HACKTHEBOX

Penetration Test Report

HTB Certified Penetration Testing Specialist (CPTS)

ELOQUIA

Report of Findings

Candidate Name: Pentester
Customer: Eloquia Corp.
Date: February 18, 2026
Version: 1.0

Contents

1 Executive Summary **2**
- 1.1 Assessment Overview . . . 2
- 1.2 Scope . . . 2
- 1.3 Recommendations Summary . . . 2

2 Internal Network Compromise Walkthrough **3**
- 2.1 Phase 1: OAuth CSRF Account Takeover . . . 3
  - 2.1.1 Vulnerability Identification . . . 3
  - 2.1.2 Exploitation Steps . . . 3
- 2.2 Phase 2: Remote Code Execution (SQLite) . . . 3
  - 2.2.1 Vulnerability Identification . . . 3
  - 2.2.2 Exploitation Steps . . . 3
- 2.3 Phase 3: Lateral Movement (DPAPI) . . . 4
  - 2.3.1 Credential Extraction . . . 4
  - 2.3.2 Exploitation Steps . . . 4
- 2.4 Phase 4: Privilege Escalation . . . 4
  - 2.4.1 Service Analysis . . . 4
  - 2.4.2 Binary Hijacking & Race Condition . . . 4
- 2.5 Proof of Compromise . . . 5

3 Technical Findings Details **6**
- 3.1 Finding 1: Insecure Service Permissions . . . 6
- 3.2 Finding 2: Unauthenticated RCE via SQLite . . . 6
- 3.3 Finding 3: OAuth Cross-Site Request Forgery . . . 6

4 Remediation Summary **7**
- 4.1 Short Term . . . 7
- 4.2 Long Term . . . 7

A Appendix **8**
- A.1 Exploited Hosts . . . 8
- A.2 Compromised Users . . . 8

Chapter 1

Executive Summary

1.1 Assessment Overview

This report details the findings of a penetration test conducted against the **Eloquia** infrastructure. The objective was to identify security vulnerabilities that could lead to unauthorized access, data exfiltration, or service disruption.

During the assessment, a critical exploit chain was identified that allowed for full system compromise. The attack path began with an OAuth misconfiguration on the external web application, allowing for account takeover. This was leveraged to execute arbitrary code via SQLite extension loading. Inside the network, weak credential protection (DPAPI) facilitated lateral movement, and a final privilege escalation was achieved by exploiting insecure file permissions on the custom “Failure2Ban” service.

1.2 Scope

The scope of this assessment was limited to the single host:

- **Hostname:** eloquia.htb
- **IP Address:** 10.10.11.xxx (Target)

1.3 Recommendations Summary

Immediate action is required to remediate the identified critical vulnerabilities. The primary recommendations are:

- Implement strict `state` parameter validation in OAuth flows.
- Disable the `load_extension()` function in SQLite configurations.
- Revoke Write permissions for non-administrative users on system service binaries.

Chapter 2

Internal Network Compromise Walkthrough

2.1 Phase 1: OAuth CSRF Account Takeover

2.1.1 Vulnerability Identification

The application's OAuth implementation failed to validate the `state` parameter during the authorization flow. This allowed an attacker to force a victim (Admin) to consume a malicious authorization code.

2.1.2 Exploitation Steps

1. Created a valid banner image satisfying the $>$20KB requirement using Python PIL.
2. Deployed the `exploit.py` script targeting the OAuth endpoint.
3. Created a malicious article containing a meta-refresh redirect.
4. Reported the article to the Admin Bot.
5. Upon the bot's visit, the CSRF payload executed, linking the attacker's account to the admin session.
6. **Result:** Gained administrative access to the Eloquia dashboard.

2.2 Phase 2: Remote Code Execution (SQLite)

2.2.1 Vulnerability Identification

The administrative panel allowed database interaction. It was discovered that the SQLite instance had the `load_extension()` function enabled, allowing the loading of external libraries.

2.2.2 Exploitation Steps

1. Compiled a malicious Windows DLL (`revshell.dll`) containing a reverse shell payload.

```
x86_64-w64-mingw32-gcc revshell.c -shared -o revshell.dll
```

2. Uploaded the DLL via the `/article/create/` endpoint.
3. Triggered execution via the SQL Explorer:

```sql
SELECT load_extension('static/assets/images/blog/revshell.dll', '
    sqlite3_revshell_init');
```

4. **Result:** Reverse shell obtained as user `eloquia\web`.

