SECURITY VULNERABILITY ASSESSMENT AND PENETRATION TESTING

Report on the results and associated recommendations arising from the Vulnerability assessment and penetration testing of two identified systems (Windows and Debian Linux)

Name : S Hemanth

Table of Contents

EXECUTIVE REPORT

TECHNICAL REPORT

Executive Report

Executive Summary

Simplilearn had asked to perform a security vulnerability assessment (VA) and penetration testing (PT) on the two identified systems (Windows and Debian Linux), document all the findings in a clear and repeatable manner, and provide remediation recommendations.

Approach

Testing was performed under a "white box" approach with the goal of identifying unknown weaknesses on the two identified systems. Testing was performed from a non-evasive standpoint with the goal of uncovering as many misconfigurations and vulnerabilities as possible. Testing was performed remotely via a host that was provisioned specifically for this assessment using both manual and automated methodology. Each weakness identified was documented and manually investigated to determine exploitation possibilities and escalation potential.

Scope

The scope of this assessment was to assess the two identified systems, whose IP addresses are 172.31.38.44 (Windows) and 172.31.46.74 (Debian Linux or Ubuntu), and check for vulnerabilities, perform a penetration test on those vulnerabilities and check if it is exploitable. This VA and PT will show if the infrastructure has any vulnerabilities that might be exploitable which might threaten the confidentiality, integrity, and availability(CIA Triad) of the systems.

Methodology and Tools Used

Assessment was done by both manual and automated methods to find the vulnerabilities in the system

- **Information gathering and Reconnaissance – OSINT tools like shodan**
- **Scanning and Enumeration – nmap**
- **Vulnerability assessment – Nessus Tool**
- **Analysis – sslscan, bash scripts, nmap scripts**

Assessment Findings

During the vulnerability assessment Windows server, we were able to identify three (3) vulnerabilities. The findings were categorized by severity level, with one (1) of the findings being assigned a high-risk rating, two (2) medium-risk and fifteen (15) informational findings.

During the vulnerability assessment of Ubuntu server, we were able to identify three (3) vulnerabilities. The findings were categorized by severity level, with three (3) of the findings being assigned a medium-risk rating and twenty-seven (27) informational findings.

While testing it was observed that the patch and vulnerability management to be well-maintained on both Windows and Ubuntu server. None of the findings in this report were related to missing operating system. Each flaw discovered during testing was related to a misconfiguration or lack of hardening, with most falling under the categories of self-signed SSL certificates (on Windows and Ubuntu) and expiry of SSL certificate (on Ubuntu) which cannot be trusted and was not found in the list of known certificate authorities. Further, while testing for open ports on Windows, it was observed that port 3389 (RDP) and 8443 were open and while testing for open ports on Ubuntu, it was observed that port 22 (SSH) and 8443 were open.

1. Windows

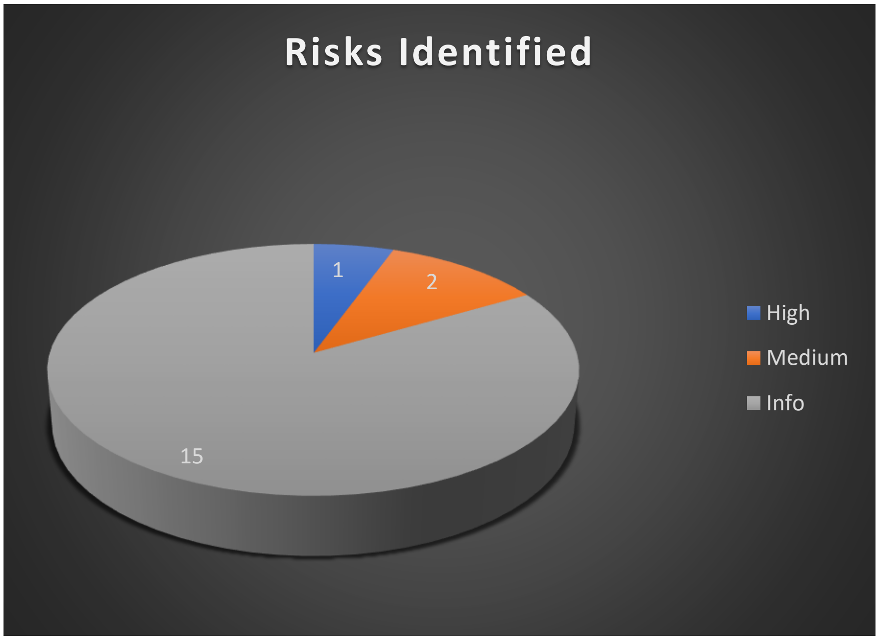


Fig 1: Risk Identified in Windows

Serial No	Severity	Vulnerability	Description
1	High	SSL Medium Strength Cipher Suites Supported (SWEET32)	The remote host supports the use od SSL ciphers that offer medium strength encryption. The uses of 3DES encryption suit is causing this vulnerability which is called SWEET32
2	Medium	SSL Certificate Cannot Be Trusted	The server's X.509 certificate cannot be trusted. The certificate is incorrectly used for digital signatures and does not match the supplied URI (signed by unknown certificate authority). Inspecting the certificate, it has an overall scare of "*zero*" and a grade of "*T*".
3	Medium	SSL Self-Signed Certificate	The SSL certificate was found at the top of the certificate chain sent by the remote host, but is self-signed and was not found in the list of known certificate authorities.

