INSTITUTE FOR ADVANCED COMPUTING
AND
SOFTWARE DEVELOPMENT
AKURDI, PUNE

DOCUMENTATION ON

ENCRYPTED TRAFFIC ANALAYSIS USING SSL-PROXY

PG-DITISS SEPT-2023

SUBMITTED BY

Group No: 25

MACHANI CHANDRA MOULI (239410)
BACHHAV YOGESH HIRALAL (239453)

MS. RUTUJA KULKARNI
PROJECT GUIDE

MR. ROHIT PURANIK
CENTRE CO-ORDINATOR

ABSTRACT

This project presents an analysis of the main mechanisms of decryption of SSL/TLS traffic. Also,the approach for intercepting and decrypting traffic transmitted over SSL/TLS is observed and tested. The developed approach has been automated and can be used for remote listening of the network, which will allow to decrypt transmitted data in a mode close to real time. The implementation of an adaptive proxy is carried out in between a client and SSL/TLS web server using more practical and approach that can moderate the ongoing and future SSL/TLS sessions. The aim is not to compromise user privacy.

TABLE OF CONTENTS

SECTION I

SECTION-II

1.INTRODUCTION

Secure Sockets Layer (SSL) is an application-level protocol that provides encryption technology for the Internet. SSL, also called Transport Layer Security (TLS), ensures the secure transmissionof data between a client and a server through a combination of privacy, authentication, confidentiality, and data integrity. SSL relies on certificates and private-public key exchange pairs for this level of security. The SSL protocol secures a wide range of wire-line and wireless applications, including Web commerce, Internet communications and financial management.

A *proxy server* is an intermediary between a user's computer and the Internet. A user first connectsto a proxy server when requesting web pages, videos or any data online. The proxy server then retrieves data that have been previously cached. If an entirely new request, the proxy server gets data from the original source and caches it for future use.

A Secure Sockets Layer (SSL) proxy server ensures secure transmission of data with encryption technology. Security in an SSL connection relies on proxy SSL certificates and private-public key exchange pairs. SSL offload and SSL inspection features require the servers to share their secret keys to be able to decrypt the SSL traffic.

The SSL proxies control Secure Sockets Layer – SSL traffic -to ensure secure transmission of data between a client and a server. The SSL proxy is transparent, which means it performs SSL encryption and decryption between the client and the server. The SSL proxy also reproduces server certificates so the server can make a secure (SSL) or unsecure (HTTP) connection to a web server. HTTPS proxies use the SSL layer to encrypt any information going between our endpoint and the website, service or server you want to access. Proxy servers can be used to block the use of cookies, prevent websites from knowing our geographical location and they can also be used to block access to specific types of websites.

In most applications, the proxy server actually acts as an intermediate certificate authority throughthe system. It will be important to ensure that all devices on the system have the correct certificateinstalled before using the SSL on proxy servers as the intermediate certificate may not be trusted.

1.1 SSL and TLS Protocols:

SSL and TLS are both cryptographic protocols that provide authentication and data encryption between servers, machines and applications operating over a network (e.g. a client connecting to a web server). SSL is the predecessor to TLS.SSL was originally developed by Netscape and first came onto the scene way back in 1995 with SSL 2.0 (1.0 was never released to the public). Version 2.0 was quickly replaced by SSL 3.0 in 1996 after a number of vulnerabilities were found. When the next version of the protocol was released in 1999, it was standardized by the Internet Engineering Task Force (IETF) and given a new name: Transport Layer Security, or TLS. Transport Layer Security (TLS) is a new version of the Secure Sockets Layer version 3 (SSLv3) protocol, which is no longer recommended for use due to its security vulnerabilities. It provides confidentiality, data integrity, non-repudiation, replay protection, and authentication through digital certificates directly on top of the TCP protocol.

The TLS protocol is currently used for securing the most common network protocols, such asHTTP, FTP, and SMTP, and is part of Voice over Internet Protocol (VoIP) and Virtual Private Network (VPN) protocols.

1.2 SSL-TLS HANDSHAKE

The TLS connection can be divided into two phases:

- an initial handshake
- and application data transfer

- The initial handshake begins with a *Client Hello* message identifying which protocolversion is used, the cipher suite list, and extensions.
- The following messages of the initial handshake are used for peer authentication using X.509 certificates and shared secret establishment based on agreed parameters.
- All TLS messages exchanged during the initial handshake are not encrypted until shared keys are established and confirmed by *Finished* messages.
- The TLS protocol consists of configurable cryptographic algorithms and different sub-protocols which form a layered design.

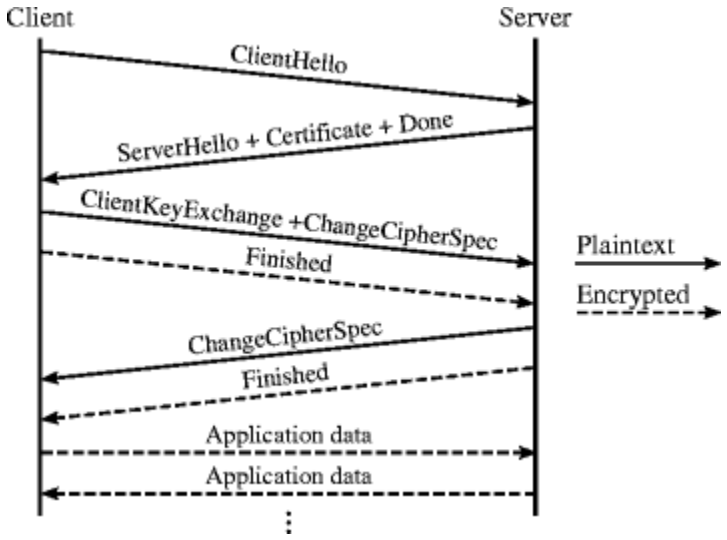


Fig: 1.1 SSL-TLS HANDSHAKE

1.3 What are SSL Certificates and Why do I Need Them?

SSL certificates are an essential component of the data encryption process that make internettransactions secure. They are digital passports that provide authentication to protect the confidentiality and integrity of website communication with browsers.

The SSL certificate's job is to initiate secure sessions with the user's browser via the securesockets layer (SSL) protocol. This secure connection cannot be established without the SSLcertificate, which digitally connects company information to a cryptographic key.

1.4 How SSL Certificates Work?

- A browser or server attempts to connect to a website (i.e. a web server) secured with SSL. The browser/server requests that the web server identify itself.
- The web server sends the browser/server a copy of its SSL certificate.
- The browser/server checks to see whether or not it trusts the SSL certificate. If so, it sends a message to the web server.
- The web server sends back a digitally signed acknowledgment to start an SSL encrypted session.
- Encrypted data is shared between the browser/server and the web server.

