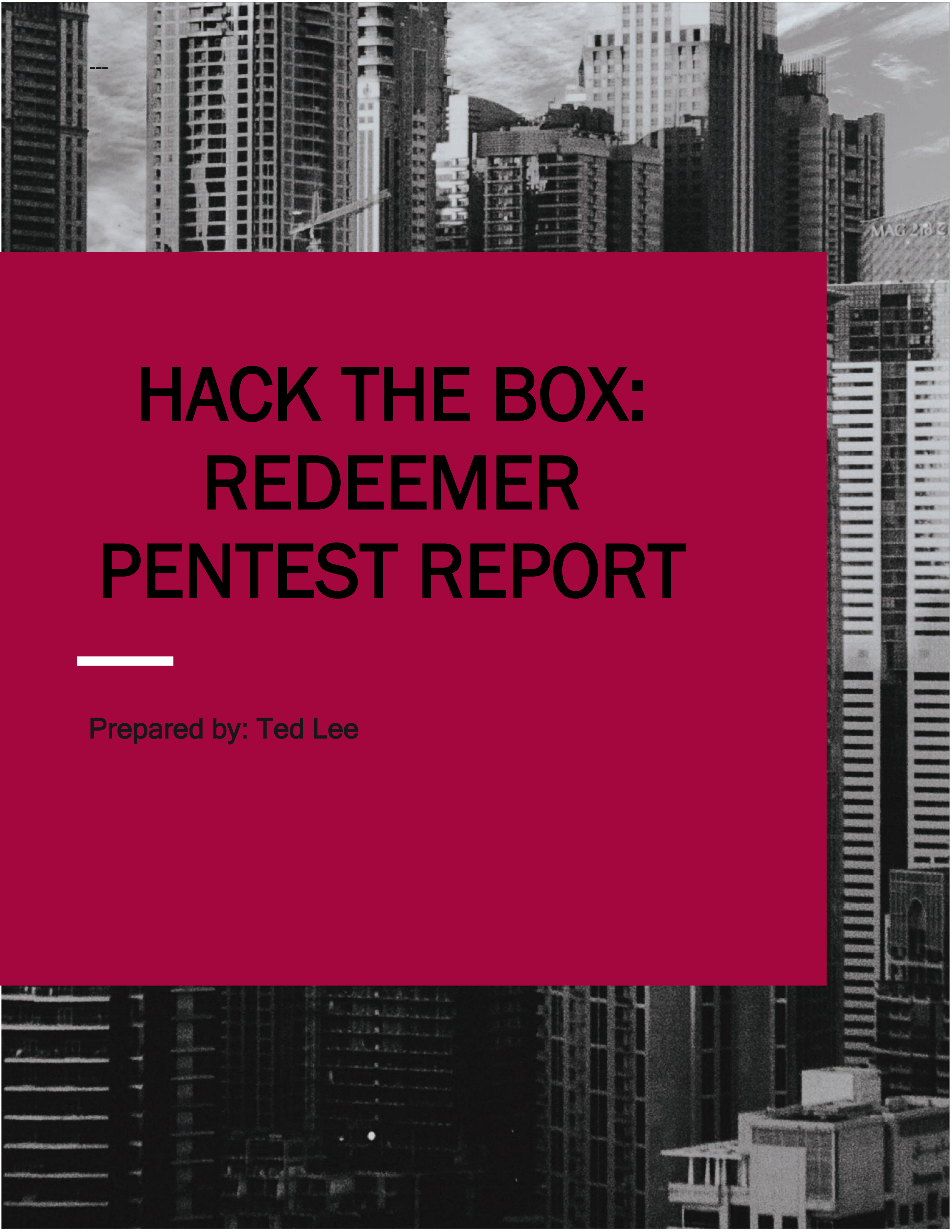



HACK THE BOX: REDEEMER PENTEST REPORT

Prepared by: Ted Lee

--

1 Executive Summary

The Computer Science (CS) department at The University of Texas at San Antonio (UTSA) offers a class on penetration testing (pentesting). For the students to obtain practical, hands-on training to improve their technical skills regarding pentesting, they need to complete the free available lab environments that are provided in 3 different tiers of the Starting Point learning track on HackTheBox (HTB). Writing a report is an essential skill in relation to pentesting and writing an assessment report is one of the requirements for this class.

The system chosen for this demo report is from Tier 0 of Starting Point and is the fourth box called Redeemer.