Table 1: Findings table 1

2. Ubuntu

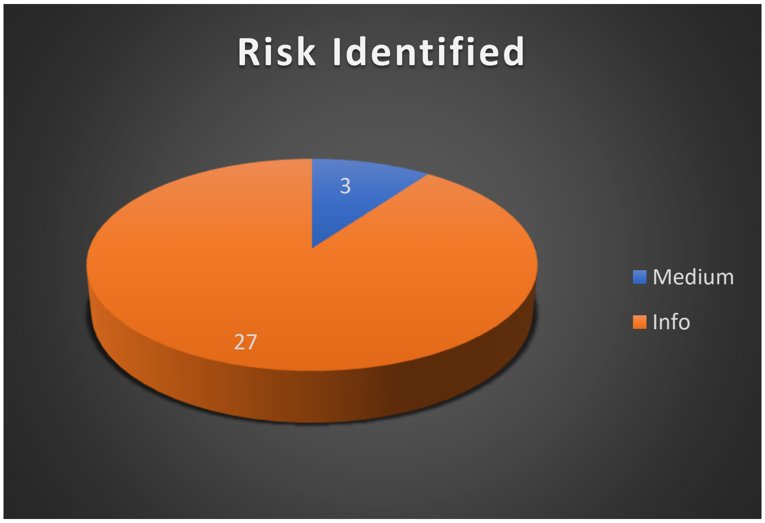


Fig 2: Risk Identified in Ubuntu

Serial No	Severity	Vulnerability	Description
1	Medium	SSL Certificate Cannot Be Trusted	The server’s X.509 certificate cannot be trusted. The certificate is incorrectly used for digital signatures and does not match the supplied URI (signed by unknown certificate authority). Inspecting the certificate, it has an overall scare of “*zero*” and a grade of “*T*”.
2	Medium	SSL Self-Signed Certificate	The SSL certificate was found at the top of the certificate chain sent by the remote host, but is self-signed and was not found in the list of known certificate authorities.
3	Medium	SSL Certificate Expiry	The SSL certificate has already expired. Not valid before : April 2 2021 Not valid after : April 2 2022

Table 2: Findings table 2

Area of Improvement

- Purchase or generate SSL certificate from a trusted certificate authority.
- Performing security tests often would check for any risks that would threaten the Confidentiality, Integrity and Availability of the systems.

Technical Report

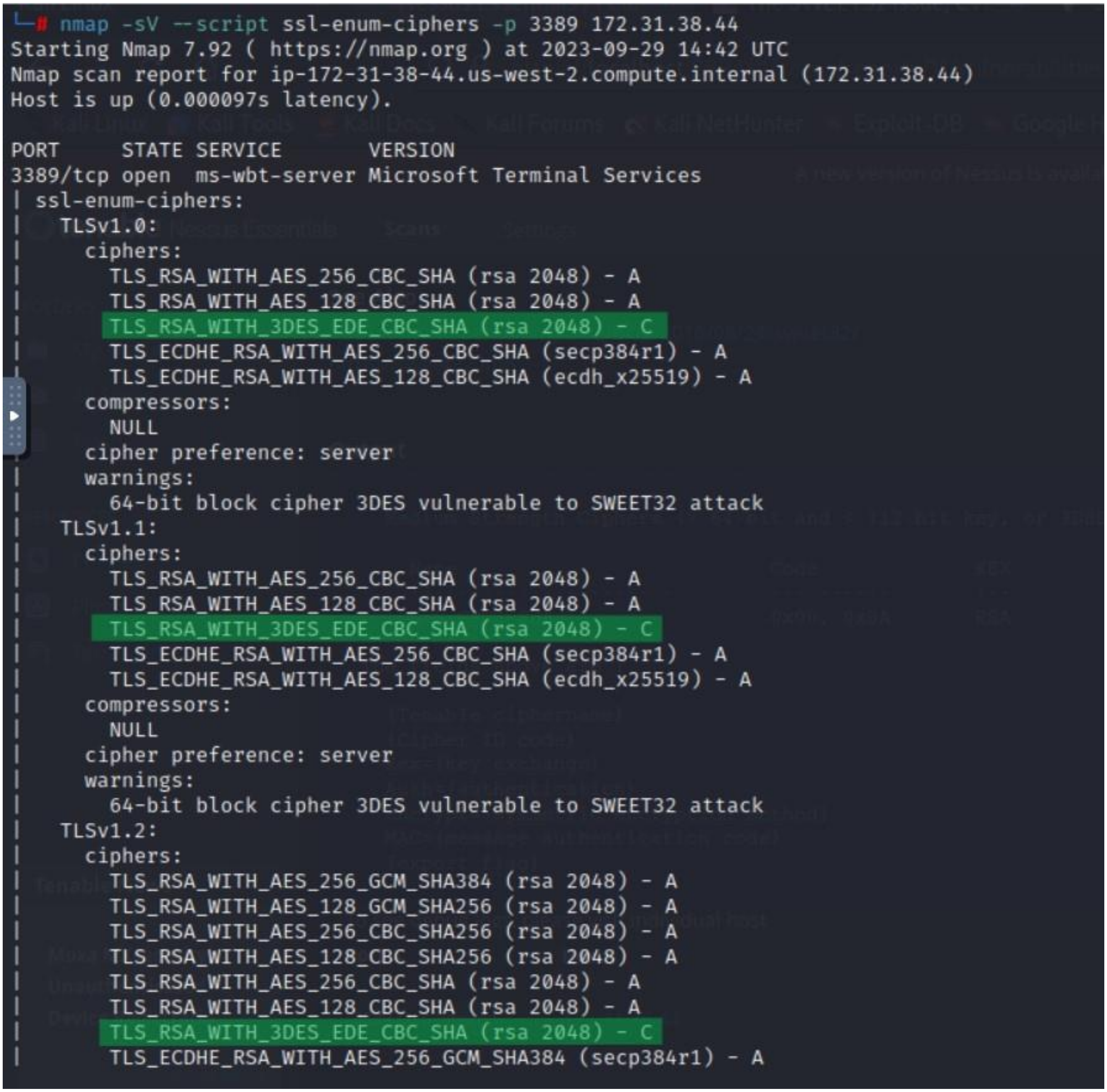
Technical Findings Detail of Windows

- **SSL Medium Strength Cipher Suites Supported (SWEET32)**

Severity	High
CVE	CVE-2016-2183
CVSS v3.1 Score	7.5
Description (Incl. Root Cause)	The remote host supports the use old SSL ciphers that offer medium strength encryption. Medium strength is regarded as the encryption that uses at least 64bits and less than 112 bits or else that uses 3DES encryption suite. The DES and Triple DES ciphers, as used in the TLS, SSH, and IPSec protocols and other protocols and products, have a birthday bound of approximately four billion blocks, which makes it easier for remote attackers to obtain cleartext data via a birthday attack against a long-duration encrypted session
Security Impact	It is considerably easier to circumvent medium strength encryption if the attacker is on the same physical network
Remediation	To help protect against this vulnerability, you need to disable some older cyphers in the registry.

