provide, rather than a program that acted as a client connecting to a remote server. After more trial and error, we got the model to generate a program correctly as a password brute-forcing tool.
* While we eventually generated a functional password brute-forcing attack tool from the Phind model, the initially provided program took the input of a single password and then remotely tried to connect to the host to "verify" the password. This method would require you to manually run the program with different passwords as input rather than the desired functionality of trying a list of credentials from an input file. When attempting to refine the program to read an input file instead, the Phind model refused to provide the updated source code, except this time, it did not reference any legal or ethical implications or liabilities and instead stated that it could not do it without providing a reason. We tried the next day and successfully got it to generate a functioning program. We assumed the initial response was filtering from that same session window or some other silent error.
* While applying the fuzzing workflow on the multi-threaded banner-grabbing attack tools, the Proxmox server crashed. Upon rebooting, it got stuck in a boot loop from a corrupted bootloader, forcing us to reimage it. Fortunately, we maintained consistent backups and did not incur much loss. Since the multi-threaded banner-grabber tools are relatively simple and do not interact directly with the OS or bootloader, we think the corruption was just happenstance.

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

|  |  |
| --- | --- |
| Responsibility | Task |
| Noah Sickels | Develop network traffic-receiving module with Scapy/Pyshark libraries.  Integrate Git-LFS into the GitHub repository. |
| Adam Brannon | Finalize AI attack tool comparison & results reports. |
| William Lochte | Develop network traffic-sending module with Scapy/Pyshark libraries. |

## Milestone Status Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Milestone / Task | Projected Due Date | Completion Date | Status | Notes |
| Gen-AI Brute-Force Attack Tools Developed | 2/28 | 2/24 | Complete | Successfully generated a brute force attack tool from the Phind & GitHub Copilot models. |
| Results of Analysis & Fuzzing on Gen-AI Banner-Grabber Attack Tools Compared | 2/28 | 2/28 | Complete | Written comparison between prompts, code structure, & end results from static analysis & program functionality. |
| Fuzzing Workflow Applied to Gen-AI Brute-Force Attack Tools | 3/7 | 3/7 | Complete | Applied fuzzing workflow with AFLnet & workflow script to both brute force attack tools. |
| Static Analysis Performed on Gen-AI Brute-Force Attack Tools | 3/7 | 3/2 | Complete | Performed analysis on both brute force attack tools using LDRA, Valgrind, & Flawfinder. |
| Gen-AI Multi-threaded Banner-Grabber Attack Tools Developed | 3/14 | 3/10 | Complete | Successfully generated a multi-threaded banner-grabbing attack tool from the Phind & GitHub Copilot models. |
| Results of Analysis & Fuzzing on Gen-AI Multi-threaded Banner-Grabbing Attack Tools Compared | 3/14 | 3/14 | Complete | Added results of analysis, prompts, & end results to existing comparison report to be finalized during next reporting period. |
| Static Analysis Performed on Gen-AI Multi-threaded Banner-Grabber Attack Tools | 3/21 | 3/15 | Complete | Performed analysis on both multi-threaded banner-grabbing attack tools using LDRA, Valgrind, & Flawfinder. |
| Fuzzing Workflow Applied to Gen-AI Multi-threaded Banner-Grabber Attack Tools Compared | 3/21 | 3/21 | Complete | Applied fuzzing workflow with AFLnet & workflow script to both multi-threaded banner-grabbing attack tools. |