The assessment team found several strengths and weaknesses on the Redeemer system. Some of the strengths that were inferred from the system were the use of a Remote Dictionary Server (Redis), limited exposure, and awareness of least privilege. When Redis is configured properly, it does not expose unnecessary ports and has a minimal attack surface. However, for educational purposes, the team was able to discover multiple vulnerabilities that could be exploited as an example if Redis was incorrectly configured. Out of the several weaknesses highlighted on Section 5, the team recommends HTB to bind Redis to "localhost" or "127.0.0.1" and then expose access through SSH port forwarding or a web proxy. This makes the environment closer to real-world scenarios which is what the class wants to impress on its students.

Lastly, the assessment team would like to thank HackTheBox and the entire team for providing the lab environment needed for testing purposes.

Contents

Table of Figures/Tables

2 Introduction

This section includes a quick background on the purpose for the penetration test. Additionally, it provides the scope of the assessment and the organization to which each section belongs to.

2.1 Background

The Computer Science (CS) department at The University of Texas at San Antonio (UTSA) offers a class on penetration testing (pentesting). For the students to obtain practical, hands-on training to improve their technical skills regarding pentesting, they need to complete the free available lab environments that are provided in 3 different tiers of the Starting Point learning track on HackTheBox (HTB). Writing a report is an essential skill in relation to pentesting and writing an assessment report is one of the requirements for this class.

The system chosen for this demo report is from Tier 0 of Starting Point and is the fourth box called Redeemer.

2.2 Scope

For this assessment, the limitations outlined as the Redeemer system and the only IP address that was provided for this test. The other machines provided by HTB in the other tiers that make up Starting Point or any other machines or environments that comprise HTB were not included in this assessment.

2.3 Report Organization

The rest of this report is organized as follows: Section 3 provides an overview of the Redeemer system. Sections 4 and 5 describe the methodology used by the team and the activities they conducted. The assessment results are found in Section 6, with conclusions and recommended follow-on activities in Section 7. Appendices include additional information related to the assessment.

3 System Overview

The purpose of the Redeemer lab is to provide an understanding of databases and the importance of securing them as they are a gateway of communication between businesses and their clientele regarding their information such as transactions, inventory, or personal information.

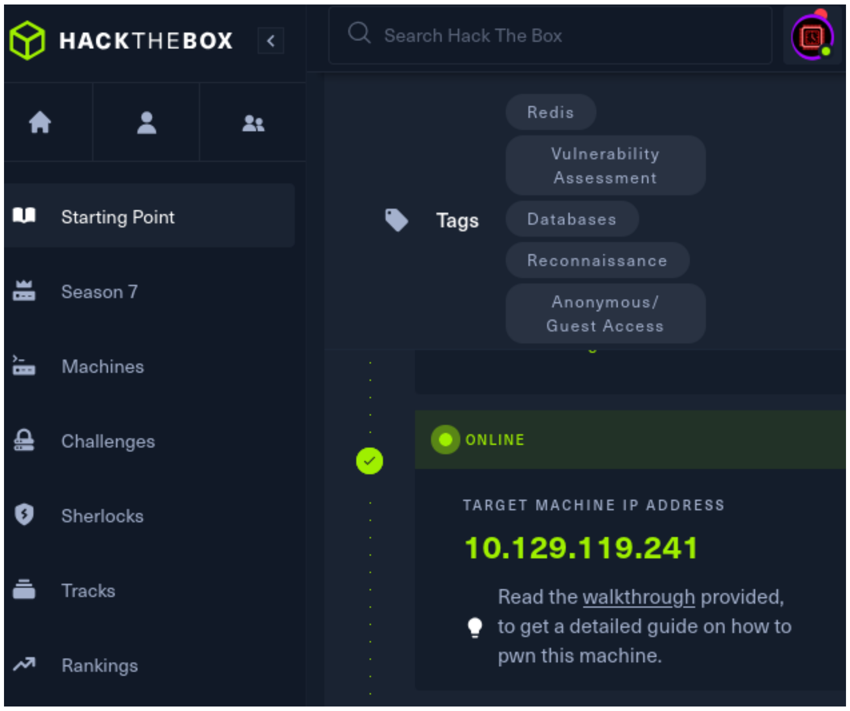
Figure 1: Logo for the Redeemer System

3.1 Target System: Redeemer

The Redeemer System is located on Tier 0 of the HTB Starting Point network. This network is comprised of isolated IP addresses that prevent anyone else from connecting to them. To connect to Redeemer, a user must either use Pwnbox, which is a virtual machine (VM) that is provided by HTB, or OpenVPN to connect to the IP address.

