Tool Selection Report

Charger Active Defense – G12

Noah Sickels, Adam Brannon, William Lochte

**Background**

Our senior design group is the second team working on the Charger Active Defense project. One of the main differences between our team and the first team was that we needed to choose tools that were different from those they selected, specifically Nmap and Hydra. Using the tool listing provided on Kali, we identified several potential candidates that met two key requirements: they must be written in C or C++ and have network functionality. We found six possible tools to screen for use from these two requirements, as shown in Table 1 below.

Table 1: Attack Tool Candidates

|  |  |  |  |
| --- | --- | --- | --- |
| Tool | Version | Last Update | Source % C |
| Medusa | 2.2 | November 14, 2022 | 70.7% |
| Masscan | 1.3.2 | December 3, 2022 | 99.8% |
| Reaver | 1.6.6 | September 1, 2023 | 99.4% |
| Aircrack-ng | 1.7 | August 24, 2024 | 71.9% |
| Yersinia | 0.8.2 | November 13, 2022 | 95.9% |
| Netdiscover | 0.10 | October 29, 2022 | 99.0% |

After selecting the attack tool candidates, we performed an initial screening process. This process included LDRA static analysis testing on the tool's source code files, Valgrind memory leak testing on the compiled binaries, and open-source intelligence gathering on the tools' repository "Issues" pages.

**Note:** In our proposal, we included a table that outlined "estimated importance," "estimated likelihood of vulnerability," and "priority." We derived these values from the most recent activity on their respective repository pages. However, upon closer examination, we found that much of the recent activity for many of these tools involved styling, formatting, or updating licensing information, which did not accurately reflect the tools' actual importance.

Based on our initial screening, we have provided a more accurate version of this table, as shown in Table 2 below. While the LDRA Quality Review results may indicate higher scores for tools with lower priority (primarily Netdiscover), this assessment was also influenced by the specific nature of each tool, as explained further below.

Table 2: Attack Tool Candidate Priority

|  |  |  |
| --- | --- | --- |
| Tool | LDRA Results (“All Metrics”) | Priority |
| Medusa | 85 | 1 |
| Masscan | 83 | 2 |
| Reaver | 80 | 3 |
| Netdiscover | 94 | 3 |
| Aircrack-ng | 82 | 4 |
| Yersinia | 74 | 5 |

**Selection Rationale**

**Medusa**

Medusa is a modular login brute-forcer, similar to Hydra, designed to support a wide range of services that utilize remote authentication [2]. It includes numerous custom libraries and examples of modules for various services and has native support for over thirty-two protocols. Because each protocol's authentication method is managed differently, each module is defined in separate ".mod" files.

During our initial testing with Valgrind, we observed significant differences in memory leaks across the modules. For instance, the SSH module showed a memory loss of 3,256 bytes, while the PostgreSQL module reported a loss of 38,342 bytes. Given Medusa's extensive support for various protocols, its capability to create custom modules for those not natively supported, and its similar nature to the first team's tool, Hydra, we have chosen it as one of our primary attack tools.

**Masscan**

Masscan is a high-speed mass IP port scanner capable of transmitting up to ten million packets per second from a single machine [3]. It is highly regarded in the industry and among professionals and is considered one of the first port scanners still actively maintained. Its core functionality is similar to that of Nmap, and it supports many of Nmap's arguments, along with several integrated libraries.

One of Masscan's key features is its ability to perform banner grabbing on remote targets. Although it is often used with ICMP echo requests, its capability for banner grabbing shows that it can interact with nearly any port or protocol provided. This versatility allows for extensive testing of a wide range of services.

Additionally, Masscan can accept PCAP files as input, enabling scanning and enumeration. This feature allows us to use Masscan alongside other tools to generate fuzzed network traffic.

**Reaver**

Reaver is a brute-force attack tool specifically designed to target access points with Wi-Fi Protected Setup (WPS) enabled [4]. We prioritized Reaver over other tools because it includes a built-in feature for analyzing PCAP files for WPS PINs. This functionality allows us to use Reaver in conjunction with other tools to generate fuzzed network traffic in PCAP file format, which enables thorough testing of the tool with the ingested traffic.

However, using Reaver beyond custom-generated PCAP files requires special configuration of the Metasploitable2 target to properly mimic an access point or the use of a hardware access point for testing. Due to this limitation, we ultimately decided not to select Reaver as one of our tools.

**Netdiscover**

Netdiscover is an active/passive address reconnaissance tool, mainly developed for wireless networks without a DHCP server [5]. It is foundationally similar in functionality to Masscan but less widely used.

**Aircrack-ng**

Aircrack-ng is a comprehensive suite of tools designed to evaluate Wi-Fi network security. It focuses on several key areas: monitoring, attacking, testing, and cracking wireless networks [6]. While the suite is well-regarded and widely used in the industry, many of its tools are not accessible from the attack surface.

For example, components like airmon-ng are specifically used to configure network adapters on a local host for monitoring mode, and they do not interact with external entities. Similarly, tools like aircrack-ng only apply on a local host and cannot be used against remote targets, as specified by the project requirements. Due to these limitations, we have placed Aircrack-ng lower in priority.

**Yersinia**

Yersinia is a framework designed for executing layer two attacks that exploit vulnerabilities in various network protocols, including the Dynamic Host Configuration Protocol (DHCP), 802.1Q, 802.1X, and VLAN Trunking Protocol (VTP), among others [8]. However, we decided to assign it the lowest priority due to Yersinia's outdated nature and the increased complexity of fuzz testing layer two-specific protocol stacks.

**References**

[1] Kali, “Kali Tools | Kali Linux Tools,” *Kali Linux*. https://www.kali.org/tools/

‌

[2] “Debian Security Tools Packaging Team / medusa · GitLab,” *GitLab*, 2018. https://salsa.debian.org/pkg-security-team/medusa (accessed Oct. 27, 2024).

‌ [3] R. D. Graham, “robertdavidgraham/masscan,” *GitHub*, Oct. 31, 2020. <https://github.com/robertdavidgraham/masscan>

[4] “Debian Security Tools Packaging Team / reaver · GitLab,” *GitLab*, 2023. https://salsa.debian.org/pkg-security-team/reaver (accessed Oct. 27, 2024).

[5] “Debian / netdiscover · GitLab,” *GitLab*, 2018. https://salsa.debian.org/debian/netdiscover (accessed Oct. 27, 2024).

‌ [6] Aircrack-ng, “Aircrack-ng,” *Aircrack-ng.org*, 2009. <https://www.aircrack-ng.org/>

[7] Aircrack-ng, “Aircrack-ng,” *GitHub*, Apr. 20, 2022. <https://github.com/aircrack-ng/aircrack-ng>

‌ [8] tomac, “tomac/yersinia,” *GitHub*, Apr. 18, 2019. https://github.com/tomac/yersinia

‌