Finding Evidence:

To check for this vulnerability we used the nmap script command "*nmap -sV –script ssl-enum-ciphers -p 3389 172.31.38.44***" to connect via the open port and scan for ssl ciphers.**



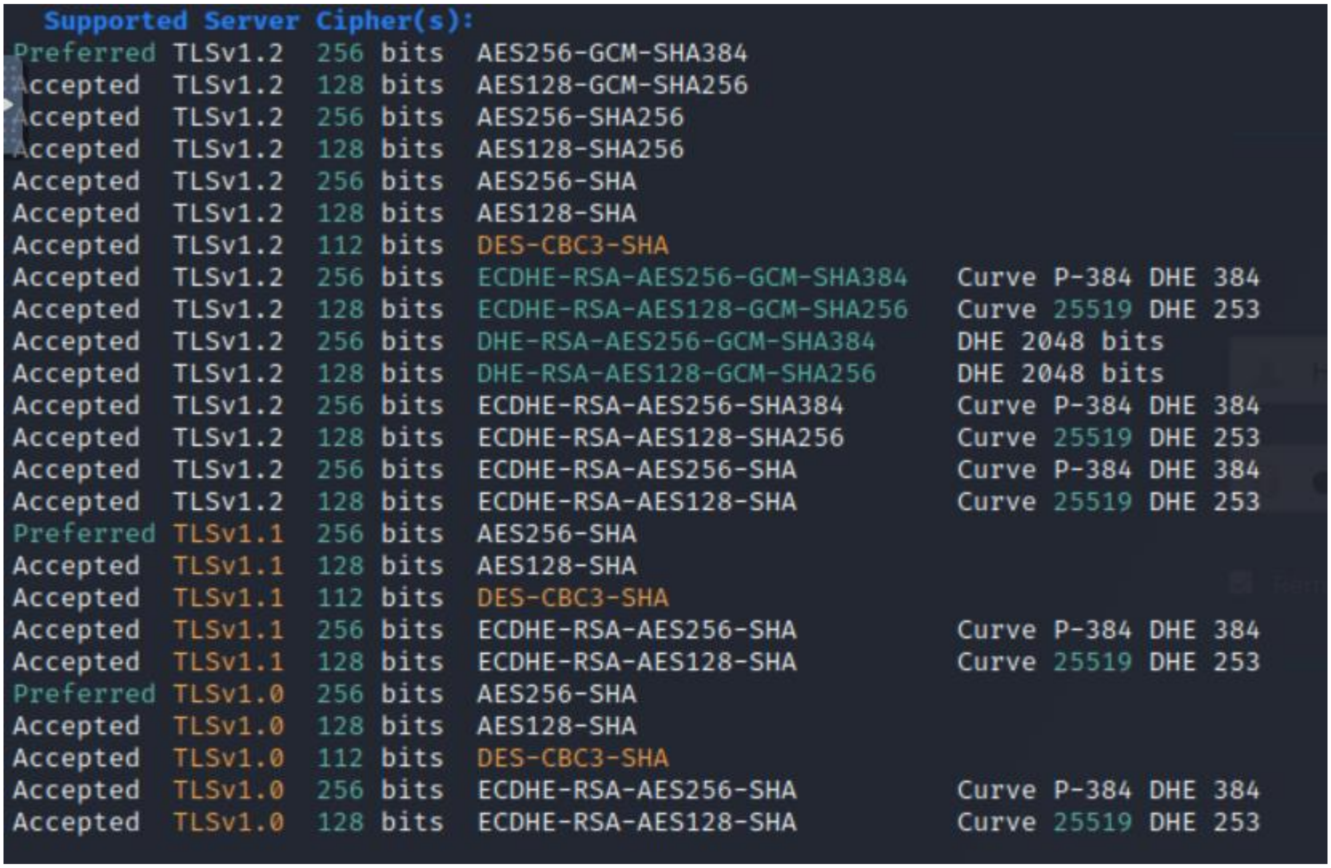
```
└─# nmap -sV --script ssl-enum-ciphers -p 3389 172.31.38.44
Starting Nmap 7.92 ( https://nmap.org ) at 2023-09-29 14:42 UTC
Nmap scan report for ip-172-31-38-44.us-west-2.compute.internal (172.31.38.44)
Host is up (0.000097s latency).

PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp384r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
      compressors:
        NULL
      cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|   TLSv1.1:
|     ciphers:
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp384r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|   TLSv1.2:
|     ciphers:
|       TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp384r1) - A
```

Fig 3 : Finding 3DES ciphers

Here we see that the ciphers grade is "C" which is not secure enough and it also shows that it is vulnerable to SWEET32 attack.