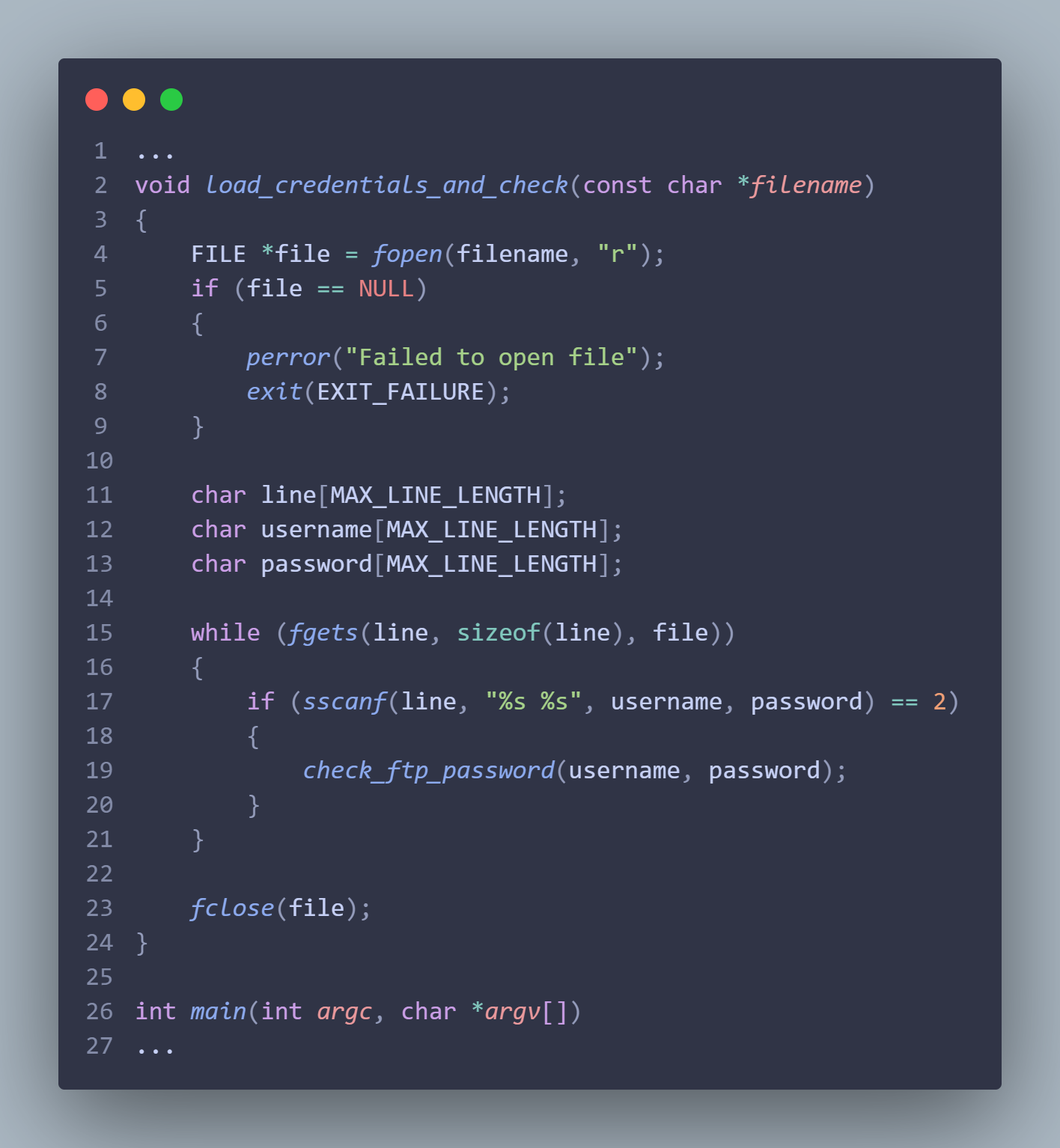
## Level of Effort / Individual Responsibility Record

|  |  |  |
| --- | --- | --- |
| Member | Hours / Period | Total Hours / Spring |
| Noah Sickels | 64 | 158 |
| Adam Brannon | 24 | 63 |
| William Lochte | 20 | 55 |
| Total | 108 | 276 |