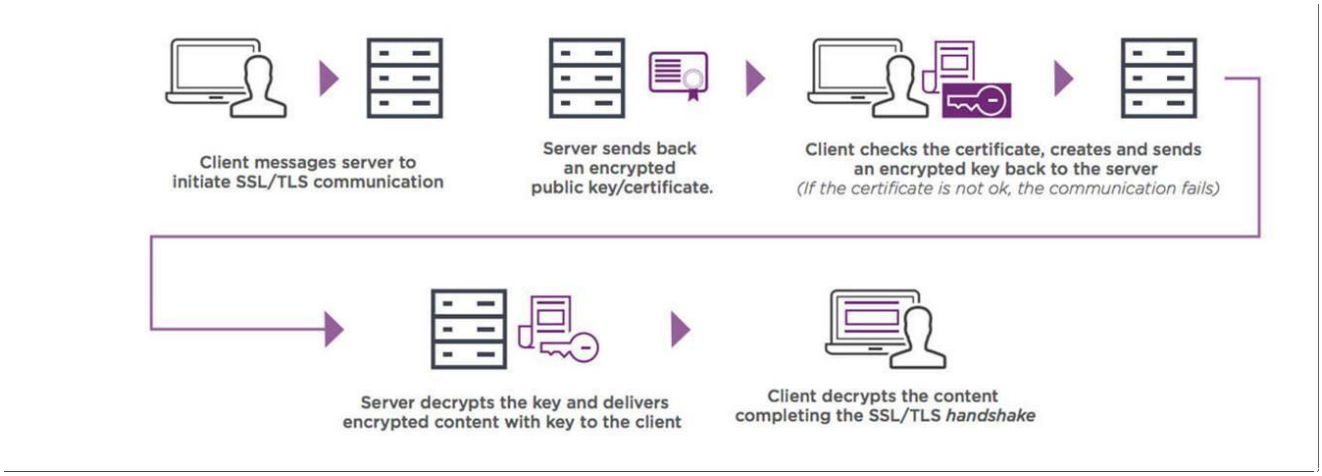


Fig: 1.2 Working of SSL certificates

2. LITERATURE SURVEY

Analysis and Literature Review of IEEE 802.1x (Authentication) Protocols:

International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-3, Issue-5, June 2014 (Umesh Kumar , Praveen Kumar, Sapna Gambhir)

- The IEEE 802.11 standard is most popular standard for WLANs.802.1x protocol known as Port based protocol and proposed by IEEE. Its main task is to solve WLAN user authentication. Authentication is the process of verifying user's identities when they want to use the server resources.
- ***Jhy-Cheng Chen and Yu-Ping Wang*** mentioned implementation of different type of EAP (Extensible authentication Protocol) techniques. Implementing these techniques is quite complex but in this paper it is shown that by the help of WIRE 1x, it is easy to deploy.
- It is an OPEN SOURCE implementation of client side because if client side is strong then communication is more secure. These WIRE 1x easily work with windows and it supports nearly all of the Authentication mechanisms defined in EAP. WIRE 1x also provide secure manner of communication for user who is using WLAN. It is release in worldwide on 18th June 2003.

(***References***: Jyh-Cheng and Yu-Ping Wang "Extensible Authentication Protocol (EAP) and IEEE 802.1x: Tutorial and Empirical Experience" IEEE DEC 2005 ISSN 0163-6804/05)

- ***Bahareh Shojaie, Iman Saberi and Seyyed Morteza Alavi*** has done study of EAP-TLS to compare two type of Extensible Authentication Protocol – transport layer security (EAP-TLS) so that another technique can be provided by using cryptographic methods. The new technique used Elliptical Curve Digital Signature Algorithm (ECDSA) and SHA-256 to provide very high security and also high performance. It also compares it with the existing EAP-TLS method and show that the new techniques provide strong security, high speed and more efficiency by using same level of memory as compare to EAP-TLS.

(***References***: Bahareh Shojaie, Iman Saberi, Mazleena Salleh, Mahan Niknafskermani and Seyyed Morteza Alavi "Improving EAP-TLS Performance Using Cryptographic Methods" International conference on computer & Information Science 2012)

3. SOFTWARE REQUIREMENTS SPECIFICATIONS

3.1 Introduction

3.1.1 Purpose

The purpose of documenting the SRS (Software Requirements Specification) is that it is the basisfor our entire project. It lays the framework that the team involved in development will follow. It is used to provide critical information to the team. SRS helps to ensure that requirements are fulfilled and it also helps to make decisions about life cycle of project. Writing an SRS can also minimize overall development time and costs.

3.1.2 Document Conventions

The details from abstract have been inherited for completing the details of this SRS document.The font style and size that are used in this document are as follows

Font style	Times new roman	Normal	size 12	For text
Font style	Times new roman	Bold and Italics	size 12	For Sub headings
Font style	Times new roman	Bold	size 14	For headings

The conventions that this document is using in the form of acronyms, abbreviations and key terms
that need description for better understanding are as follows-

SSL-Secure Sockets Layer
TLS-Transport Layer Security
HTTPS-Hyper-text Transfer Protocol Secure
MITM-Man in the Middle
CSR-Certificate Revocation List

3.1.3 Intended Audience and Reading Suggestions

This Software Requirements document is intended for: −Developers who can review project's capabilities and more easily understand where their efforts should be targeted to improve or addmore features to it.

End users can also use this document to know about SSL proxy and its usage for analyzing encrypted traffic. This document also facilitates the use within Internet.

3.1.4 Product Scope

The scope of the project is to analyze and monitor the data that is sent from client to server and server to client within LAN. This project presents the analysis of HTTPS-HTTP over SSL/TLS, the most common encrypted network traffic protocols. In a communication encrypted by SSL/TLS, the hosts have to first agree on encryption methods and their parameters. SSL proxy provides secure transmission of data between a client and a server through a combination of Authentication, Confidentiality, Integrity.

3.2. Overall Description

3.2.1 Product Perspective

It is possible to use a proxy server to protect the user's information from attacks or from beingvisible.

SSL split is designed to transparently terminate connections that are redirected to it using anetwork address translation engine. SSL split then terminates SSL/TLS and initiates a new SSL/TLS connection to the original destination address, while logging all data transmitted.

3.2.2 Product Functions

- For the server, SSL proxy acts as a client—Because SSL proxy generates the shared pre-master key, it determines the keys to encrypt and decrypt.
- For the client, SSL proxy acts as a server—SSL proxy first authenticates the original server and replaces the public key in the original server certificate with a key that is known to it. It then generates a new certificate by replacing the original issuer of the certificate with its own identity and signs this new certificate with its own public key (provided as a part of the proxy profile configuration). When the client accepts such a certificate, it sends a shared pre-master key encrypted with the public key on the certificate. Because SSL proxy replaced the original key with its own key, it is able to receive the shared pre-master key. Decryption and encryption take place in each direction (client and server), and the keys are different for both encryption and decryption.
- Proxy servers can be used to block the use of cookies, prevent websites from knowing your geographical location and they can also be used to block access to specific types of websites.
- In some cases, they can be used to block social media sites from access on business computers or to prevent people from going to sites with malware or other types of viruses.

