Threat Model

Charger Active Defense – G12

Noah Sickels, Adam Brannon, William Lochte

**System Model Overview**

**Components**

* Kali Linux 2024.4 Virtual Machine (Attacker)
  + Runs AI-generated attack tools.
  + Runs Masscan attack tool.
  + Runs AFLnet wrapper on each attack tool for network protocol fuzzing.
* Metasploitable2 Virtual Machine (Victim)
  + Hosts vulnerable services.
  + Runs Python service to monitor network traffic and replay active defense responses.
* Network Interface
  + Facilitates the communication between the attacker and the victim.
* Active Defense Response Database
  + Storage of all found crashes or hangs within attacking applications.

**Data Flow**

1. **Attack Initiation**: Kali VM sends traffic to the Metasploitable2 VM.
2. **Response Handling**: The ChAD service on the victim VM sends back active defense responses to the attacking VM.
3. **AFLnet Fuzzing**: AFLnet modifies the attack patterns to detect crashes or hangs within the attacking application.
4. **Replay Mechanism**: The ChAD service replays the active defense responses to the attack tools once network traffic is detected from the specific network interface.
5. **Analysis**: Logs for incoming and outgoing network traffic, sent active defense responses, and found crashes or hangs from AFLnet are collected to evaluate successes.

**A diagram of a diagram

AI-generated content may be incorrect.**

Figure 1: ChAD Threat Model Diagram

**STRIDE Model**

Table 1: STRIDE Threat Analysis

|  |  |  |
| --- | --- | --- |
| Threat | Description | Mitigation |
| Spoofing | An Attacker may impersonate the Kali VM or victim VM to inject false data. | Mutual authentication, network isolation, cryptographic signing of attack payloads. |
| Tampering | Malicious actors could alter the attack or replay payloads. | Integrity verification, logging and monitoring payload transformations. |
| Repudiation | Attackers may deny active defense response payloads. | Comprehensive logging and timestamping of events. |
| Information Disclosure | Sensitive information could be leaked through attack tools or active defense responses (passwords, usernames, hostname, IP addresses) | Restrict logging of sensitive data in the controlled environment. |
| Denial of Service (DoS) | Overloading the replay service or attack tools with excessive traffic could cause failure and downtime. | Rate limiting, monitoring resource usage, segmentation of test cases. |
| Elevation of Privilege | Exploiting vulnerabilities with the attack tools to escalate privileges. | Running tests in sandboxed environments with least privilege. |

**Attack Tree**

**Goal:** Compromise the Integrity or Availability of Attack Tools

1. **Exploit AFLnet to Cause Unintended Behavior**
2. Inject malformed responses to crash the fuzzing workflow.
3. Use crafted responses to trigger crashes or hangs.
4. **Overload Attack Tools via Replay Mechanism**
5. Send active defense response payloads to cause crash or hang within the attacking application.
6. Replay malformed responses to trigger tool exceptions.
7. **Interfere with Metsploitable2’s Response Handling**
8. Alter attack payloads mid-transit.
9. Poison replay service logs with false data.
10. **Trigger Unhandled Exceptions in AI-Generated Tools**
11. Identify input validation gaps.
12. Inject adversarial attack patterns.

**Mitigations**

**Defensive Measures**:

* **Input Validation**:
* **Rate Limiting**:
* **Logging & Monitoring**: Log anomalous behaviors and unexpected failures.
* **Sandboxing**: Isolate attack tools and fuzzers to prevent full system crashes.
* **Network Isolation**: Isolate attacker and victim virtual machines on a separated network, disconnected to the internet or other critical assets.
* **Integrity Checks**: Ensure active defense responses maintain integrity while at rest and in-transit.

**DREAD Analysis**

Table 2: DREAD Analysis Scores

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Threat Scenario | Damage | Reproducibility | Exploitability | Affected Users | Discoverability | Total Risk |
| Crash attack tools via malformed responses. | 7 | 9 | 8 | 6 | 8 | 38 (High) |
| Unhandled exception in attack tool. | 6 | 8 | 7 | 5 | 8 | 34 (Medium) |
| Denial of Service on replay service. | 8 | 9 | 6 | 5 | 7 | 35 (High) |
| Information leakage from attack tool. | 7 | 7 | 6 | 4 | 6 | 30 (Medium) |

**Test Cases**

**Crash Testing AFLnet & Attack Tools**

* Send specially crafted responses with invalid protocol or data formats.
* Replay responses at high speed to cause memory exhaustion.
* Introduce unexpected data types (e.g., Unicode, null bytes, overflows).

**Monitoring Replay Service Stability**

* Test with varying response rates to assess service resilience.
* Log replay accuracy and ensure payloads are delivered correctly.

**Analyzing Attack Tool Responses**

* Check if tools handle malformed responses gracefully.
* Detect if certain payloads consistently crash the attack tool.

**Security Hardening**

* Validate authentication mechanisms to prevent unauthorized modifications.
* Ensure data integrity when passing responses between systems.