For multiple users to be able to utilize the same VM from Redeemer, the IP address created by the user is unique so there is no accidental crossover.

The IP address created for this test was: 10.129.119.241



TARGET MACHINE IP ADDRESS

10.129.119.241

Read the walkthrough provided, to get a detailed guide on how to pwn this machine.

Figure 2: Image of IP address provided for Redeemer from HTB

3.2 Network Exposure & Service Footprint

During the enumeration period, the user first pings the target IP address to verify the connectivity and availability of the target. We then used Nmap, which checks the Top 1000 ports, and found only port 6379 open. This port was used to service Redis and is a TCP port.

Details of the enumeration period and the scans can be found in Section 5.1. The following table summarizes the characteristics of port 6379.

Table 1: Result of Nmap Scan

Port	Protocol	State	Service	Version
6379	TCP	Open	Redis	Redis key-value store 5.0.7

3.3 Other System Information

Some additional characteristics that were revealed by the Nmap scan are as follows:

- The host responded to ping (ICMP Echo Requests) in 1002ms with 0% loss
- Using the nmap scan line provided by the official walkthrough took too much time as the -p- option scans all ports available so –min-rate 5000 was added to mitigate the issue
- The scan took a total of 47.4 seconds

4 Assessment Methodology

The team used the Penetration Testing Execution Standard (PTES) methodology located at: http://www.pentest-standard.org/index.php?Main_Page . This methodology is an industry, standard practice that ensures a consistent and comprehensive approach to an assessment.

The PTES methodology is broken up into 7 parts:

- Pre-Engagement Interactions
- Intelligence Gathering
- Threat Modeling
- Vulnerability Analysis
- Exploitation
- Post-Exploitation
- Reporting

In the pre-engagement interactions section, the scope and the rules of engagement were already predefined as this is a controlled lab environment. The assessment was limited to a single target host with known objectives which simulate a black-box scenario. In real-world scenarios, staying inside the scope and following the rules of engagement are crucial as going outside of the scope or failing to follow the rules of engagement could lead to serious legal issues.

Initial reconnaissance was performed using the Nmap scan with the -p- option to scan all ports available. As mentioned above, the tag –min-rate 5000 was added as scanning all available ports took too much time and was used to mitigate this issue. One single port was identified as open because of the Nmap scan which was port 6379 using the TCP protocol. The service that was identified from port 6379 was Redis which is an open-source advanced NoSQL key-value data store used as a database, cache, and message broker. The database is stored in the server's RAM (in-memory) to enable fast data access. Redis also writes the contents of the database to disk at varying intervals to persist it as a backup, in case of failure. Redis runs as a server-side software so its main function is in its server component. The server listens for connections from clients, programmatically or through the command-line interface (CLI). The command-line interface is a tool that gives a user complete access to Redis's data and its functionalities which are utilized in the upcoming process.

Based off the information collected from the intelligence gathering, a threat model was established:

- Target: Redis service/port 6379

- Attack Vector: Unauthorized access due to lack of authentication and open binding using the redis-cli tool
- Impact Potential: Data exfiltration, unauthorized command execution, system compromise, release of sensitive, personal information

The risk was determined to be high due to the unauthenticated and network-exposed nature of the Redis instance.

There are three main vulnerabilities that were identified as a result of using the redis-cli tool:

- The service had no authentication (no password set)
- Sensitive data (including the main flag) was stored in plaintext
- The service accepted and responded to administrative controls

These factors confirmed the presence of a critical security misconfiguration vulnerability rather than a software vulnerability. Properly configured redis software should focus on network isolation, authentication, and encryption. This includes binding redis to a specific IP address, configuring a password, enabling TLS encryption, and renaming or disabling harmful commands.

Using standard redis-cli commands such as “redis-cli -h {target-IP}” we are able to connect to the target system. Once we are connected to the target, we can gather more information about the system using “info” where we learn the version and more importantly, the keyspace. From accessing the additional information about the system, we find that this service has 1 database with 4 keys. We then use “select 0” to select the database and “keys *” to list all the keys present in the database. Finally, we can use the command “get” to obtain the values of all of the keys in the database including the flag key which is the main objective of the lab.

Post-exploitation activities were limited to the nature of the lab environment. In real-world scenarios, this level of access could be leveraged for:

- Inject cron jobs or SSH keys
- Modify application behavior
- Elevate privileges or pivot within the internal network

In this lab, the goal was met by retrieving the target flag, simulating a data breach.

Findings were documented in this report, with detailed explanations of the discovered misconfigurations, associated risks, and mitigation recommendations. The report structure aligns with professional standards, providing technical and executive-level insights.