- It is possible to set this up so that those access sites on the internet from the internal network will go through the HTTPS proxy servers. This will be the same as saying it is using the SSL on proxy servers to access the website and then receiving the response.
- Advanced IT managers can set up SSL on proxy servers to exclude specific websites or specific categories of websites. These could be sites already known to be secure or to allow access to one or more websites within a category or group of sites that are blocked by the proxy server.
- In addition to the security threat, encrypted traffic makes it difficult for IT to assess bandwidth usage and apply intelligent content control policies to ensure maximum user productivity.

3.2.3 Operating Environment

We have installed CentOS version 7.6 on our virtual machines which has specifications of 4 GBRAM and 40 GB hard disk drive space and software and tools like Wireshark to study the

3.2.4 Design and Implementation Constraints

- Client can only verify the connection between itself and the SSL-split.
- Protocols and ciphers that SSL-split negotiates may be invisible to the client.
- Some SSL-inspecting software fails to validate the certificates of systems that it connects to. In some cases, the software may attempt to perform some validation of the certificate, but the validation may be insufficient. (*Risks: Clients cannot know if they are connected to a legitimate site or not.)*
- Users of client systems may not know why certificate validation failed, or even if it failed.
- Not every web client may receive certificate validation. Various web browser versions and non-browser software may be exempt from validation.

3.2.5 User Documentation

The following references have been used for installation steps that can be used for a normal user:https://www.digitalocean.com/community/tutorials/how-to-create-an-ssl-certificate-on-apache- for-centos-7

https://drive.google.com/file/d/12YaGIGs0-xfpqMNAY3rzUbIyed-Tso8W/view

https://www.roe.ch/SSLsplit

https://blog.heckel.io/2013/08/04/use-sslsplit-to-transparently-sniff-tls-ssl-connections/

3.2.6 Assumptions and Dependencies

The pre-requisites that are needed for successful usage of project and the dependencies that theproject has on external factors are:

- The user should have sufficient knowledge about computers.
- The computer should have internet connection and Internet server capabilities.
- The client should be available in the same network.
- The gateway IP of the client is configured as IP of the SSL proxy device.

3.3 External Interface Requirements

3.3.1 User Interfaces

The user interface for the software shall be compatible to any browser such as InternetExplorer, Mozilla or Chrome by which user can access to the system.

3.3.2 Hardware Interfaces

We require a Desktop computer with 1 GB RAM,256 HDD, Intel Core i3 processor,100 Base-TEthernet Card. Since the application must run over the internet, all the hardware shall require toconnect internet will be hardware interface for the system (e.g. Modem, WAN – LAN, EthernetCross-Cable.

3.3.3 Software Interfaces

The mandatory software that have been used for the creation, development and the deploymentfor this project are:

- OpenSSL
- SSL split
- Operating System: Windows, Linux, Mac

SSL split depends on the OpenSSL, libevent 2.x, libpcap and libnet 1.1.x libraries by default;libpcap and libnet are not needed if the mirroring feature is omitted.
SSL/TLS features and compatibility greatly depend on the version of OpenSSL linked against SSLsplit currently supports the following operating systems and NAT mechanisms:

- FreeBSD: pf rdr and divert-to, ipfw fwd, ipfilter rdr
- OpenBSD: pf rdr-to and divert-to
- Linux: netfilter REDIRECT and TPROXY
- Mac OS X: pf rdr and ipfw fwd

4. ARCHITECTURE

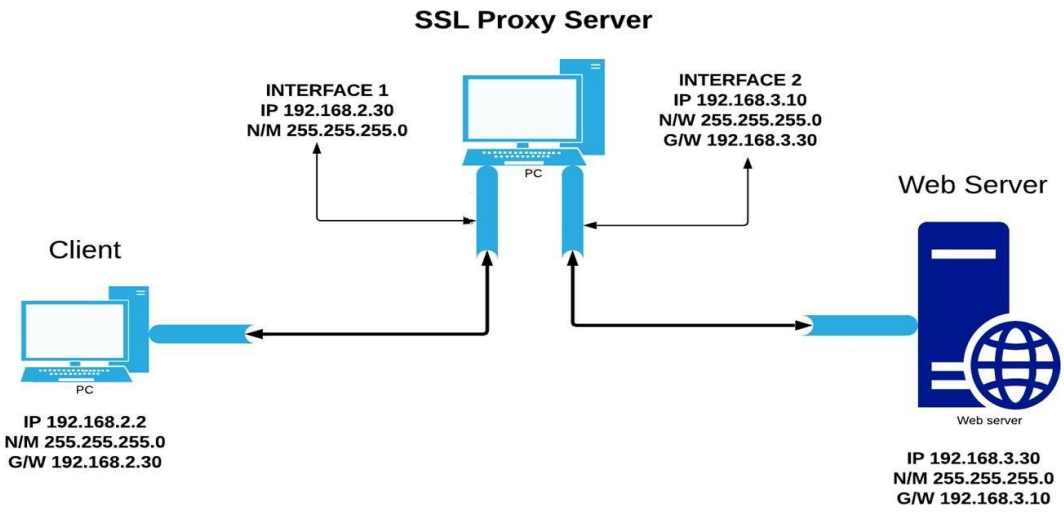


fig 4.1 SSL Proxy deployment diagram

We can infer the following from the above diagram:

- 3-system hierarchy is created where the configurations are made with following parameters:

Client side:

IP address: 192.168.2.2

Subnet Mask:255.255.255.0/24

Gateway:192.168.2.30

- Proxy side:

Interface 1- IP address:192.168.2.30

Subnet Mask: 255.255.255.0/24

Interface 2-IP address: 192.168.3.10

Subnet Mask: 255.255.255.0/24

Gateway:192.168.3.30

- Server side:

IP address: 192.168.3.30

Subnet Mask:255.255.255.0/24

Gateway:192.168.3.10

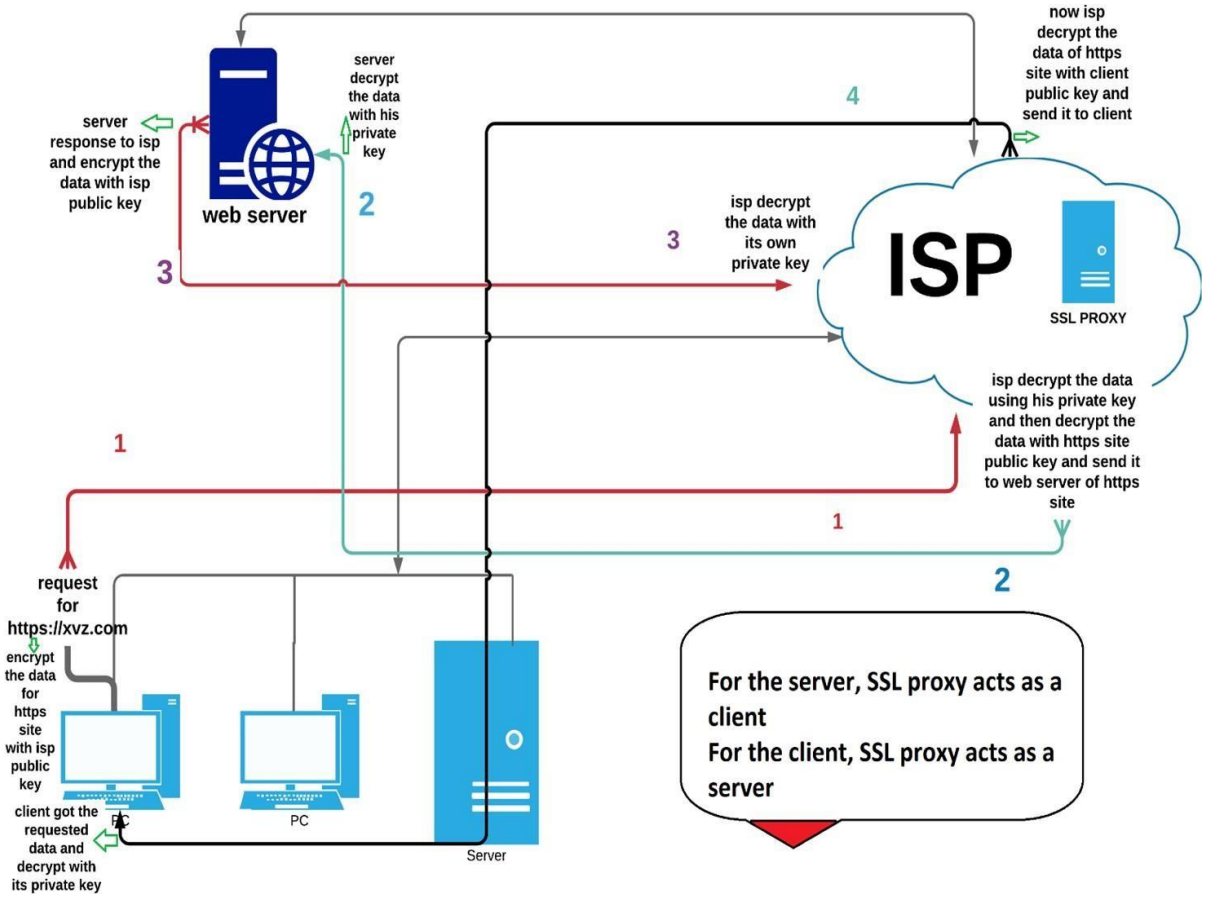


fig 4.2 architecture of SSL Proxy

- For the server, SSL proxy acts as a client—Because SSL proxy generates the shared pre-master key, it determines the keys to encrypt and decrypt.
- For the client, SSL proxy acts as a server—SSL proxy first authenticates the original server and replaces the public key in the original server certificate with a key that is known to it. It then generates a new certificate by replacing the original issuer of the certificate with its own identity and signs this new certificate with its own public key (provided as a part of the proxy profile configuration). When the client accepts such a certificate, it sends a shared pre-master key encrypted with the public key on the certificate. Because SSL proxy replaced the original key with its own key, it is able to receive the shared pre-master key. Decryption and encryption take place in each direction (client and server), and the keys are different for both encryption and decryption.

- SLsplit picks up SSL connections and pretends to be the server the client is connecting to. To do so, it dynamically generates a certificate and signs it with a the private key of a CA certificate that the client must trust.
- SSLsplit connects to the server just like a normal client — forwarding all the traffic the actual client writes on the SSL socket.

5.IMPLEMENTATION

This chapter deals with the implementation and configuration of the SSL-split along with thedeployment of server and client so as to establish connection between server and client with intermediary SSL-split.

As we have seen the architecture of SSL-split in chapter-4, the deployment diagram needs aserver and a client.

PREREQUISITES

We will need access to a CentOS 7 server with a non-root user that has ***sudo*** privileges. We willalso need to have Apache installed in order to configure virtual hosts for it. If we haven't alreadydone so, we can use ***yum*** to install Apache through CentOS's default software repositories:

Step-1 Installation of apache web server with SSL certificate

sudo yum install httpd

Step-2 Next, enable Apache as a CentOS service so that it will automatically start after a reboot:

sudo systemctl enable httpd.service

After these steps are complete, we can log in as our non-root user account through SSH.

Step-3 Install mod ssl

This mod ssl will automatically be enabled during installation, and Apache will be able to startusing an SSL certificate after it is restarted. We don't need to take any additional steps for ***modssl*** to be ready for use.

sudo yum install mod_ssl

Step-4 Create a new certificate

Our Apache web server is ready to use encryption, now we can generate a new SSL certificate. The certificate will store some basic information about our apache site, and will be accompaniedby a key file that allows the server to securely handle encrypted data.

we need to create a new directory to store our private key

sudo mkdir /etc/ssl/private

Our files kept within this directory must be kept strictly private, so we will modify the permissions to make sure only the root user has access

sudo chmod 700 /etc/ssl/private

Step-5 Now that we have a location to place our files, we can create the SSL key and certificate files with openssl:

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt

openssl*:* This is the basic command line tool for creating and managing OpenSSL certificates, keys, and other files.

req -x509*:* This specifies that we want to use X.509 certificate signing request (CSR) management. The "X.509" is a public key infrastructure standard that SSL and TLS adhere to for key and certificate management.

-nodes*:* This tells OpenSSL to skip the option to secure our certificate with a passphrase. We need Apache to be able to read the file, without user intervention, when the server starts up. A passphrase would prevent this from happening, since we would have to enter it after every restart.

-days 365*:* This option sets the length of time that the certificate will be considered valid. We set it for one year here.

-newkey rsa:2048: This specifies that we want to generate a new certificate and a new key at the same time. We did not create the key that is required to sign the certificate in a previous step, so we need to create it along with the certificate. The rsa:2048 portion tells it to make an RSA key that is 2048 bits long.

-keyout*:* This line tells OpenSSL where to place the generated private key file that we are creating.

-out*:* This tells OpenSSL where to place the certificate that we are creating.