Used sslscan to check for supported ciphers

Command : *sslscan 172.31.38.44:3389*



```
  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  AES256-GCM-SHA384
Accepted  TLSv1.2  128 bits  AES128-GCM-SHA256
Accepted  TLSv1.2  256 bits  AES256-SHA256
Accepted  TLSv1.2  128 bits  AES128-SHA256
Accepted  TLSv1.2  256 bits  AES256-SHA
Accepted  TLSv1.2  128 bits  AES128-SHA
Accepted  TLSv1.2  112 bits  DES-CBC3-SHA
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve P-384 DHE 384
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  DHE-RSA-AES256-GCM-SHA384     DHE 2048 bits
Accepted  TLSv1.2  128 bits  DHE-RSA-AES128-GCM-SHA256     DHE 2048 bits
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve P-384 DHE 384
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve P-384 DHE 384
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Preferred TLSv1.1  256 bits  AES256-SHA
Accepted  TLSv1.1  128 bits  AES128-SHA
Accepted  TLSv1.1  112 bits  DES-CBC3-SHA
Accepted  TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve P-384 DHE 384
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Preferred TLSv1.0  256 bits  AES256-SHA
Accepted  TLSv1.0  128 bits  AES128-SHA
Accepted  TLSv1.0  112 bits  DES-CBC3-SHA
Accepted  TLSv1.0  256 bits  ECDHE-RSA-AES256-SHA          Curve P-384 DHE 384
Accepted  TLSv1.0  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
```

Fig 4 : Supported server Ciphers along with the encryption bits used

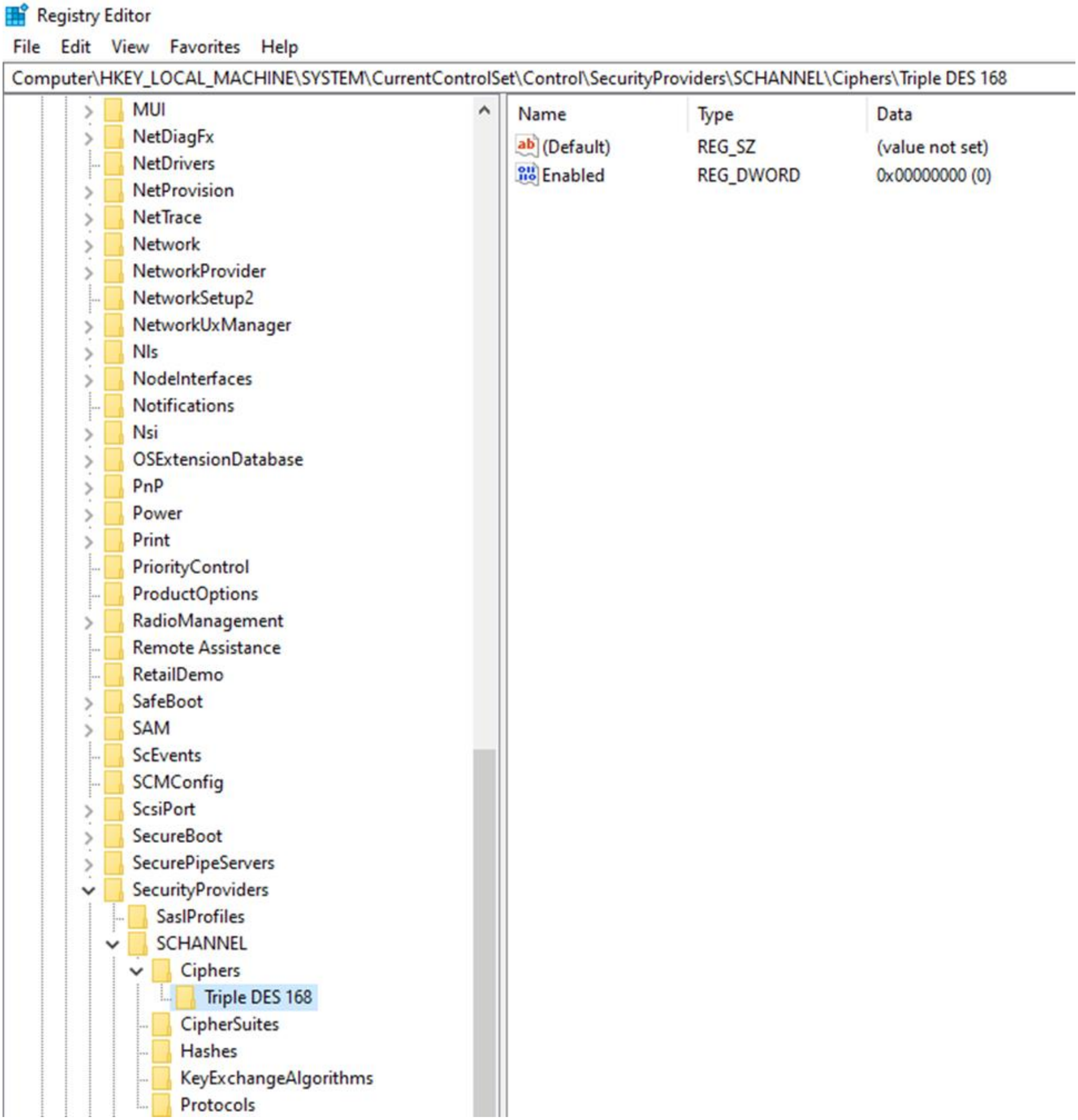
It is observed that DES cipher uses 112bit encryption which is medium dtrength

Remediation

Disable 3DES **:** To disable 3DES on your Windows server, set the following registry key:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\Triple DES 168]