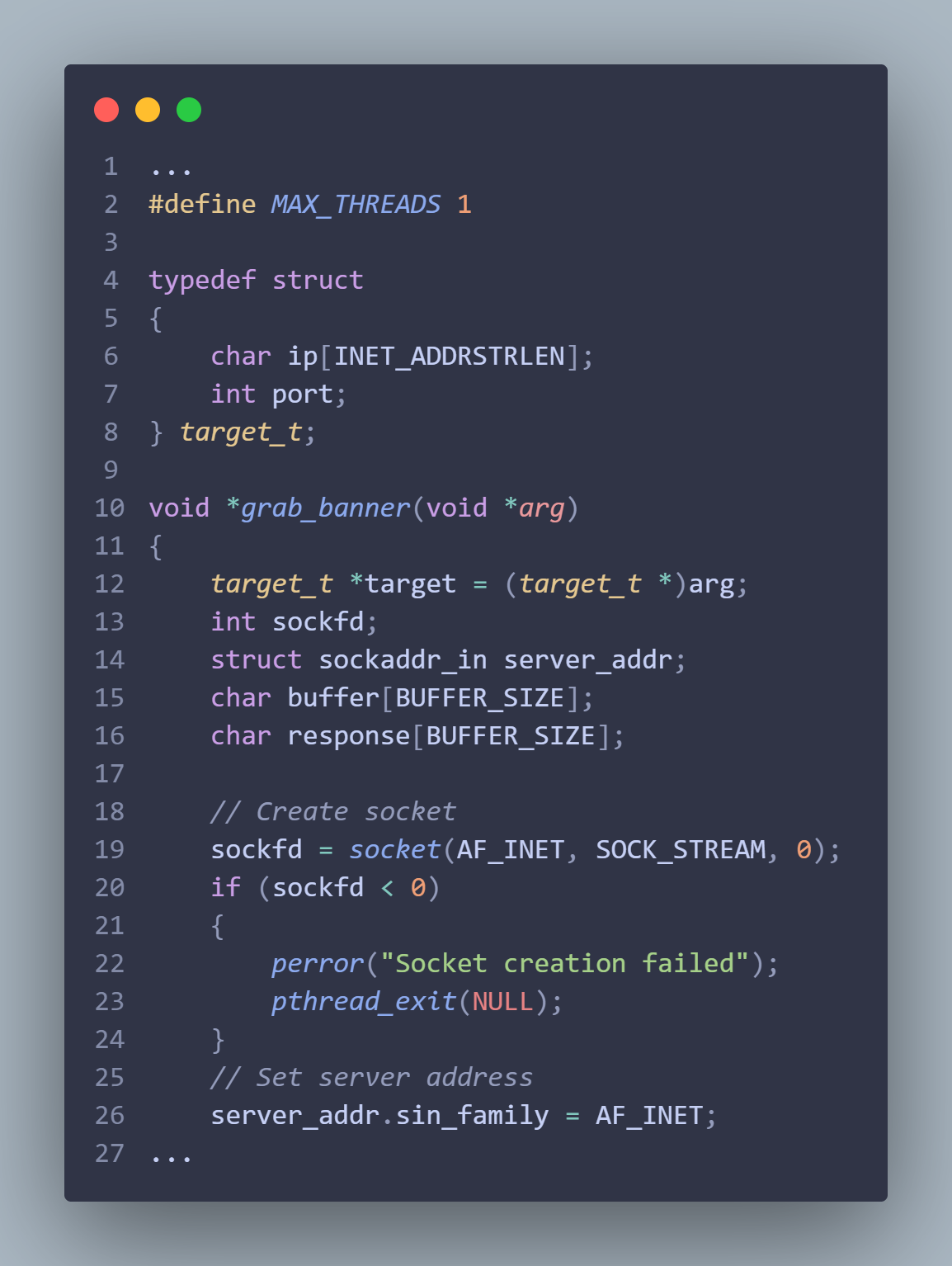
|  |  |
| --- | --- |
| Member | Individual Accomplishments |
| Noah Sickels | Analyzed FlawFinder, Valgrind, and LDRA analysis results from the AI brute-force attack tools & compiled them into a written report.  Analyzed FlawFinder, Valgrind, and LDRA analysis results from the AI multi-threaded banner-grabbing attack tools & added them to the existing report.  Compared & analyzed the code structure, prompt results & feedback for the AI brute-force attack tools & compiled them into a written report.  Compared & analyzed the code structure, prompt results & feedback for the AI multi-threaded banner-grabbing attack tools & added them to the existing report.   Applied the fuzzing workflow with AFLnet to the set of AI brute-force attack tools.  Applied the fuzzing workflow with AFLnet to the set of AI multi-threaded banner-grabbing attack tools.   Updated the README & User Guide with more information on the Python service & the AI attack tools. |
| Adam Brannon | Generated the password brute-force attack tool from the Phind model.  Performed Valgrind memory leak analysis & FlawFinder static analysis on brute-force attack tools from both models.  Generated the multi-threaded banner-grabbing attack tool from the GitHub Copilot model.  Performed LDRA static analysis on multi-threaded banner-grabbing attack tools from both models. |
| William Lochte | Generated the password brute-force attack tool from the GitHub Copilot model.  Performed LDRA static analysis on brute-force attack tools from both models.  Generated the multi-threaded banner-grabbing attack tool from the Phind model.  Performed Valgrind memory leak analysis & FlawFinder static analysis on multi-threaded banner-grabbing attack tools from both models. |

## Milestone Completion & Analysis

* Generated a password brute-forcing attack tool from the GitHub Copilot & Phind models.