This summary was written with the help of ChatGPT.

5 Assessment Activities

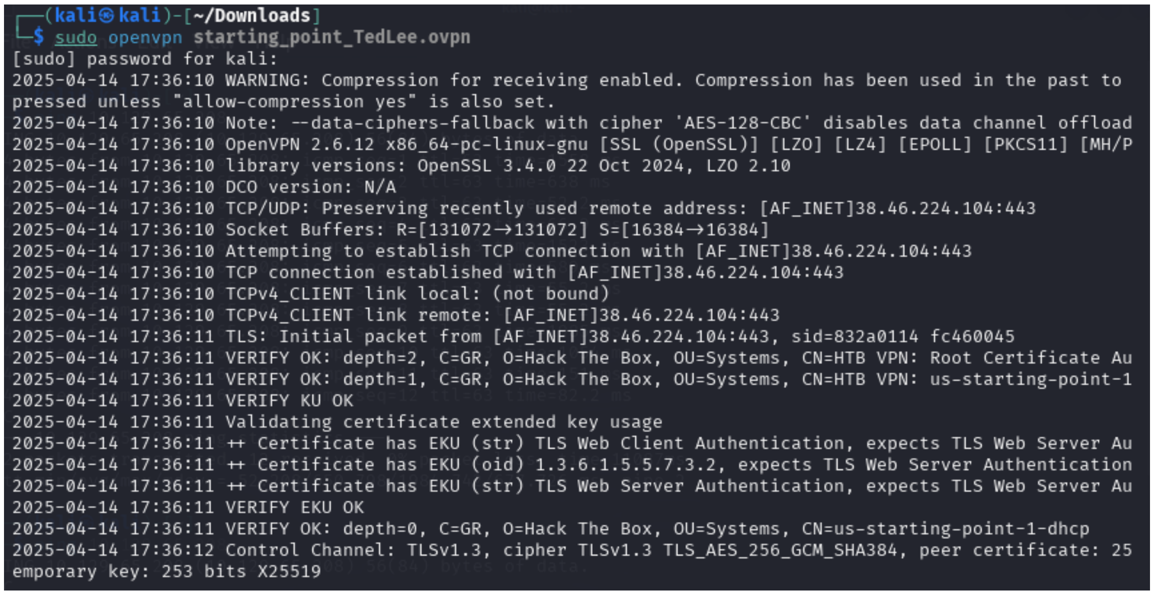
This section covers the activities performed by the assessment team at a high level. However, despite the simplicity of Redeemer as it is an educational and controlled environment, the assessment team believes that there is enough detail and evidence to recreate the assessment.

All guidance and instructions were provided by the official writeup on HTB.

5.1 Network Connection

The team first connects to the HTB network using either Pwnbox or OpenVPN on their Kali Linux virtual machine. For this assessment, the assessment team decided to connect to the OpenVPN client in order to complete this test.

One alibi to using the OpenVPN client is that the user must use the command “sudo” in order to allow OpenVPN to modify the network interfaces. Without using the “sudo” command, this may cause a failure to properly connect to the VPN.



```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo openvpn starting_point_TedLee.ovpn
[sudo] password for kali:
2025-04-14 17:36:10 WARNING: Compression for receiving enabled. Compression has been used in the past to
pressed unless "allow-compression yes" is also set.
2025-04-14 17:36:10 Note: --data-ciphers-fallback with cipher 'AES-128-CBC' disables data channel offload
2025-04-14 17:36:10 OpenVPN 2.6.12 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/P
2025-04-14 17:36:10 library versions: OpenSSL 3.4.0 22 Oct 2024, LZO 2.10
2025-04-14 17:36:10 DCO version: N/A
2025-04-14 17:36:10 TCP/UDP: Preserving recently used remote address: [AF_INET]38.46.224.104:443
2025-04-14 17:36:10 Socket Buffers: R=[131072→131072] S=[16384→16384]
2025-04-14 17:36:10 Attempting to establish TCP connection with [AF_INET]38.46.224.104:443
2025-04-14 17:36:10 TCP connection established with [AF_INET]38.46.224.104:443
2025-04-14 17:36:10 TCPv4_CLIENT link local: (not bound)
2025-04-14 17:36:10 TCPv4_CLIENT link remote: [AF_INET]38.46.224.104:443
2025-04-14 17:36:11 TLS: Initial packet from [AF_INET]38.46.224.104:443, sid=832a0114 fc460045
2025-04-14 17:36:11 VERIFY OK: depth=2, C=GR, O=Hack The Box, OU=Systems, CN=HTB VPN: Root Certificate Au
2025-04-14 17:36:11 VERIFY OK: depth=1, C=GR, O=Hack The Box, OU=Systems, CN=HTB VPN: us-starting-point-1
2025-04-14 17:36:11 VERIFY KU OK
2025-04-14 17:36:11 Validating certificate extended key usage
2025-04-14 17:36:11 ++ Certificate has EKU (str) TLS Web Client Authentication, expects TLS Web Server Au
2025-04-14 17:36:11 ++ Certificate has EKU (oid) 1.3.6.1.5.5.7.3.2, expects TLS Web Server Authentication
2025-04-14 17:36:11 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server Au
2025-04-14 17:36:11 VERIFY EKU OK
2025-04-14 17:36:11 VERIFY OK: depth=0, C=GR, O=Hack The Box, OU=Systems, CN=us-starting-point-1-dhcp
2025-04-14 17:36:12 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, peer certificate: 25
emporary key: 253 bits X25519
```