“Enabled”=dword:00000000



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\Triple DES 168

```
> MUI
> NetDiagFx
  NetDrivers
> NetProvision
> NetTrace
> Network
> NetworkProvider
  NetworkSetup2
> NetworkUxManager
> Nls
> NodeInterfaces
  Notifications
> Nsi
> OSExtensionDatabase
> PnP
> Power
> Print
  PriorityControl
  ProductOptions
> RadioManagement
  Remote Assistance
  RetailDemo
> SafeBoot
> SAM
  ScEvents
  SCMConfig
> ScsiPort
> SecureBoot
> SecurePipeServers
v SecurityProviders
    SaslProfiles
  v SCHANNEL
    v Ciphers
        Triple DES 168
      CipherSuites
      Hashes
      KeyExchangeAlgorithms
      Protocols
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
Enabled	REG_DWORD	0x00000000 (0)

Fig 5: Remediation step

- **SSL Certificate Cannot Be Trusted**

Severity	Medium
CVE	None
CVSS v3.1 Score	6.5
Description (Incl. Root Cause)	The server's X.509 certificate cannot be trusted. This situation can occur because the top of the certificate chain sent by the server is not descended from a known public certificate authority. This can occur either when the top of the chain is an unrecognized or when intermediate certificates are missing that would connect the top of the certificate chain to a known public certificate authority.
Security Impact	If the remote host is a public host in production, any break in the chain makes it more difficult for users to verify the authenticity and identity of the web server. This could make it easier to carry out man-in-the-middle attacks against the remote host.
Remediation	Purchase or generate a proper SSL certificate for this service.

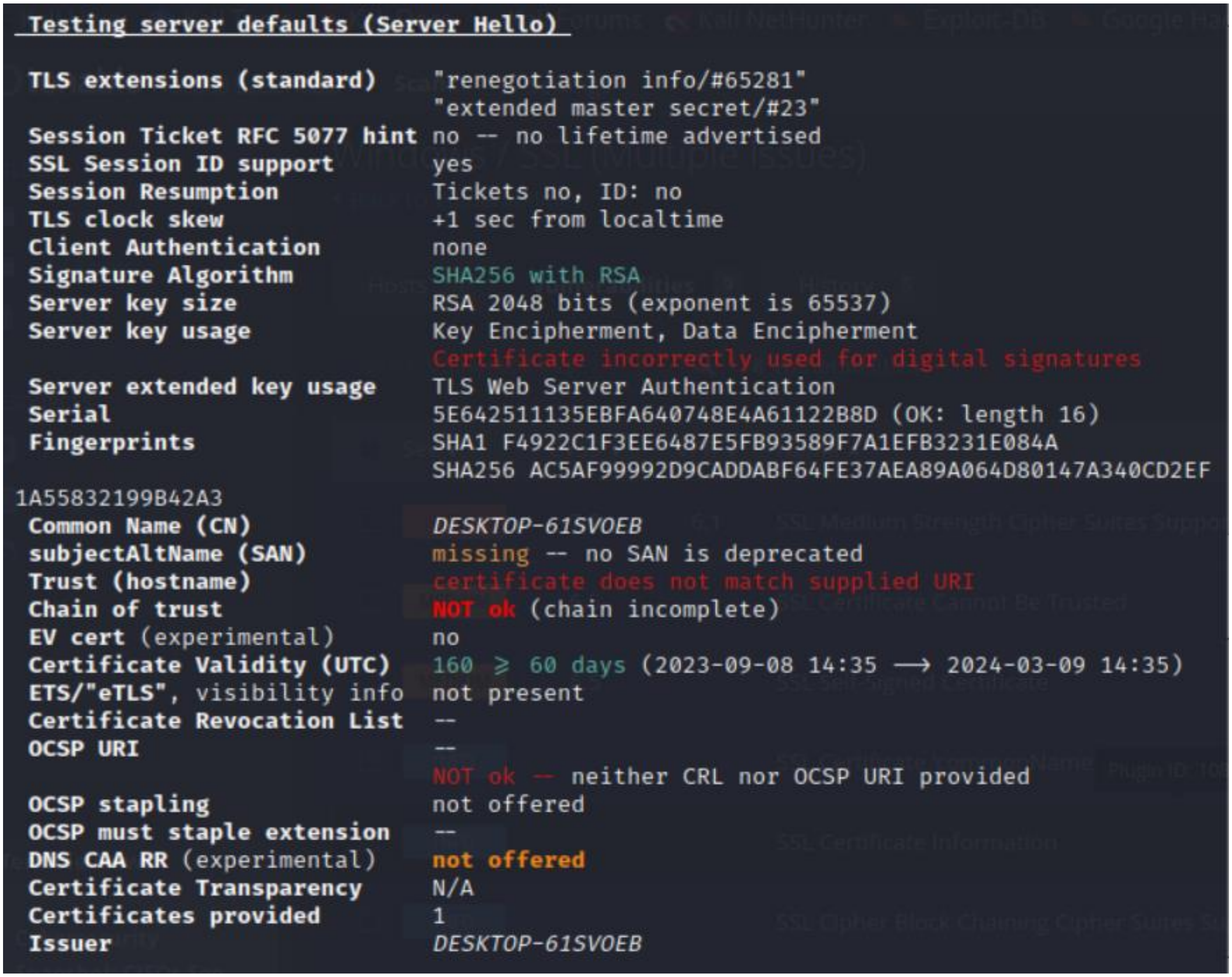
Finding Evidence

To check for this vulnerability we used a bash script to test TLS/SSL encryption anywhere on any port.

Command used : *“bash testssl.sh 172.31.38.44:3389”*

First, we cloned the code from GitHub and then ran the bash script

testssl.sh is a free command line tool which checks a server's service on any port for the support of TLS/SSL ciphers, protocols as well as recent cryptographic flaws and more.