The full list of prompts will look something like this:

Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:MH
Locality Name (eg, city) [Default City]:PUNE
Organization Name (eg, company) [Default Company Ltd]:CDAC
Organizational Unit Name (eg, section) []:DITISS
Common Name(eg,your name or your server's hostname) []:GROUP25.COM
Email Address []:GROUP25@GMAIL.COM

Both of the files that we created will be placed in the appropriate subdirectories of the ***/etc/ssl*** directory. While we are using OpenSSL, we should also create a strong Diffie-Hellman group,which is used in negotiating Perfect Forward Secrecy with clients.

We can do this by following command:

sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048

We are doing this configuration in centos 7, so since the version of Apache that ships with CentOS 7 does not include the ***SSLOpenSSLConfCmd*** directive, we will have to manuallyappend the generated file to the end of our self-signed certificate.

cat /etc/ssl/certs/dhparam.pem | sudo tee -a /etc/ssl/certs/apache-selfsigned.crt

This ***apache-selfsigned.crt*** file now have both the certificate and the generated Diffie-Hellman group.

Set Up the Certificate

We now have all of the required components of the finished interface. The next thing to do is toset up the virtual hosts to display the new certificate.

Now we did some changes in ssl configuration file because we need to ensure that our SSLcertificate is correctly applied to our site.

Uncomment the ***DocumentRoot*** line and replace with your domain name or server IP address

<VirtualHost _default_:443>

DocumentRoot "/var/www/html/"
ServerName DomainName / IP of server:443

After that find the ***SSLProtocol*** and ***SSLCipherSuite*** lines and either delete them or comment them out. The configuration which we will be pasting will offer more secure settings than the default included with CentOS's Apache:

#SSLProtocol all -SSLv2

#SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5:!SEED:!IDEA

Now we need to find the ***SSLCertificateFile*** and ***SSLCertificateKeyFile*** lines and change them to the directory we made at ***/etc/httpd/ssl***:

SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt

SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key

At the end of the configuration file we need paste the following lines to complete the configuration of apache web server with SSL Certificate

 # Begin copied text

 # from https://cipherli.st/

 # and https://raymii.org/s/tutorials/Strong_SSL_Security_On_Apache2.html

SSLCipherSuite EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
SSLProtocol All -SSLv2 -SSLv3
SSLHonorCipherOrder On
 # Disable preloading HSTS for now. You can use the commented out header line that includes
 # the "preload" directive if you understand the implications.
#Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains; preload"
Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains"
Header always set X-Frame-Options DENY
Header always set X-Content-Type-Options nosniff
 # Requires Apache >= 2.4
SSLCompression off
SSLUseStapling on
SSLStaplingCache "shmcb:logs/stapling-cache(150000)"

 # Requires Apache >= 2.4.11
 # SSLSessionTickets Off

Now we need to modify the Unencrypted Virtual Host File to Redirect to HTTPS.

To redirect all traffic to be SSL encrypted, we created a file ending with ***.conf*** in the ***/etc/httpd/conf.d*** directory

Inside we need to create a ***VirtualHost*** block to match requests on port 80. Inside, we are usingthe ***ServerName*** directive to again match our domain name or IP address. Then add ***Redirect*** tomatch any requests and send them to the ***SSL VirtualHost.***

<VirtualHost *:80>

ServerName domain name or IP
Redirect "/" "https://domain name or IP/"

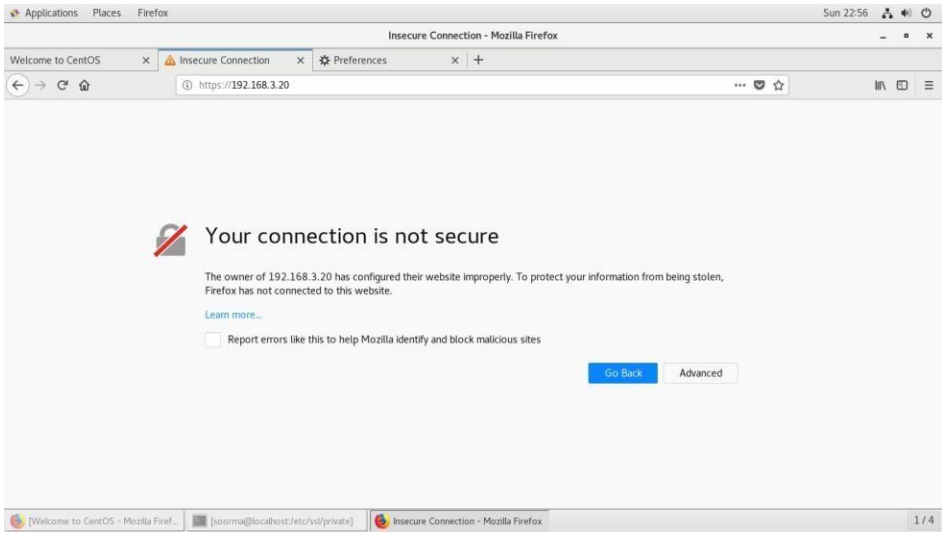
</VirtualHost>

If have an **iptables** firewall running, the commands we need to run are highly dependent on ourcurrent rule set. For a basic rule set, we add HTTP and HTTPS access by following commands

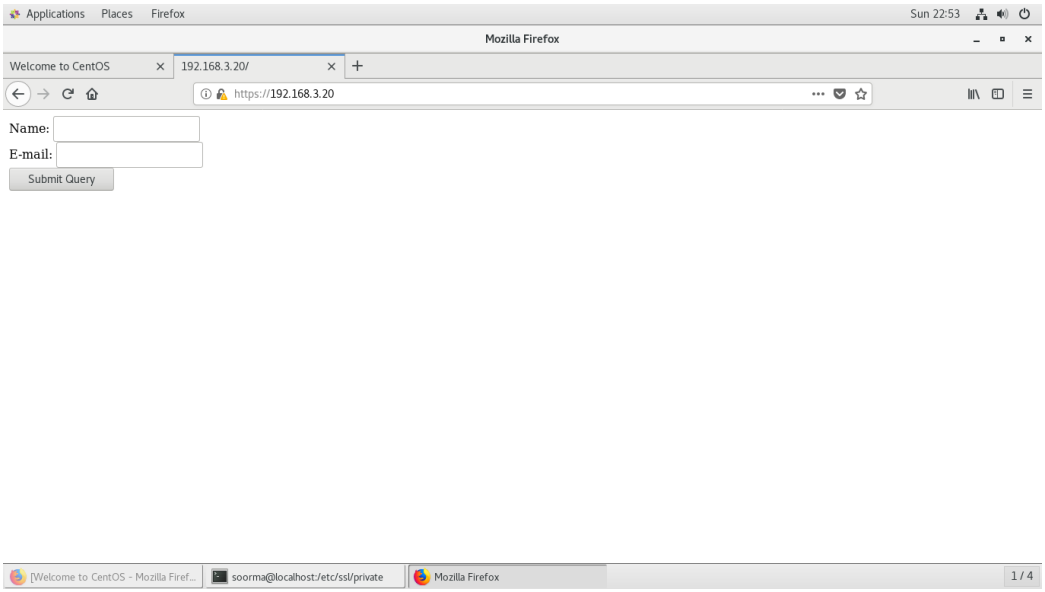
sudo iptables -I INPUT -p tcp -m tcp --dport 80 -j ACCEPT

sudo iptables -I INPUT -p tcp -m tcp --dport 443 -j ACCEPT

In web browser, we need to search domain name or IP with ***https://*** to see our certificate in our apache web server