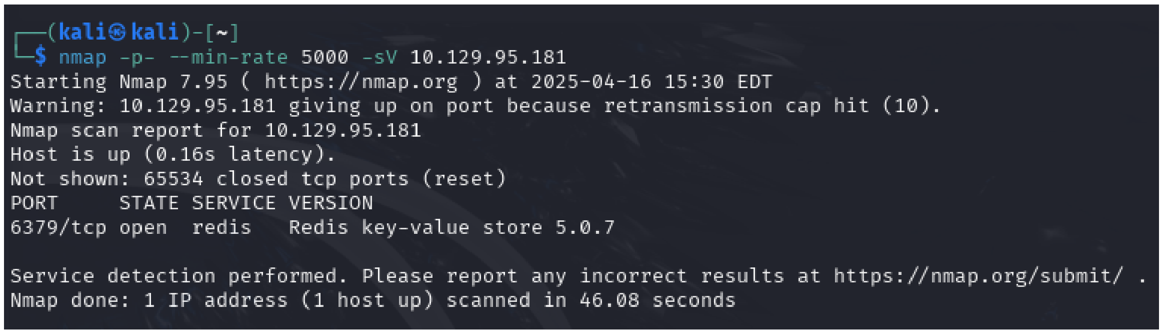
Figure 3: Image of connection to HTB VPN

5.2 Enumeration

After the team was able to connect to the VPN, we scanned the Redeemer VM using the following command:

“nmap -p- --min-rate 5000 -sV 10.129.65.208”

The only open port that was found by the scan was port 6379 which uses the TCP protocol. We found the service that was being used by this port was Redis which is the main objective in retrieving the target flag.



```
┌──(kali㉿kali)-[~]
└─$ nmap -p- --min-rate 5000 -sV 10.129.95.181
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-16 15:30 EDT
Warning: 10.129.95.181 giving up on port because retransmission cap hit (10).
Nmap scan report for 10.129.95.181
Host is up (0.16s latency).
Not shown: 65534 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
6379/tcp open  redis   Redis key-value store 5.0.7

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 46.08 seconds
```

Figure 4: Result of Nmap Scan

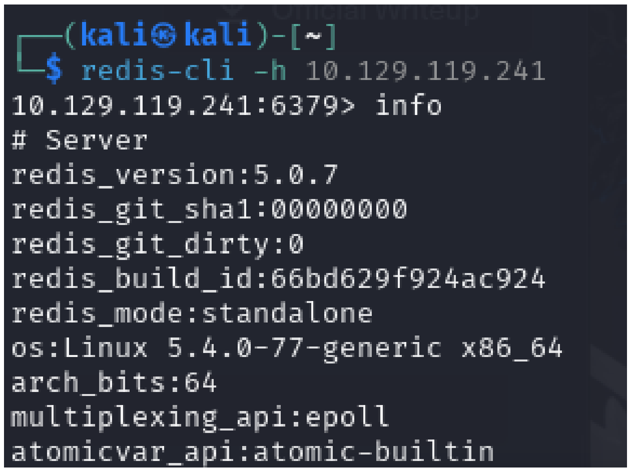
5.3 Redis Service Inspection

Once the target was identified with Redis being the sole service that was exposed, we can exploit this vulnerability with the command-line interface tool or redis-cli. We first installed the redis-cli tool using the command: “sudo apt install redis-tools”.

