Day 7

Exploitation Analyst

Hacking the SSL Network protocol:

Self-Signed Certificates:

What are Self signed certificates?

A self-signed certificate is a digital certificate signed by the same entity that created it, instead of a trusted Certificate Authority (CA). It provides encryption via SSL/TLS but lacks third-party validation, so its authenticity can't be independently verified. Browsers typically show a warning when encountering self-signed certificates. These are often used for internal testing, development environments, or local services. While they enable encrypted communication, they are not suitable for production use where trust is essential.

Can we use self-signed certificate in local network of the company?

If your websites are hosted internally and only accessible by employees within your organization's network, using a self-signed SSL certificate can be a practical and cost-effective solution. Self-signed certificates offer the same level of encryption and data integrity as CA-signed certificates but without involving an external Certificate Authority. Since the sites are only accessible within the internal environment, and the certificate can be manually trusted across all company systems, they still protect data from being intercepted within the network. However, the risk arises when users are not careful about trusting the correct certificate or if network access is extended (e.g., via VPNs or BYOD policies).

How this self-signed certificate method can be made more secured if used in local network of any company?

To enhance the security of this setup, organizations should implement a controlled certificate distribution process where the self-signed certificate is manually and securely installed into the trusted root certificate stores of all authorized employee devices. Enforcing strict certificate validation (e.g., by enabling certificate pinning or fingerprint verification in applications), setting short certificate expiration with proper rotation policies, and restricting access to internal services through firewalls or VPNs can significantly reduce attack vectors like MITM or rogue insider attacks. Additionally, using internal Certificate Authorities (e.g., via Microsoft AD CS) offers a scalable and more secure alternative to raw self-signed certs, while still keeping everything internal.

How to create a valid self-signed SSL Certificate?

YouTube:

https://youtu.be/VH4gXcvkmOY?si=Cws3D1cH_hjfOXL6

Steps:

This command creates a **2048-bit RSA private key**, saved in a file called mysite.key:

```
┌──(root㉿kali)-[~]
└─# openssl genrsa -out mysite.key 2048
```

This command generates a **CSR** (certificate signing request), which contains your public key and identity information (like domain name, company, country, etc.):

```
┌──(root㉿kali)-[~]
└─# openssl req -new -key mysite.key -out mysite.csr

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:BIhar
Locality Name (eg, city) []:Muz
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Aditya
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:IT
Email Address []:adii.utsav@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:Chlbhaag@9211
An optional company name []:oo
```

This command takes your CSR and private key and generates a **self-signed SSL certificate** valid for 365 days:

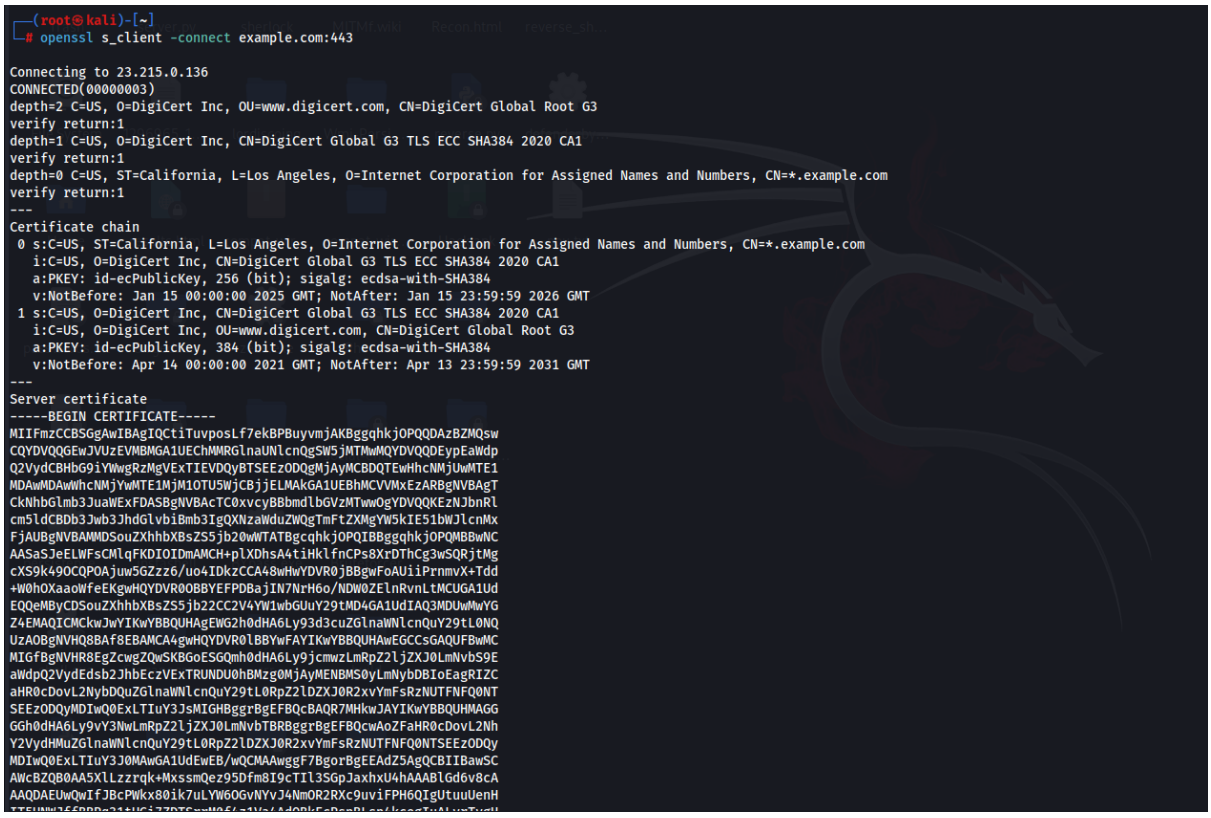
```
┌──(root㉿kali)-[~]
└─# openssl x509 -req -days 365 -in mysite.csr -signkey mysite.key -out mysite.crt

Certificate request self-signature ok
subject=C=IN, ST=BIhar, L=Muz, O=Aditya, OU=IT, CN=IT, emailAddress=adii.utsav@gmail.com
```