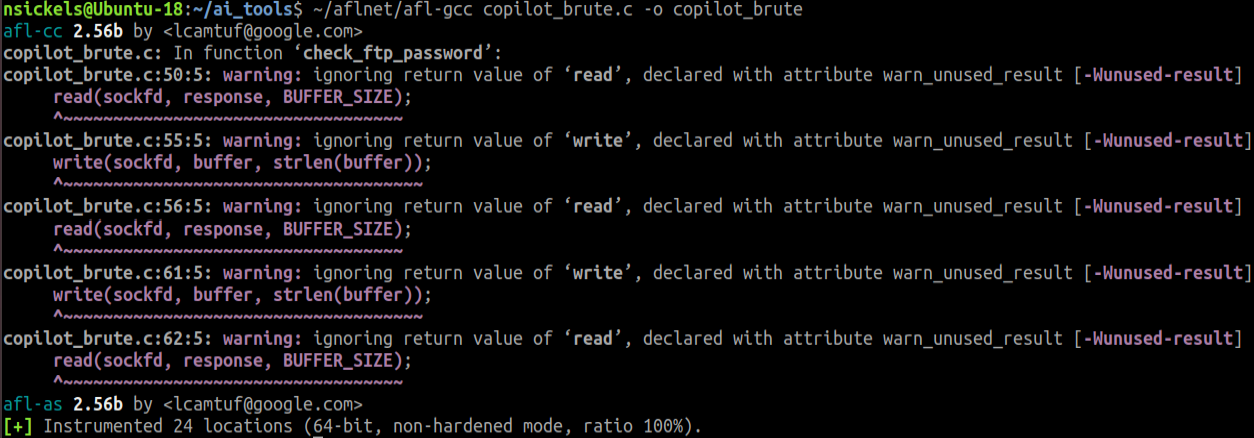
  
 *Copilot Phind*

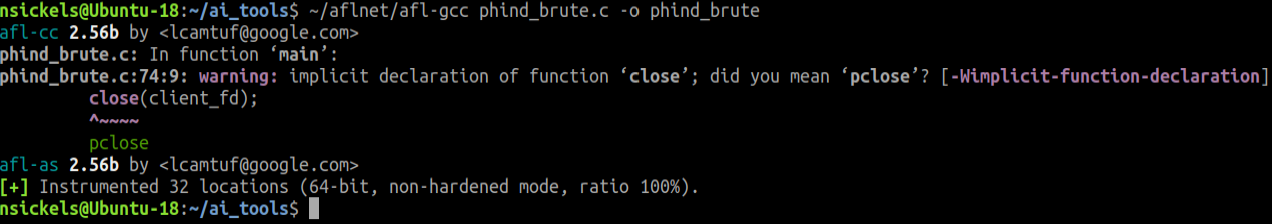
* Generated a multi-threaded banner-grabbing attack tool from the GitHub Copilot & Phind models.

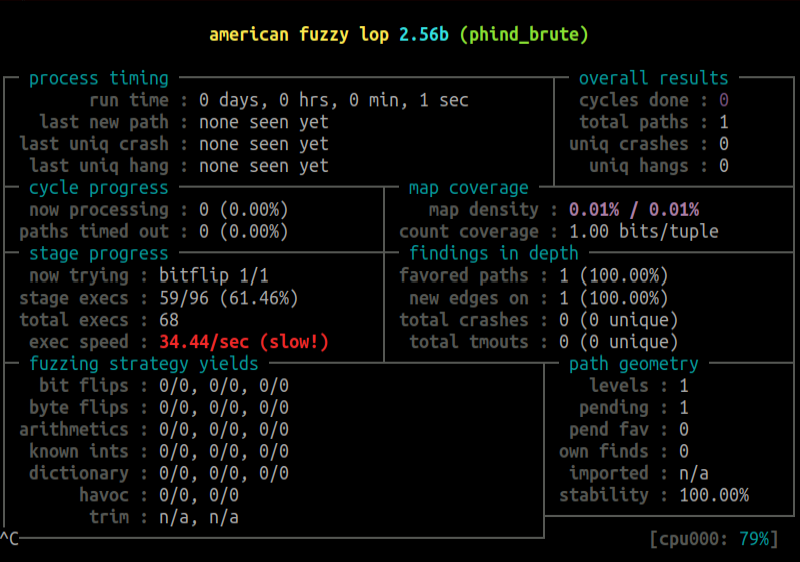
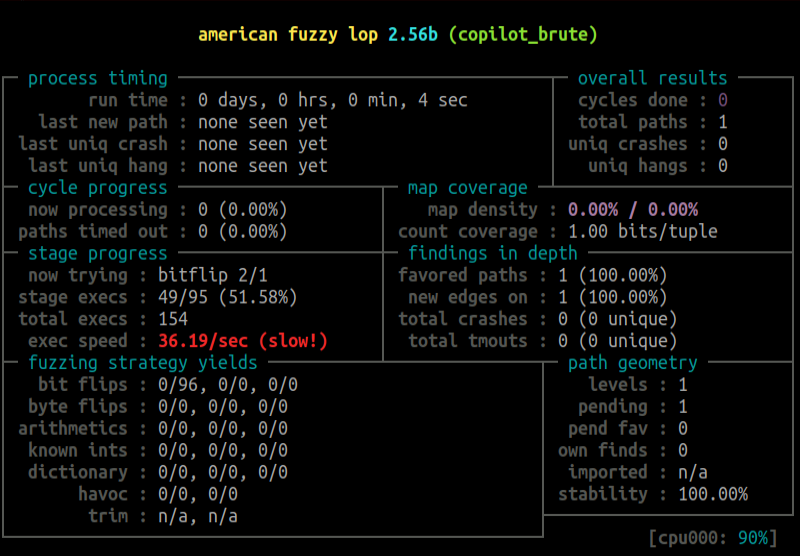