After installing the necessary tools, we can use the command: “redis-cli -help” in order to determine which tools to use next. To connect to the target host, we can use the command: “redis-cli -h {target_IP}” and we were able to connect to the host via IP address and port.

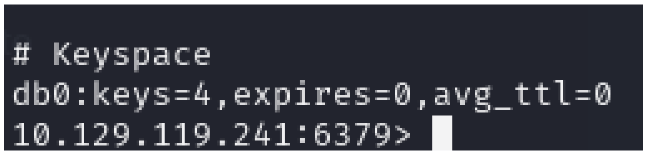
5.4 Flag Extraction & Post-Exploitation

Once the team was connected to the target host, we were able to use the command: “info” in order to gather further information about the server. The full Redis server summary can be found in Appendix A: Full Summary of Redis Server



```
┌──(kali㉿kali)-[~]
└─$ redis-cli -h 10.129.119.241
10.129.119.241:6379> info
# Server
redis_version:5.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:66bd629f924ac924
redis_mode:standalone
os:Linux 5.4.0-77-generic x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
```

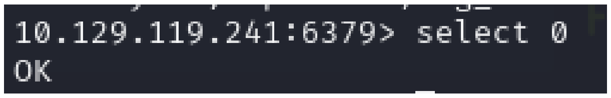
Figure 5: Connection to target host and server info



```
# Keyspace
db0:keys=4,expires=0,avg_ttl=0
10.129.119.241:6379>
```

Figure 6: Image of exposed database in Redis server

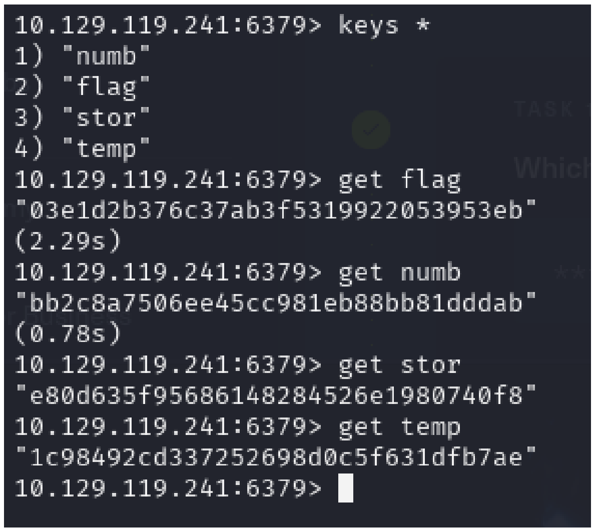
We identified that there is a database with the index “0” that contains 4 keys. We obtain the information by using the command: “select 0” to select the database, then we use the command: “keys *” in order to list out all of the keys in the database.



```
10.129.119.241:6379> select 0
OK
```

Figure 7: Selection of database 0

Once identifying all of the keys that were listed, the team used the “get” command in order to obtain the values for each key including the “flag” key which was our main objective.



```
10.129.119.241:6379> keys *
1) "numb"
2) "flag"
3) "stor"
4) "temp"
10.129.119.241:6379> get flag
"03e1d2b376c37ab3f5319922053953eb"
(2.29s)
10.129.119.241:6379> get numb
"bb2c8a7506ee45cc981eb88bb81dddab"
(0.78s)
10.129.119.241:6379> get stor
"e80d635f95686148284526e1980740f8"
10.129.119.241:6379> get temp
"1c98492cd337252698d0c5f631dfb7ae"
10.129.119.241:6379>
```

Figure 8: List of keys and their values

6 Assessment Results and Recommendations

This section describes in detail the findings that were obtained in Section 5. This includes the strengths, weaknesses, and observations made by the assessment team.

The assessment team identified several high impact vulnerabilities that could lead to unauthorized access and system compromise. While the system was in a controlled lab environment which reduced the impact of the weaknesses, the misconfiguration of the Redis server created a significant risk.

This section provides a summary of the most impactful weaknesses, which are labeled from low to critical based on risk level, highlights positive strengths of the system, and outlines any relevant observations that can include items outside the scope of the assessment.

In order to determine the risk level for each vulnerability, the team discussed the following questions:

- Is the vulnerability accessible remotely or only within the internal network?
- Does exploiting this vulnerability require authentication or can it be done anonymously?
- What kind of impact could successful exploitation have (e.g., data exfiltration, system control, lateral movement)?
- How critical is the affected system to business operations or data confidentiality?
- Are there existing controls (e.g., firewalls, monitoring, segmentation) that would mitigate or detect this attack?