2.3 Phase 3: Lateral Movement (DPAPI)

2.3.1 Credential Extraction

Post-exploitation enumeration revealed Microsoft Edge browser data stored in the `Olivia.KAT` user directory. The Data Protection API (DPAPI) keys were protected only by the user's context, which was accessible via the compromised web process.

2.3.2 Exploitation Steps

1. Compiled a custom DLL (`dpapi.dll`) to interface with Windows Crypt32 API.
2. Injected the DLL into the database process using the same SQLite technique.
3. Extracted the decrypted Master Key: `c7f1ad7...`
4. Decrypted the Edge `Login Data` database using the key.
5. **Result:** Recovered credentials for user `Olivia.KAT`: `S3cureP@sswdIGu3ss`.

2.4 Phase 4: Privilege Escalation

2.4.1 Service Analysis

Enumeration identified a non-standard service named **Failure2Ban** running as `NT AUTHORITY\SYSTEM`.

- **Binary Path:** `C:\Program Files\Qooqle IPS Software\Failure2Ban - Prototype\Failure2Ban\bin\`
- **Permissions:** User `Olivia.KAT` possessed **Write (W)** permissions on the binary.

2.4.2 Binary Hijacking & Race Condition

Since the user could not manually restart the service, a race condition was exploited during the service's automated file processing loop.

1. Created a C payload (`xpl.exe`) to copy the root flag to a temp directory.
2. Transferred the payload to the target.
3. Executed a PowerShell loop to overwrite the service binary:

```powershell
$svc = 'C:\Program Files\Qooqle IPS Software\Failure2Ban - Prototype\
    Failure2Ban\bin\Debug\Failure2Ban.exe'
while ($true) { try { Copy-Item C:\temp\xpl.exe $svc -Force; break } catch
    {} }
```

4. Triggered a service restart by flooding the log file:

```
1..100000 | ForEach-Object { Add-Content -Path "C:\Web\Qooqle\log.csv" -
    Value "192.168.1.$_,failed_login,2024-02-17,ATTACK" }
```

5. **Result:** Service restarted, executing the payload as SYSTEM.

2.5 Proof of Compromise

- **User Flag:** Retrieved from `C:\Users\Olivia.KAT\Desktop\user.txt`
- **Root Flag:** Retrieved from `C:\temp\root.txt` after service hijacking.

Chapter 3

Technical Findings Details

3.1 Finding 1: Insecure Service Permissions

Severity: CRITICAL

Description: The `Failure2Ban` service executable allows Write access to non-administrative users. This allows any compromised user on the system to replace the binary with malicious code, which is then executed with System privileges upon service restart.

Remediation: Restrict permissions on the `Qooqle IPS Software` directory. Only Administrators and SYSTEM should have Write/Modify access.

3.2 Finding 2: Unauthenticated RCE via SQLite

Severity: CRITICAL

Description: The SQLite configuration allows the loading of arbitrary extensions via `load_extension()`. Coupled with an arbitrary file upload vulnerability, this leads to Remote Code Execution.

Remediation: Disable extension loading in the SQLite configuration or strictly whitelist allowed extensions.

3.3 Finding 3: OAuth Cross-Site Request Forgery

Severity: HIGH

Description: The application does not validate the `state` parameter in the OAuth authorization flow.

Remediation: Implement a cryptographically random `state` token that is verified upon the callback return.

Chapter 4

Remediation Summary

Priority	Remediation Action	Effort
High	Disable `load_extension` in SQLite	Low
High	Fix ACLs on Failure2Ban directory	Low
Medium	Implement OAuth State Validation	Medium
Medium	Rotate all compromised credentials	Medium
Low	Review DPAPI storage implementation	High

4.1 Short Term

- Immediately correct the Access Control Lists (ACLs) for the Failure2Ban service to prevent modification by standard users.
- Disable the dynamic loading of extensions in the database configuration.

4.2 Long Term

- Conduct a full code review of the authentication module.
- Implement a robust patch management policy to ensure third-party software (like the IPS prototype) does not introduce vulnerabilities.

Appendix A

Appendix

A.1 Exploited Hosts

Host	Scope	Method	Notes
Eloquia	Production	OAuth CSRF, DLL Hijacking	Fully Compromised

A.2 Compromised Users

Username	Type	Method
eloquia\web	Service Account	SQLite RCE
Olivia.KAT	User	DPAPI Decryption
NT AUTHORITY\SYSTEM	Administrator	Service Binary Hijack