*Copilot Phind*

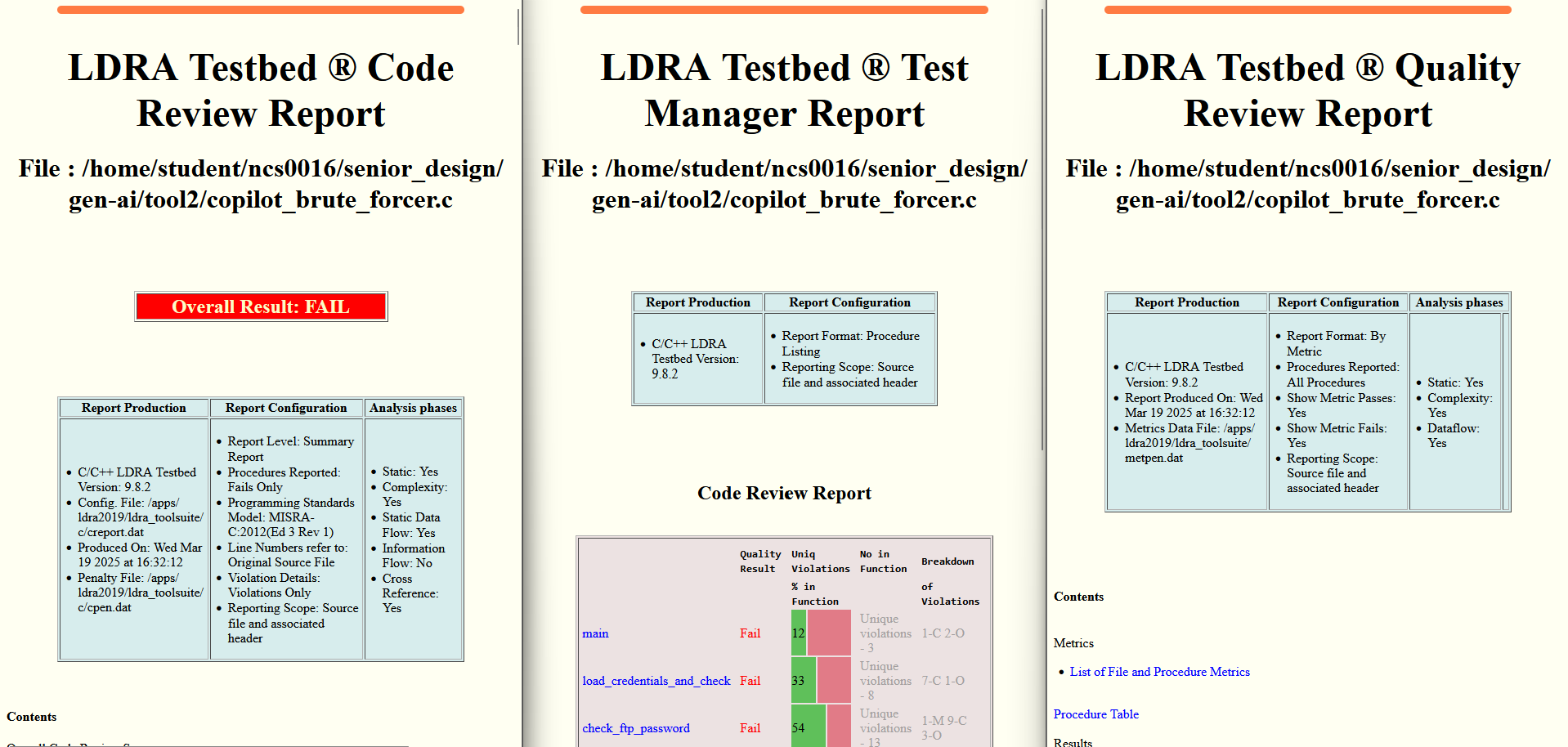
* Compiled the GitHub Copilot & Phind Brute Force attack tools with afl-gcc & applied fuzzing workflow on each.