The assessment team has listed recommendations based on the level of severity of the vulnerabilities with the primary focus of mitigation. However, this list is to be used as a guide and not a dictation as certain mitigations require different allocation of resources that the company or business may take priority in.

6.1 Weaknesses

(Critical) The Redis service was accessible without any form of authentication or access control.

Justification: The assessment team was able to gain control of the Redis service without any type of credentials. This opens up an entryway for attackers to exfiltrate any and all data that is stored in Redis, manipulate keys or values found in the databases, and even use administrative controls to change certain configurations. As Redis often contains sensitive, possibly confidential, data, this misconfiguration poses a critical threat to the company/business and their clients.

Mitigations: The team recommends setting the “requirepass” directive in redis.conf to enforce authentication. For Redis 6 and newer, implement Access Control Lists (ACLs) to restrict command usage per user and outline select personnel that have different levels of access. For this system, the Redis version was found to be 5.0.7. Lastly, disable all dangerous, high-level commands if unnecessary (e.g., CONFIG, FLUSHALL, SAVE).

(High) The Redis instance was configured to accept connections from any network interface.

Justification: Exposing Redis to the public or untrusted internal networks increases the attack surface as the server was not binded to a specific or trusted network. Combined with the lack of authentication, this leads to a high-risk scenario that invites unauthorized users or enumeration by attackers.

Mitigation: The team recommends using the “bind” directive in redis.conf to restrict the service to 127.0.01 or another trusted subnet. Use a host-based firewall to allow only specific and approved IP ranges to connect. Consider using Redis behind a secure proxy or VPN.

(Medium) Redis communicates in plaintext by default, making it vulnerable to eavesdropping and interception.

Justification: Although encryption doesn’t prevent exploitation, the lack of TLS enables attackers on the same network to observe Redis activity. This was shown when the team was able to select the single database on the server and list the keys. The keys were shown in plaintext for the team to determine which key was the objective value.

Mitigation: The team recommends enabling TLS support in Redis which is readily available in Redis 6 and newer. Using secure tunnels for Redis communication and avoiding transmitting sensitive information via Redis in untrusted environments.

(Medium) Redis allowed all clients full access to administrative-level commands with no privilege separation.

Justification: Without any form of access control or privilege separation, any user can execute commands that can delete data, change configurations, or interact with the file system. While all of the administrative-level commands were available to the team, we did not use them as they were outside of the scope, but we were still able to obtain the objective value using certain commands.

Mitigation: The team recommends using Redis ACLs to limit access to only necessary commands per user or role and monitoring command usage patterns for anomaly detection. A lower-level user or employee should not have the same level of access or privilege as a higher or executive level user.

6.2 Strengths

Despite this lab being an educational, controlled environment that was designed to have a vulnerability to exploit, the HTB team did a very thorough job in containing other unnecessary vulnerabilities. The assessment team has provided several strengths along with recommendations.

Minimal Attack Surface:

- Only a singular port (6379) was left open, indicating the absence of unnecessary exposed services. This limits the number of entry points that an attacker can exploit.
- Industry Alignment: This approach reflects the principle of attack surface reduction, which is a core tenet of secure system design.
- Recommendation: Continue to apply this principle in other environments by running only essential services and segmenting functionality as needed.

Realistic Misconfiguration Scenario:

- The system was intentionally configured to reflect a real-world scenario: an exposed unauthenticated Redis instance with default settings.
- Educational Value: The lab mirrors scenarios frequently seen in cloud environments and mismanaged DevOps pipelines, reinforcing practical awareness.
- Recommendation: Use this setup as a foundation for teaching defensive measures such as hardening, network segmentation, and secure service deployment.

Isolated Environment with No Lateral Movement Risk:

- The Redeemer system was deployed as a standalone, isolated system with no ability to move to and from other machines. This ensures that this exploitation and other exploitation in other machines remain contained within their own respective machines and environments.
- Best Practice Reference: Reflects principles from network segmentation and zero trust models, where each system is treated as a potentially compromised environment.
- Recommendation: In production environments, this same containment approach can be achieved through firewalls and access control policies.

Clear Learning Objective and Attack Path:

- As this is an educational lab system, it was structured to have a singular, well-defined attack path with guides on reconnaissance, exploitation, and privilege escalation.
- Educational Value: Reinforces methodical assessment planning that are aligned with frameworks like PTES and IDART.
- Recommendation: Consider this format for red team exercises/training scenarios, where a single vulnerability can be tested both for offense and defense.

6.3 Observations

Users are able to restart their instances of the machine.