Its Showing your connection is not secure because we are using a self-signed certificate after clicking on ***Advanced*** add the extension ,then our site will be like that shown in following image:



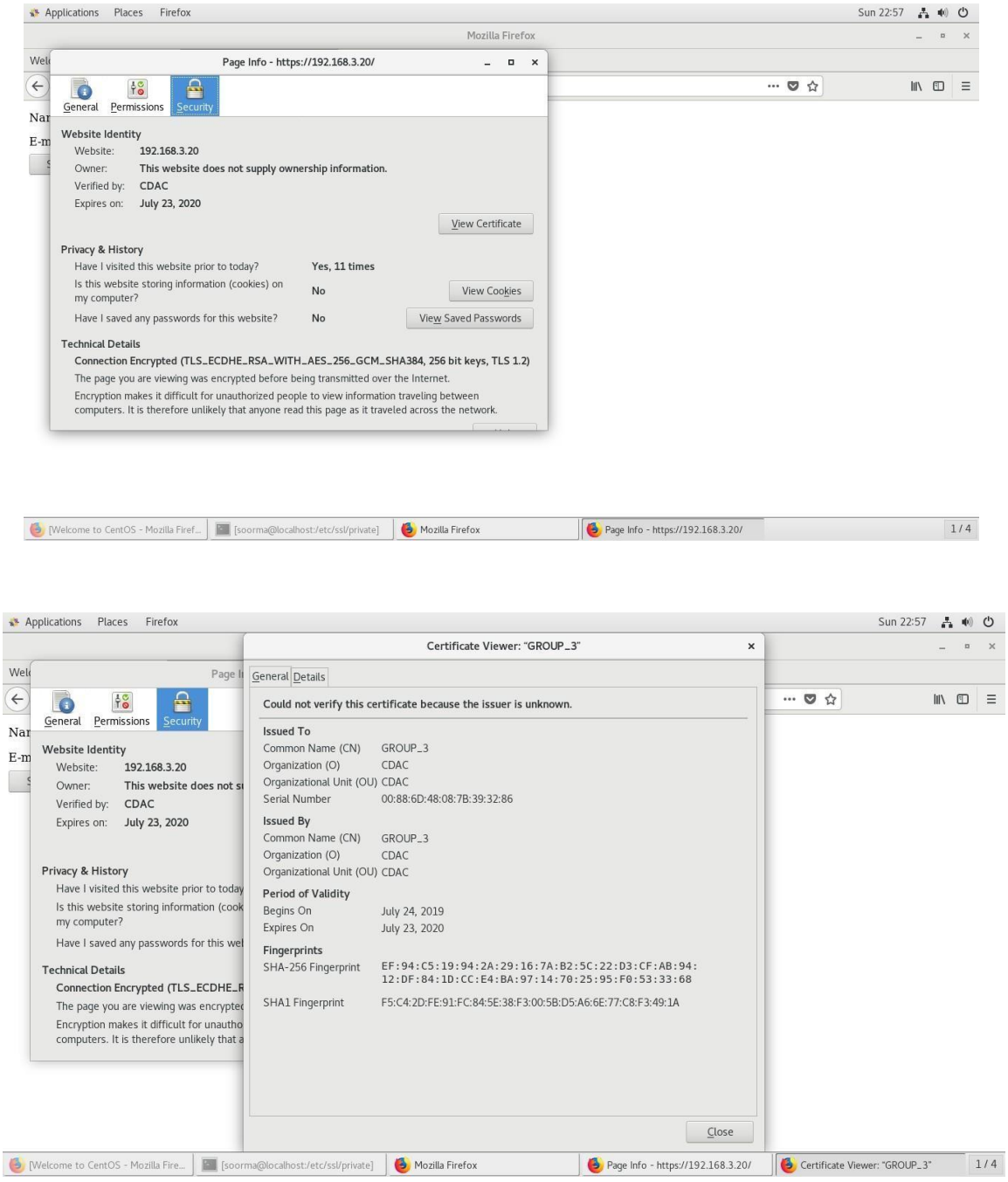


fig: The self-signed certificate we created using openssl

SSLsplit to transparently sniff TLS/SSL connections – including non-HTTP(S) protocols

SSL-split is a generic transparent TLS/SSL proxy for performing man-in-the-middle attacks onall kinds of secure communication protocols. Using SSL-split, one can intercept and save SSL-based traffic and thereby listen in on any secure connection.

Installation

To download and compile SSLsplit, we have to run the following commands:

wget http://mirror.roe.ch/rel/sslsplit/sslsplit-0.4.7.tar.bz2
bunzip2 sslsplit-0.4.7.tar.bz2
tar xvf sslsplit-0.4.7.tar
cd sslsplit-0.4.7
yum install openss-devel.x86_64 libevent-devel.x86_64,
yum install libpcap-devel.x86_64 libnet-devel.x86_64
make
mkdir /tmp/sslsplit
cd /tmp/sslsplit && mkdir logdir

These commands download and extract the source code (wget, bunzip2, tar), install necessary dependencies (yum install), and then compile it using make.

The temporary directory created at /tmp/sslsplit/logdir is later used to dump the connection log file and the raw data of the incoming and outgoing SSL sockets.

Create and install root CA certificate

For SSLsplit to act as a middle man for SSL connections, it needs to be able to generate and signcertificates that the victim trusts. In order to do so, the victim must have the attacker's root CA certificate in its trust store.

We can install it by following command:

[root@localhost sslsplit-0.5.4]# openssl genrsa -out ca.key 4096
[root@localhost sslsplit-0.5.4]# openssl req -new -x509 -days 1826 -key ca.key -out ca.crt

The first command generates a 4096-bit RSA private key in PEM format (ca.key), and the second command uses this private key to generate a self-signed root CA certificate (ca.crt). Both are needed by SSLsplit later, but only the certificate file needs to be installed in the browser or operating system of the user/client.