Note: Among the files generated while creating a self-signed SSL certificate, the most crucial one to protect is the mysite.key file — the private key. This key proves ownership of the certificate and is used to encrypt or sign data securely. If a hacker gains access to it, they can impersonate your server, decrypt SSL traffic, or sign fake certificates, completely undermining the security SSL is meant to provide. Therefore, it must be stored securely with strict file permissions and never exposed publicly or shared over insecure channels.

How to create a invalid self-signed SSL Certificate?

Steps:

Identify the Target Certificate: Here target is example.com



Create a Spoofed Certificate with Same CN:

```
┌──(root㉿kali)-[~]
└─# openssl req -newkey rsa:2048 -nodes -keyout fake.key -x509 -days 365 -out fake.crt

...+..+.+.....................+.........+......+.+...+...+..+...+..................+.......................+.+..+...
+.+...+...+.........+.....+.+...+..........+...+.......+...+...+..+......+.+.....+.+...++++++++++++++++++++++++++++
++++++++++++*..+...+...+..+...+....+.....+..............+.+.........+...+..+...+.......+...++++++++++++++++++++++++++
++++++++++++++*..+............+..+.......+.....+...+...+...+.+..............+.+........+............+.+..+..........
+...+.....+..........+...+...........+.............+..+....+..+.......+......+......+...+...+..+......+.+.....+.+..+
.+..+.......+...+..+...+.......+..............+...+.........+..........+...+.+......+.....+...+..............+...
....+.........+...+..............+..+.+........+.....+.............+...+...+....+...+...+.........+.....+...+...
.........+....+..+...+.+...+...........+....+...+............+...+..+.+..+.......+......+..+............+...+.......
...+..+............+.+.................+.......+......+............+.....+...+............+......+.+......+..+......
....+...+...+...........+..............+..........+.....+.+............+.........................+.........+...+..
.......+.+........+.++++++
.....+..............++++++++++++++++++++++++++++++++++++++++++*.++++++++++++++++++++++++++++++++++++++++++*..+....+.....+.
+..............+.....+..............+......+.+...+......+.....+....+.....+.+.....+......+.......+..+......+.......
+........+.........+......+...+....+...+.....+..........+...+......++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Los Angeles
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Internet Corporation for Assigned Names and Numbers
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:*.example.com
Email Address []:
```

Serve the Spoofed Certificate:

```
┌──(root㉿kali)-[~]
└─# openssl s_server -cert fake.crt -key fake.key -accept 443

Using default temp DH parameters
ACCEPT
```

How to protect organization from these spoofed self-signed certificates?

1. Use only trusted Certificate Authorities (CAs)
2. Enable HSTS (HTTP Strict Transport Security)
3. Implement certificate pinning in mobile and desktop apps
4. Monitor TLS/SSL traffic with IDS/IPS (e.g., Snort, Suricata, Zeek)
5. Set up SIEM alerts for certificate anomalies (self-signed, invalid chain, CN mismatch)
6. Monitor public Certificate Transparency (CT) logs for your domain
7. Educate users not to ignore browser certificate warnings
8. Avoid using insecure options like curl -k or --insecure in production
9. Ensure servers and clients verify certificate chains properly
10. Revoke compromised or misused certificates immediately

--The End--