Description: Once a user starts an instance on Redeemer, they are able to reset that instance and obtain a new target IP address. However, continuously resetting the system could result in an overload on the computer's resources.

Recommendation: Constantly keep track of the rate in which a user resets an instance on the machine in order to mitigate the risk of an overload on a computer's or the host's resources. If there are an excessive number of resets, the host could pause the instances from reloading.

The scan rate of Nmap is too slow using the given command.

Description: The option "-p-" is meant to scan all TCP on the target host which can take quite a long time. When I ran the command given: "nmap -p- -sV {target_IP}", I was waiting upwards of 30 minutes without getting any results.

Recommendation: Add the option "–min-rate 5000" between "-p-" and "-sV" in order to decrease the total time that nmap is running as this new tag send at least 5,000 packets per second.

The order in which the keys are displayed are different every time a connection is made.

Description: The assessment team noticed the order in which the keys are displayed in the database are different whenever a connection is made. This could be helpful if the names of the keys were encrypted to further confuse an attacker if they bypass the security measures.

Recommendation: Continue to randomize the order in which the keys are displayed and encrypt the names of the keys instead of displaying them in plaintext.

7 Conclusions and Follow-on Activities

The assessment team found the Redeemer system adheres with the industry best practices across their entire platform. However, the team identified several vulnerabilities which can be used by an attacker to compromise the system and the sensitive information that it may contain. The vulnerability that the team recommends prioritizing is the unauthenticated and readily accessible Redis service. This misconfiguration enabled users who were connected to the network to scan and identify an open port, connect to the Redis server, and exfiltrate a target flag value. While the environment was isolated for training purposes, this scenario closely resembles real-world issues that are frequently encountered in cloud and enterprise environments. As the design of Redeemer was intentionally vulnerable, it reinforces the importance of secure defaults, minimal exposure, and the practical consequences of overlooking basic configuration changes.

For follow-on activities, the team recommends enforcing secure Redis configurations, restricting external access to internal services, and implementing monitoring for unauthorized activity for real-world applications. Another recommendation would be to review the configuration of Redeemer to further reinforce isolation and protection from outside threats in order to maintain the educational integrity of the system.

The assessment team would like to thank HackTheBox, and their entire team for providing a safe platform for users to educate and train themselves on cybersecurity material.

8 Appendix A: Full Summary of Redis Server

8.1 Target Host Information

- IP Address: 10.129.119.241
- Host Status: Up (Responds to ping)

8.2 Open Ports

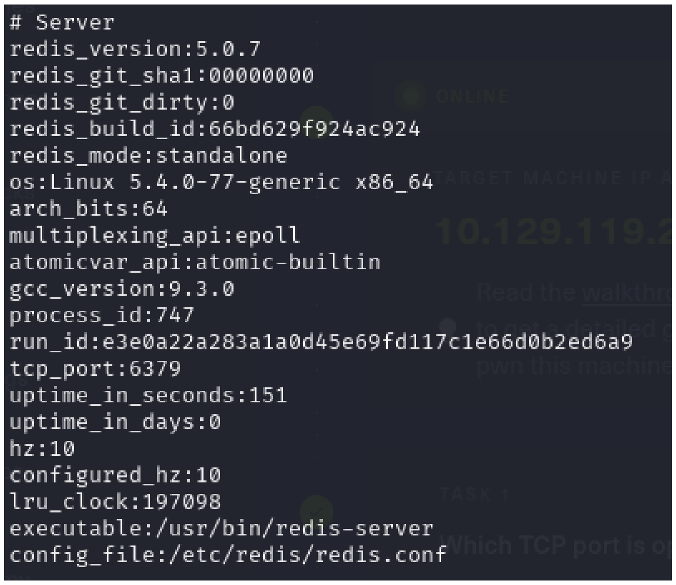
- Port: 6379
- Protocol: TCP
- State: Open
- Service: Redis
- OS Type: Linux

8.3 Uptime Information

- Uptime: 151 seconds

8.4 Operating System Information

- OS: Linux 5.4.0-77-Generic x86_64



```
# Server
redis_version:5.0.7
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:66bd629f924ac924
redis_mode:standalone
os:Linux 5.4.0-77-generic x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:9.3.0
process_id:747
run_id:e3e0a22a283a1a0d45e69fd117c1e66d0b2ed6a9
tcp_port:6379
uptime_in_seconds:151
uptime_in_days:0
hz:10
configured_hz:10
lru_clock:197098
executable:/usr/bin/redis-server
config_file:/etc/redis/redis.conf
```

Figure 9: Full information about Redis Server