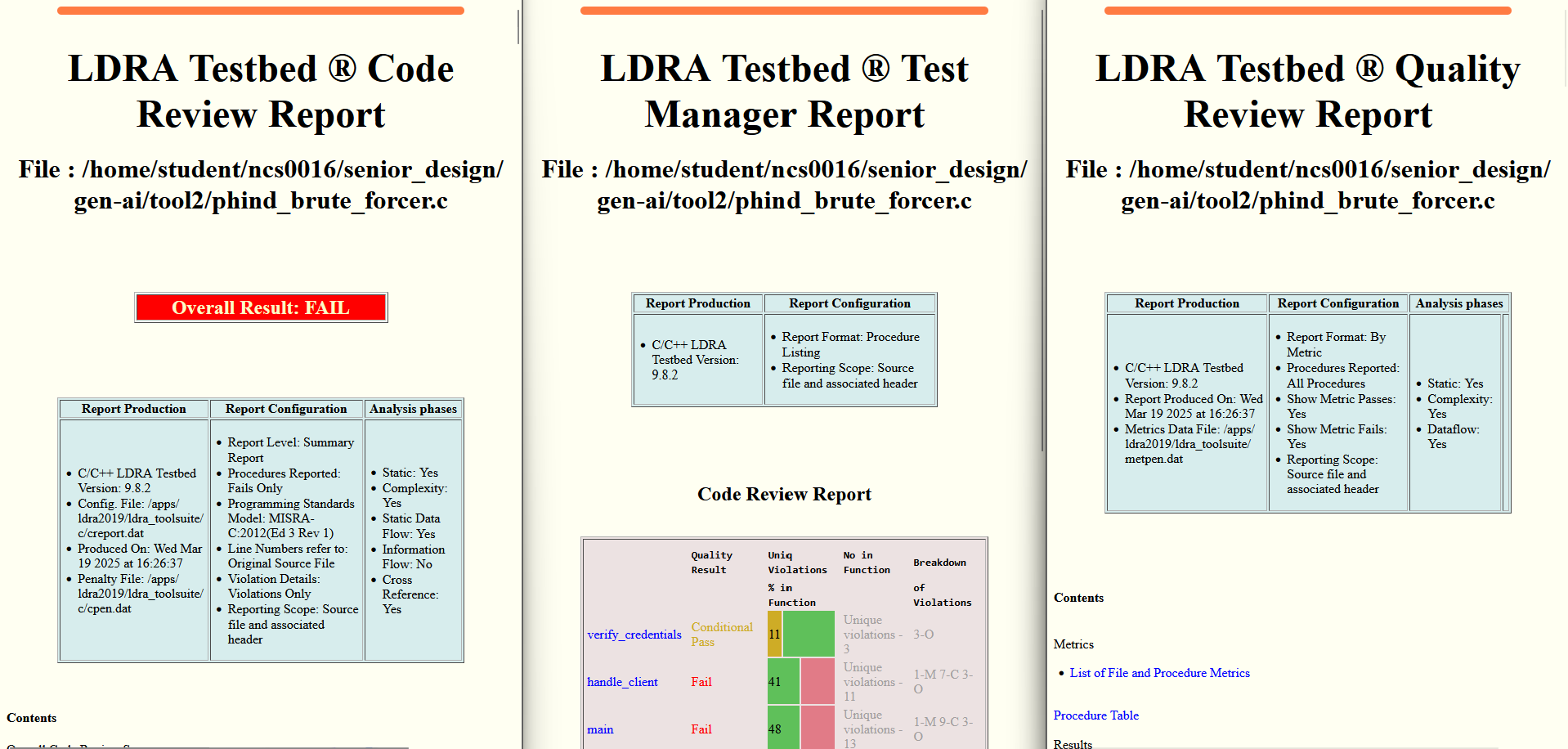




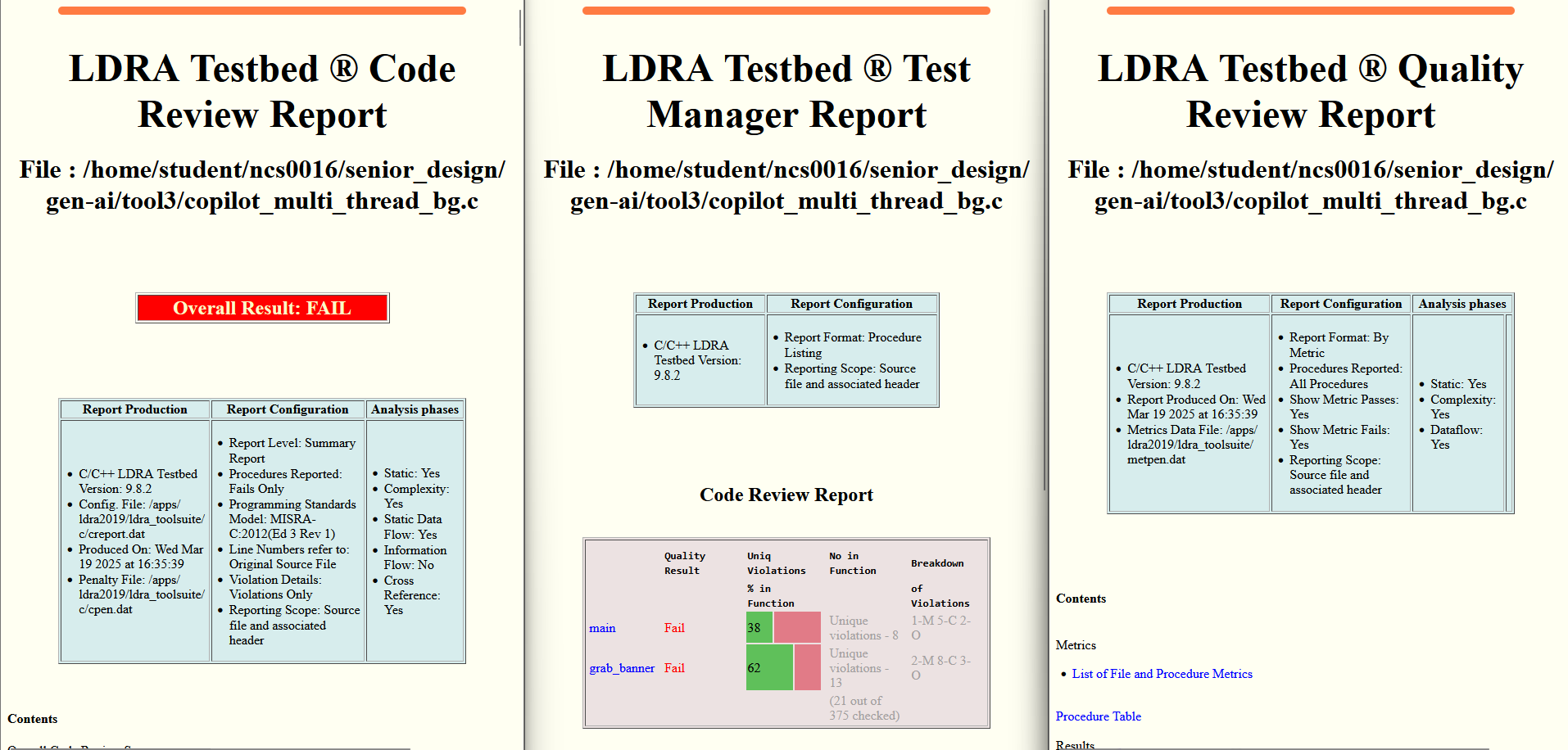
* Performed LDRA static analysis on the Brute Force and the Multi-threaded banner-grabbing attack tools from Phind and Copilot.