```
 Testing server defaults (Server Hello) 

 TLS extensions (standard)    "renegotiation info/#65281"
                              "extended master secret/#23"
 Session Ticket RFC 5077 hint no -- no lifetime advertised
 SSL Session ID support       yes
 Session Resumption           Tickets no, ID: no
 TLS clock skew               +1 sec from localtime
 Client Authentication        none
 Signature Algorithm          SHA256 with RSA
 Server key size              RSA 2048 bits (exponent is 65537)
 Server key usage             Key Encipherment, Data Encipherment
                              Certificate incorrectly used for digital signatures
 Server extended key usage    TLS Web Server Authentication
 Serial                       5E642511135EBFA640748E4A61122B8D (OK: length 16)
 Fingerprints                 SHA1 F4922C1F3EE6487E5FB93589F7A1EFB3231E084A
                              SHA256 AC5AF99992D9CADDABF64FE37AEA89A064D80147A340CD2EF
1A55832199B42A3
 Common Name (CN)             DESKTOP-61SVOEB
 subjectAltName (SAN)         missing -- no SAN is deprecated
 Trust (hostname)             certificate does not match supplied URI
 Chain of trust               NOT ok (chain incomplete)
 EV cert (experimental)       no
 Certificate Validity (UTC)   160 ≥ 60 days (2023-09-08 14:35 --> 2024-03-09 14:35)
 ETS/"eTLS", visibility info  not present
 Certificate Revocation List  --
 OCSP URI                     --
                              NOT ok -- neither CRL nor OCSP URI provided
 OCSP stapling                not offered
 OCSP must staple extension   --
 DNS CAA RR (experimental)    not offered
 Certificate Transparency     N/A
 Certificates provided        1
 Issuer                       DESKTOP-61SVOEB
```

Fig 6 : Testing Windows server defaults

Here we observe that the chain of trust is giver as "*Not ok*". Certificate is incorrectly used for digital signatures and it does not match the supplied URI, meaning that the certificates are missing that would connect the top of the certificate chain to a known public certificate authority.

Remediation

Purchase or generate a proper SSL certificate for this service.

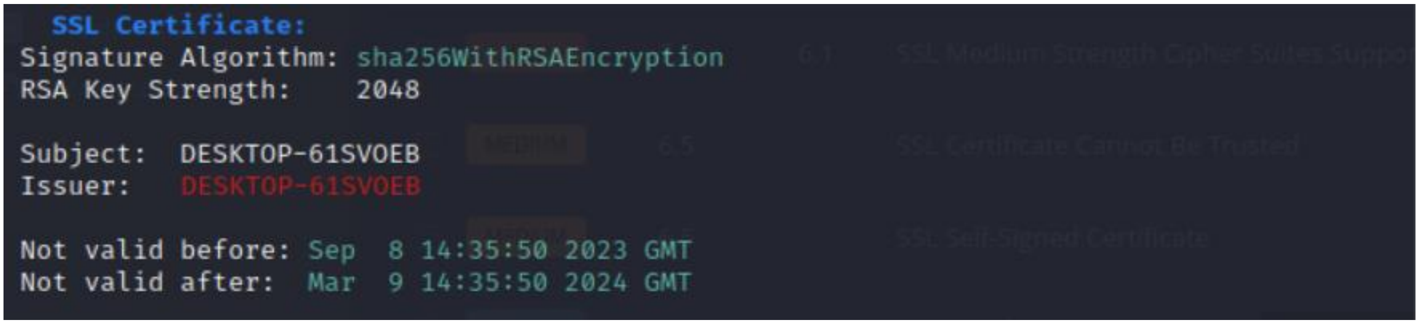
- **SSL Self-Signed Certificate**

Severity	Medium
CVE	None
CVSS v3.1 Score	6.5
Description (Incl. Root Cause)	The X.509 certificate chain for this service is not signed by a recognized certificate authority. The certificate chain contains a signature that could not be verified. Bad signatures can be fixed by getting the certificate with the bad signature to be re-signed by its issuer. Signatures that could not be verified are the result of being self-signed .
Security Impact	If the remote host is a public host in production, any break in the chain makes it more difficult for users to verify the authenticity and identity of the web server. This could make it easier to carry out man-in-the-middle attacks against the remote host.
Remediation	Purchase or generate a proper SSL certificate for this service.

Finding Evidence

To check this vulnerability we used sslscan command to verify the issuer of the SSL certificate

Command used : *sslscan 172.31.38.44:3389*



```
  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject:  DESKTOP-61SVOEB
Issuer:   DESKTOP-61SVOEB