Enable IP forwarding and NAT engine (iptables)

In this example, SSLsplit will be running on two ports: 8080 for non-SSL TCP connections suchas HTTP, SMTP or FTP, and 8443 for SSL connections such as SMTP over SSL, HTTPS, etc. Inorder to forward packets arriving at the attacker's machine to these internal ports, the NAT engine in iptables can be used.

sysctl -w net.ipv4.ip_forward=1
iptables -t nat -F
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8443
iptables -t nat -A PREROUTING -p tcp --dport 587 -j REDIRECT --to-ports 8443
iptables -t nat -A PREROUTING -p tcp --dport 465 -j REDIRECT --to-ports 8443
iptables -t nat -A PREROUTING -p tcp --dport 993 -j REDIRECT --to-ports 88443
iptables -t nat -A PREROUTING -p tcp --dport 5222 -j REDIRECT --to-ports 800

The commands above first enable IP forwarding (sysctl ...) to enable the system's router functionality. After running this command, Linux will forward IP packets not meant for the local machine to its standard/default gateway, thereby acting as a router.
To prevent Linux from forwarding everything right away, NAT rules can be defined. In this example, certain packets are redirected to the local port 8080 and 8443. Packets for the plain text traffic on ports HTTP (80) and WhatsApp (5222) are redirected to port 8080, and packets for SSL-based traffic on ports HTTPS (443), IMAP over SSL (993), SMTP over SSL (465 and 587) are redirected to port 8443

Run SSLsplit

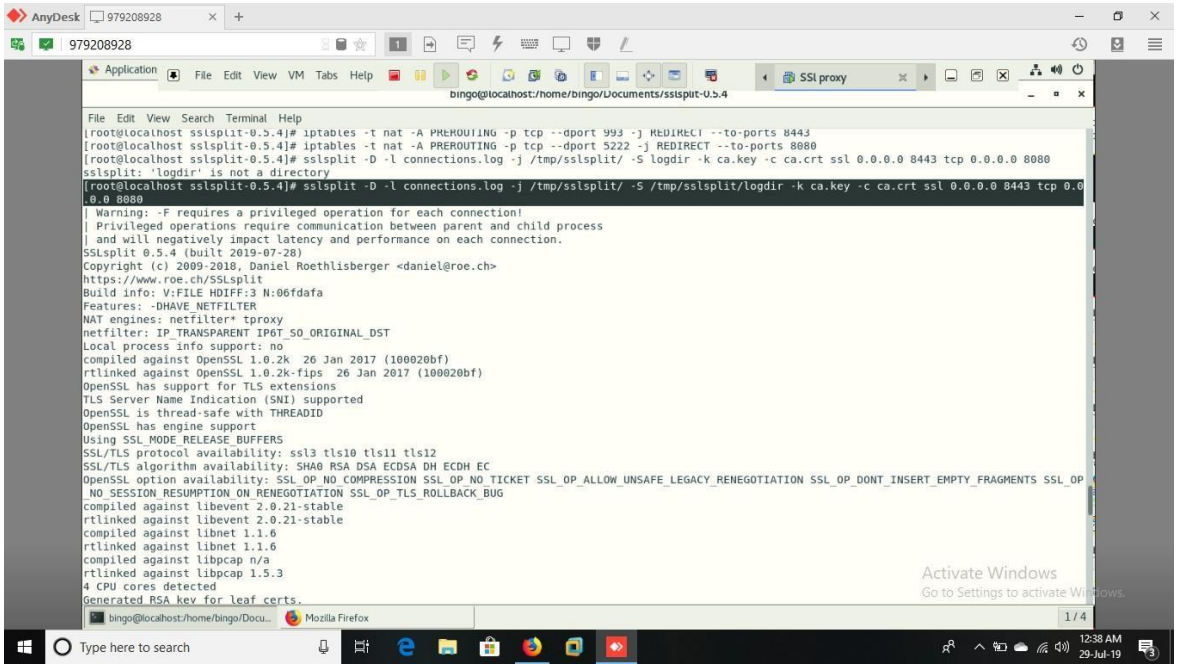
Once the IP forwarding is active and packets are being forwarded to the relevant ports, you can start SSLsplit. That sounds easier than it actually is, because SSLsplit is a very powerful tool andtherefore very flexible.

Following command is used to intercept the traffic

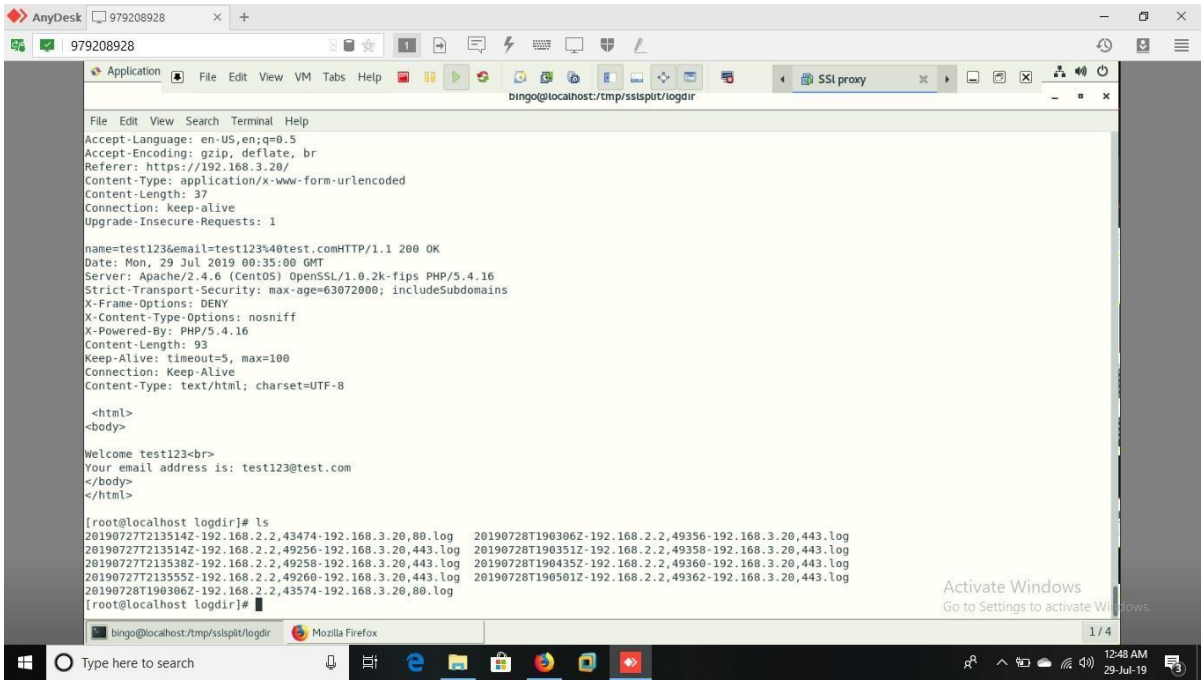
[root@localhost sslsplit-0.5.4]# sslsplit -D -l connections.log -j /tmp/sslsplit/ -S /tmp/sslsplit/logdir -k ca.key -c ca.crt ssl 0.0.0.0 8443 tcp 0.0.0.0 8080

Where this command starts SSLsplit in debug mode (-D, runs in foreground, no daemon, verboseoutput) and outputs connection attempts in the log file “connections.log” (-l ..). The actual content of the connections is written to the “/tmp/sslsplit/logdir/” (-j and -S) — each incoming/outgoing TCP stream of each connection in a separate file.