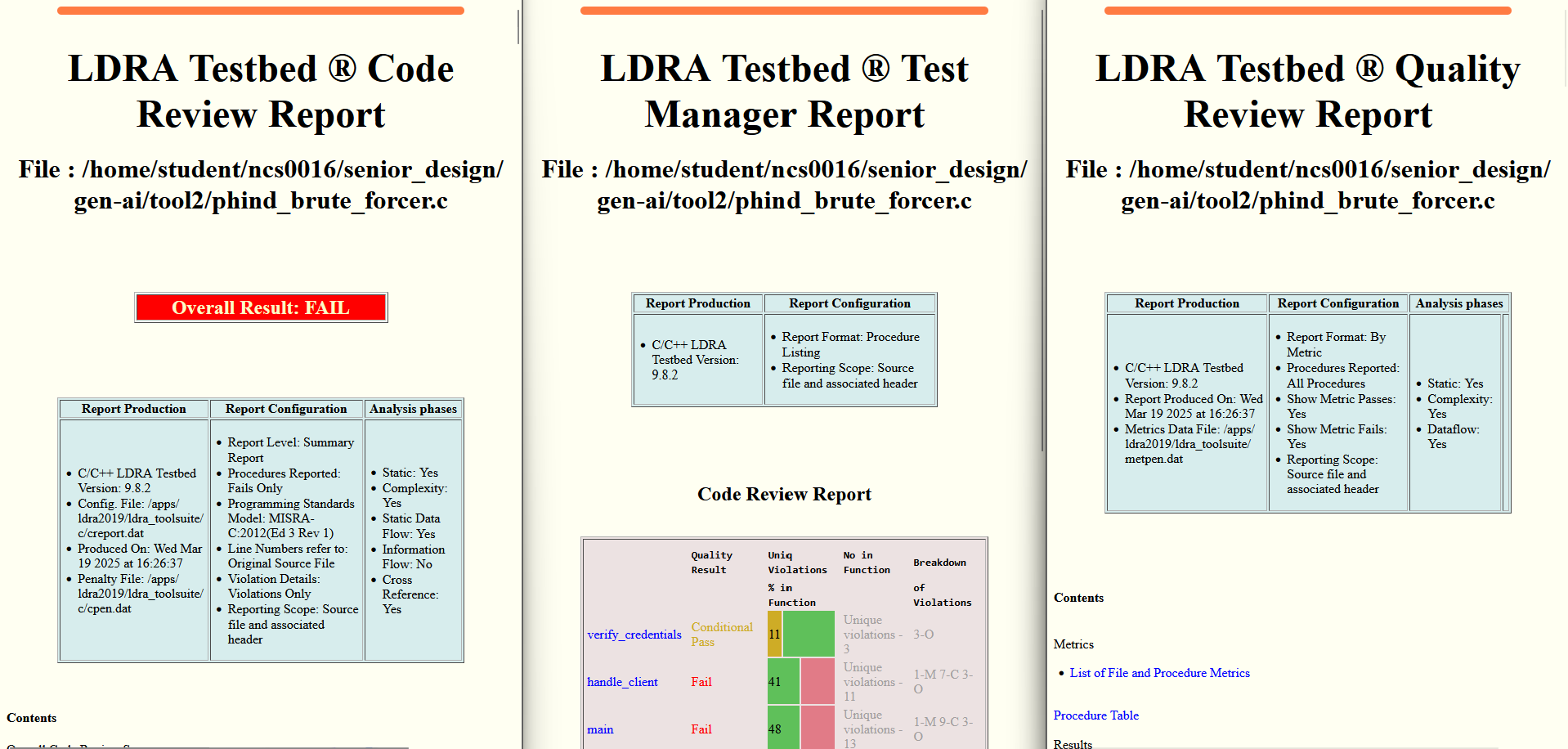
*Copilot Brute Force Tool*



*Phind Brute Force Tool*



*Copilot Multi-threaded Banner-grabber Tool*



*Phind Multi-threaded Banner-grabber Tool*

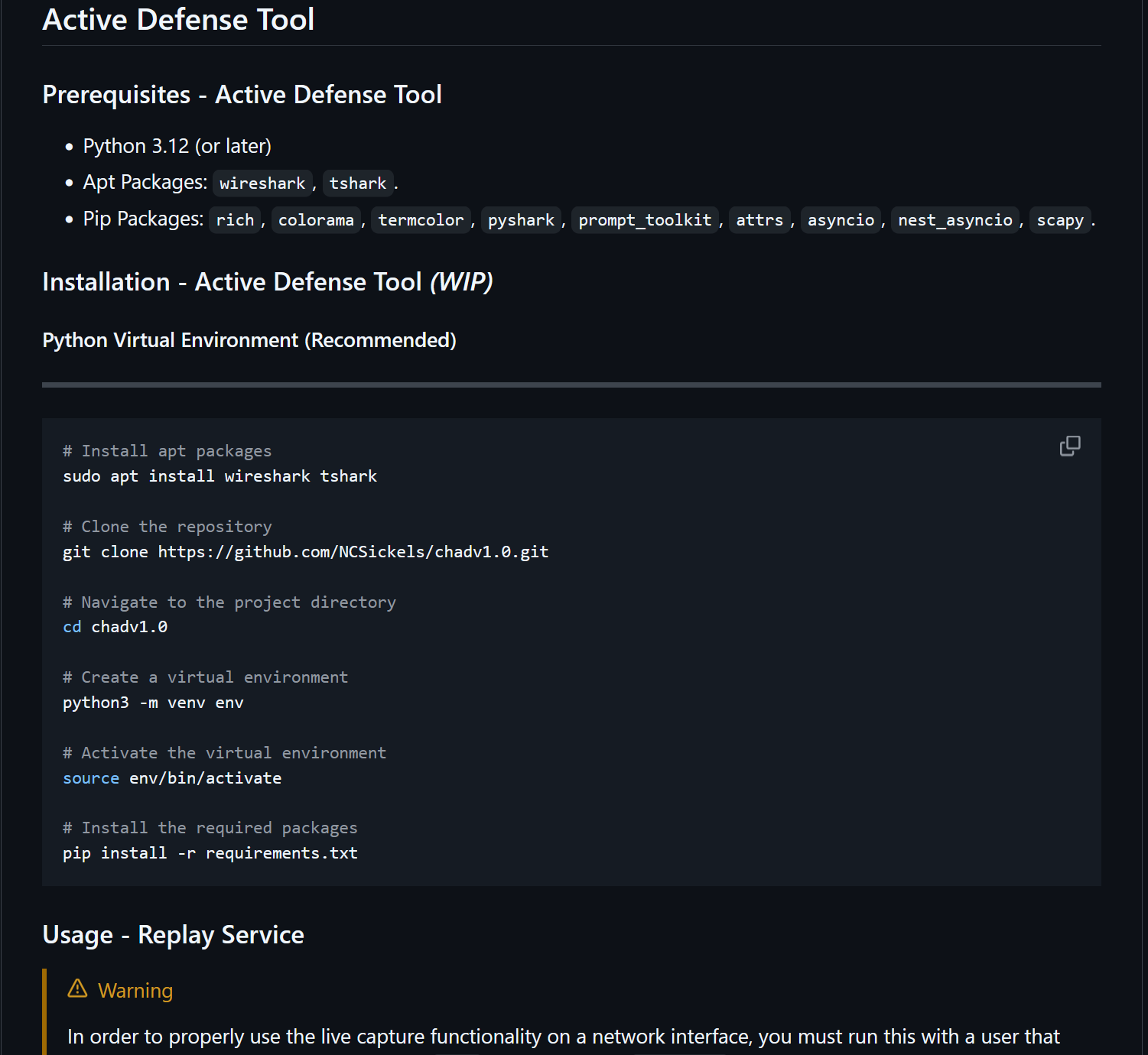
* Performed Valgrind dynamic memory leak testing on each tool.



* Performed FlawFinder static analysis on each of the generated attack tools.



* Updated README & User Guide with the Python active defense tool information.



### Mitigations & Contingency Plans

* N/A - No missed milestones.