Not valid before: Sep  8 14:35:50 2023 GMT
Not valid after:  Mar  9 14:35:50 2024 GMT
```

Fig 7: Self-signed

The issuer of the certificate is same as the hostname of the server which is clearly self-signed

Remediation

Purchase or generate a proper SSL certificate for this service.

Technical Findings Detail of Ubuntu

- **SSL Certificate Cannot Be Trusted**

Severity	Medium
CVE	None
CVSS v3.1 Score	6.5
Description (Incl. Root Cause)	The server's X.509 certificate cannot be trusted. This situation can occur because the top of the certificate chain sent by the server is not descended from a known public certificate authority. This can occur either when the top of the chain is an unrecognized or when intermediate certificates are missing that would connect the top of the certificate chain to a known public certificate authority.
Security Impact	If the remote host is a public host in production, any break in the chain makes it more difficult for users to verify the authenticity and identity of the web server. This could make it easier to carry out man-in-the-middle attacks against the remote host.
Remediation	Purchase or generate a proper SSL certificate for this service.

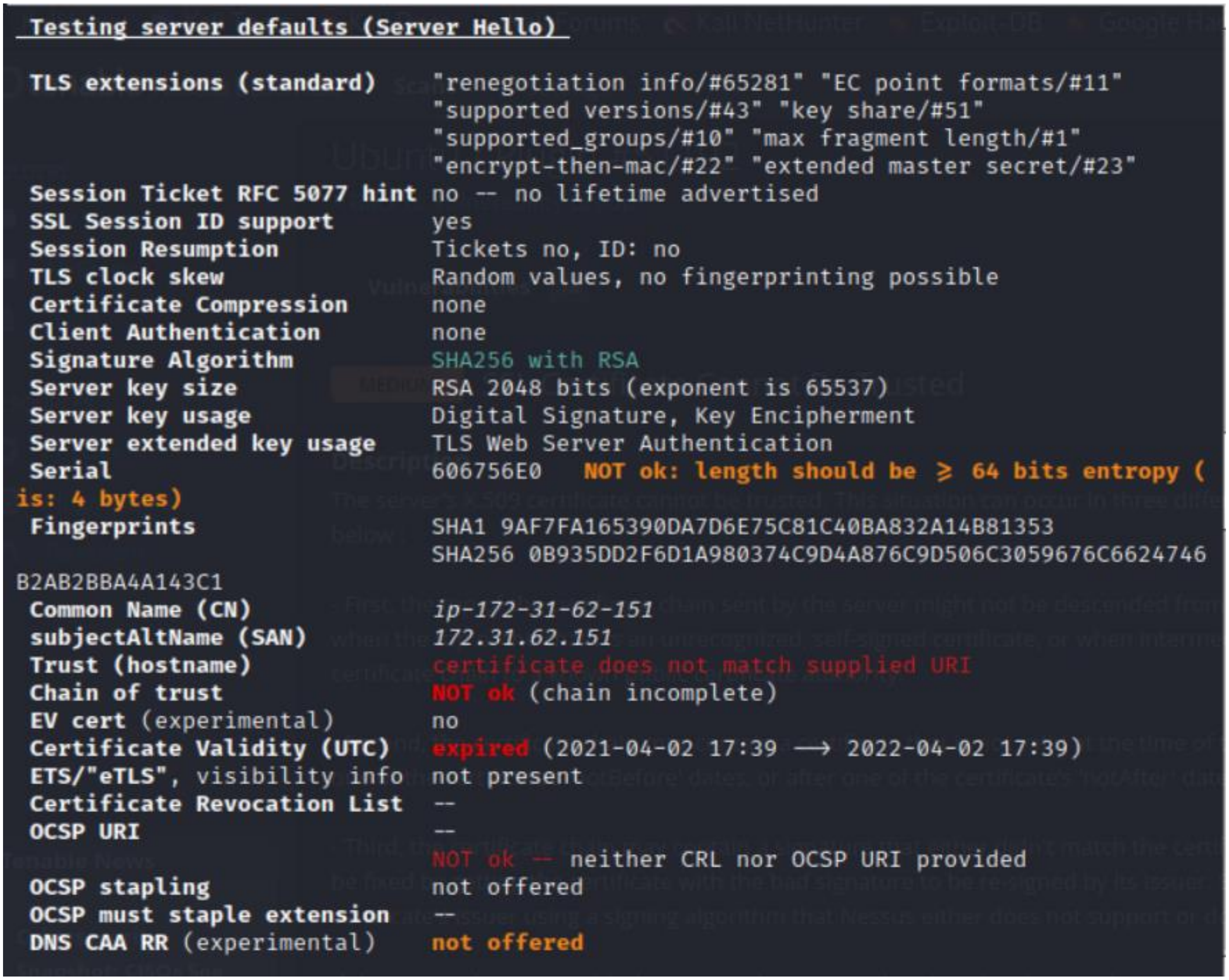
Finding Evidence

To check for this vulnerability we used a bash script to test TLS/SSL encryption anywhere on any port.

Command used : *“bash testssl.sh 172.31.46.74:8443”*

First, we cloned the code from GitHub and then ran the bash script

testssl.sh is a free command line tool which checks a server's service on any port for the support of TLS/SSL ciphers, protocols as well as recent cryptographic flaws and more.



```
 Testing server defaults (Server Hello)

 TLS extensions (standard)    "renegotiation info/#65281" "EC point formats/#11"
                              "supported versions/#43" "key share/#51"
                              "supported_groups/#10" "max fragment length/#1"
                              "encrypt-then-mac/#22" "extended master secret/#23"
 Session Ticket RFC 5077 hint no -- no lifetime advertised
 SSL Session ID support       yes
 Session Resumption           Tickets no, ID: no
 TLS clock skew               Random values, no fingerprinting possible
 Certificate Compression      none
 Client Authentication        none
 Signature Algorithm          SHA256 with RSA
 Server key size              RSA 2048 bits (exponent is 65537)
 Server key usage             Digital Signature, Key Encipherment
 Server extended key usage    TLS Web Server Authentication
 Serial                       606756E0   NOT ok: length should be ≥ 64 bits entropy (
is: 4 bytes)
 Fingerprints                 SHA1 9AF7FA165390DA7D6E75C81C40BA832A14B81353
                              SHA256 0B935DD2F6D1A980374C9D4A876C9D506C3059676C6624746
B2AB2BBA4A143C1
 Common Name (CN)             ip-172-31-62-151
 subjectAltName (SAN)         172.31.62.151
 Trust (hostname)             certificate does not match supplied URI
 Chain of trust               NOT ok (chain incomplete)
 EV cert (experimental)       no
 Certificate Validity (UTC)   expired (2021-04-02 17:39 --> 2022-04-02 17:39)
 ETS/"eTLS", visibility info  not present
 Certificate Revocation List  --
 OCSP URI                     --
                              NOT ok -- neither CRL nor OCSP URI provided
 OCSP stapling                not offered
 OCSP must staple extension   --
 DNS CAA RR (experimental)    not offered
```

Fig 8: Testing Ubuntu server defaults

Here we observe that the chain of trust is giver as “*Not ok*”. Certificate is incorrectly used for digital signatures and it does not match the supplied URI, meaning that the certificates are missing that would connect the top of the certificate chain to a known public certificate authority.

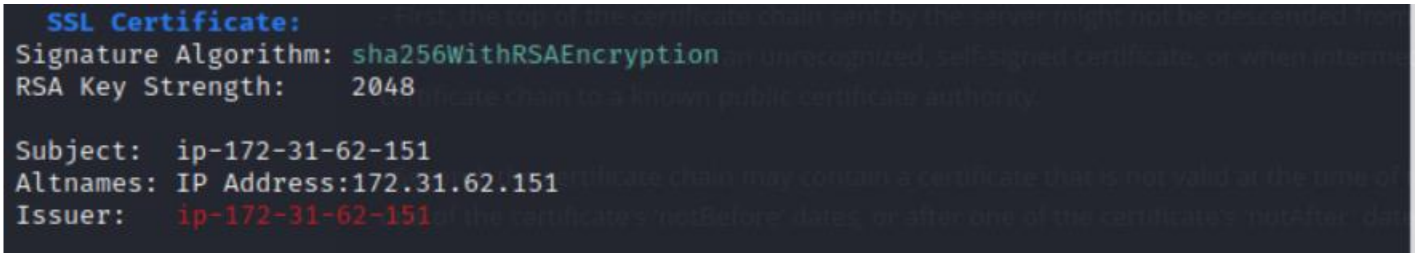
- **SSL Self-Signed Certificate**

Severity	Medium
CVE	None
CVSS v3.1 Score	6.5
Description (Incl. Root Cause)	The X.509 certificate chain for this service is not signed by a recognized certificate authority. The certificate chain contains a signature that could not be verified. Bad signatures can be fixed by getting the certificate with the bad signature to be re-signed by its issuer. Signatures that could not be verified are the result of being self-signed .
Security Impact	If the remote host is a public host in production, any break in the chain makes it more difficult for users to verify the authenticity and identity of the web server. This could make it easier to carry out man-in-the-middle attacks against the remote host.
Remediation	Purchase or generate a proper SSL certificate for this service.

Finding Evidence

To check this vulnerability we used sslscan command to verify the issuer of the SSL certificate

Command used : *sslscan 172.31.46.74:8443*



```
  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject:  ip-172-31-62-151
Altnames: IP Address:172.31.62.151
Issuer:   ip-172-31-62-151
```

Fig 9: Self-signed (Ubuntu)

The issuer of the certificate is same as the hostname of the server which is clearly self-signed

Remediation

Purchase or generate a proper SSL certificate for this service.

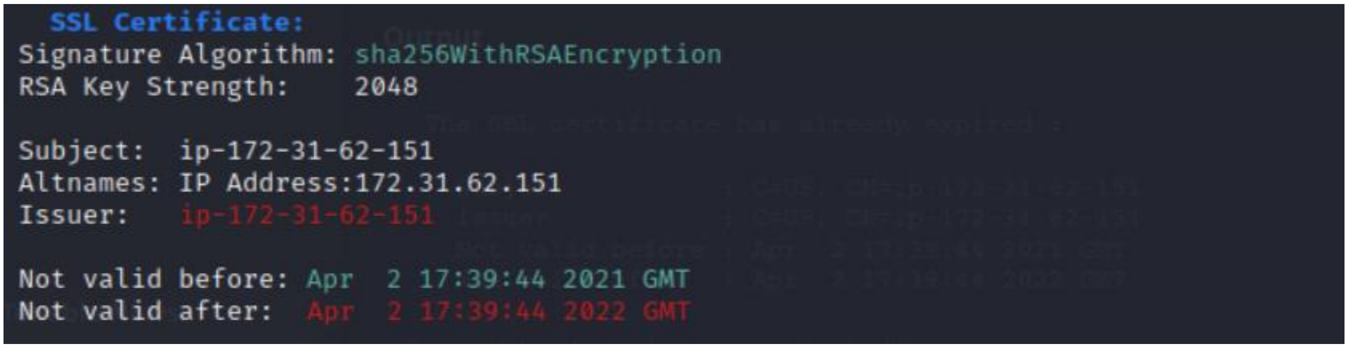
- **SSL Certificate Expiry**

Severity	Medium
CVE	None
CVSS v3.1 Score	5.3
Description (Incl. Root Cause)	The remote server's SSL certificate has already expired. The plugin of the vulnerability scanner checks expiry dates of certificates associated with SSL- enabled services on the target and reports whether any have already expired.
Security Impact	If the remote host is a public host in production, any break in the chain makes it more difficult for users to verify the authenticity and identity of the web server. This could make it easier to carry out man-in-the-middle attacks against the remote host.
Remediation	Purchase or generate a proper SSL certificate for this service.

Finding Evidence

To check this vulnerability we used sslscan command to verify the issuer of the SSL certificate

Command used : *sslscan 172.31.46.74:8443*



```
  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject:  ip-172-31-62-151
Altnames: IP Address:172.31.62.151
Issuer:   ip-172-31-62-151

Not valid before: Apr  2 17:39:44 2021 GMT
Not valid after:  Apr  2 17:39:44 2022 GMT
```

Fig 10: Expired SSL certificate (Ubuntu)

The certificate has expired as it is not valid after April 2 2022

Remediation

Purchase or generate a proper SSL certificate for this service.

Appendices

Appendix A – Finding Severities

Each finding has been assigned a severity rating of high, medium, or low. The rating is based 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 the infrastructure.

Rating	Severity Rating Definition
High	Exploitation of the technical or procedural vulnerability will cause substantial harm. Significant political, financial, and/or legal damage is likely to result. The threat exposure is high, thereby increasing the likelihood of occurrence. Security controls are not effectively implemented to reduce the severity of impact if the vulnerabilities were exploited.
Medium	Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity, and/or availability of the system, application, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment. The threat exposure is moderate-to-high, thereby increasing the likelihood of occurrence. Security controls are in place to contain the severity of impact if the vulnerabilities were exploited, such that further political, financial, or legal damage will not occur. - OR - The vulnerability is such that it would otherwise be considered High Risk, but the threat exposure is so limited that the likelihood of occurrence is minimal.
Low	Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The Confidentiality, Integrity and Availability (CIA) of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment. The threat exposure is moderate-to-low. Security controls are in place to contain the severity of impact if the vulnerabilities were exploited, such that further political, financial, or legal damage will not occur. - OR - The vulnerability is such that it would otherwise be considered Medium Risk, but the threat exposure is so limited that the likelihood of occurrence is minimal.

Appendix B – Hosts Tested for Vulnerability assessment and Penetration Testing

Host	Description
172.31.38.44	Windows Machine
172.31.46.74	Ubuntu Machine