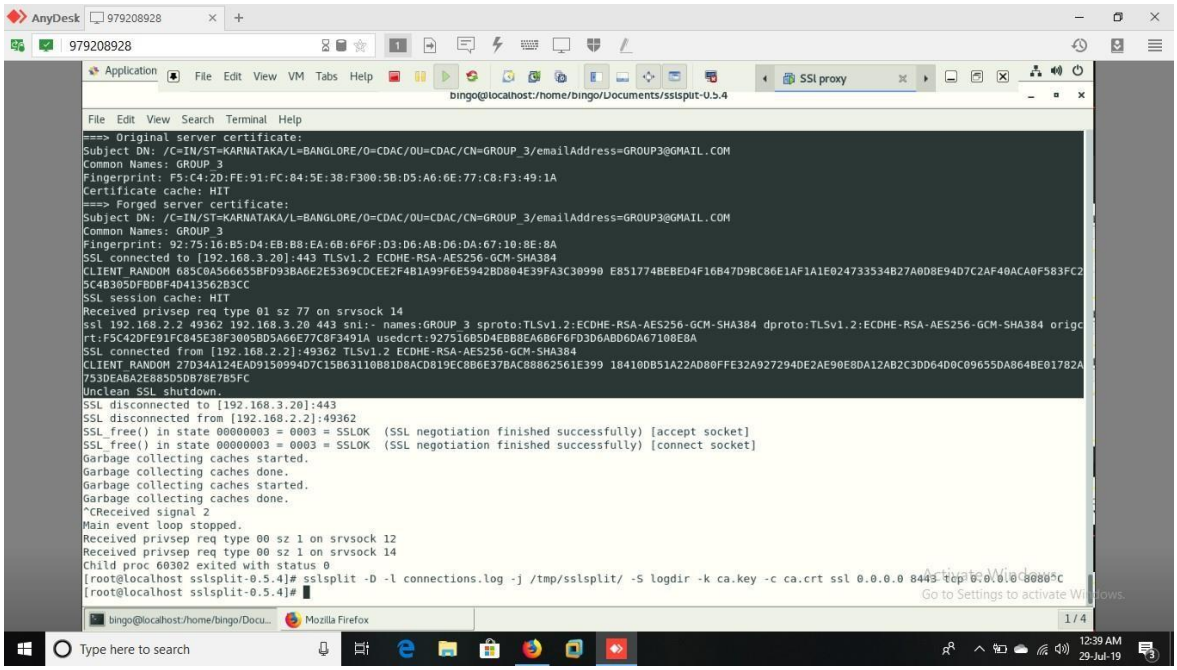
The output will be like below image

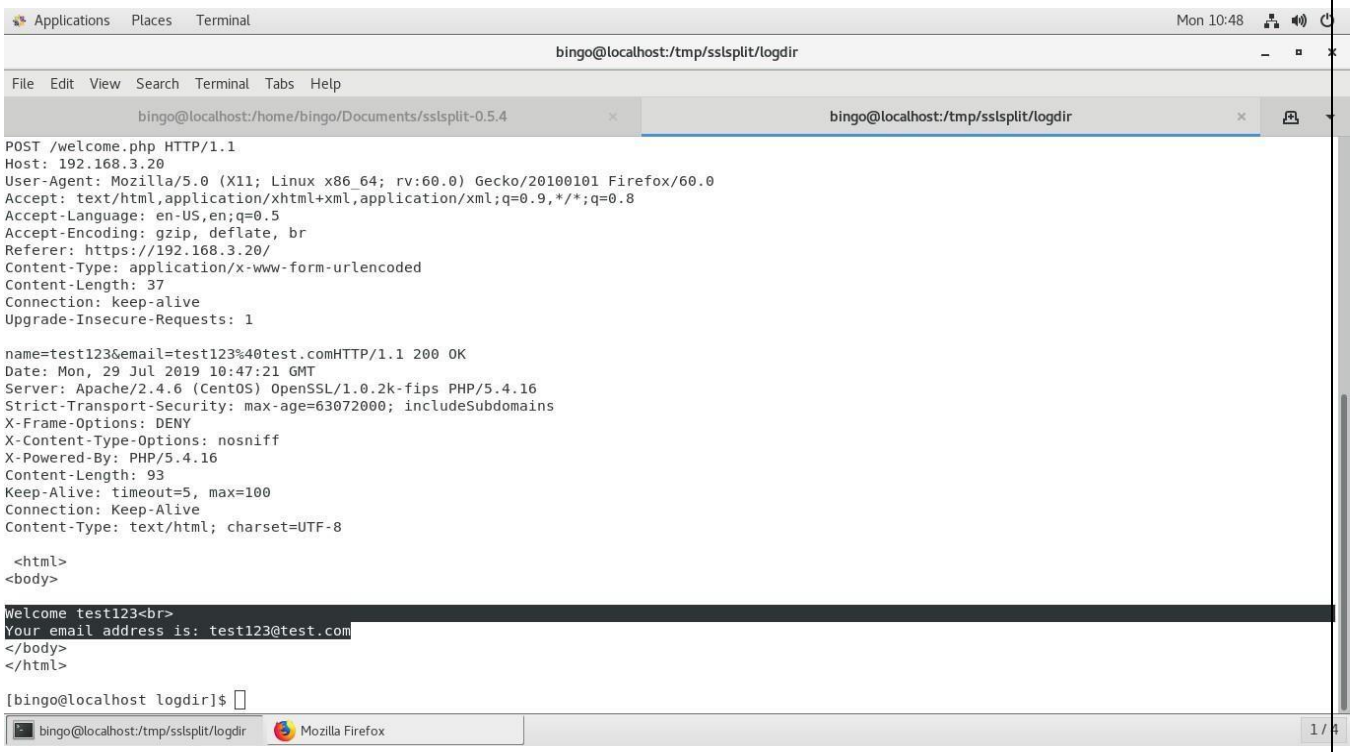


After running SSLsplit for a while, there will be quite a few files in the log directory — one for each connection or TCP socket between client and server

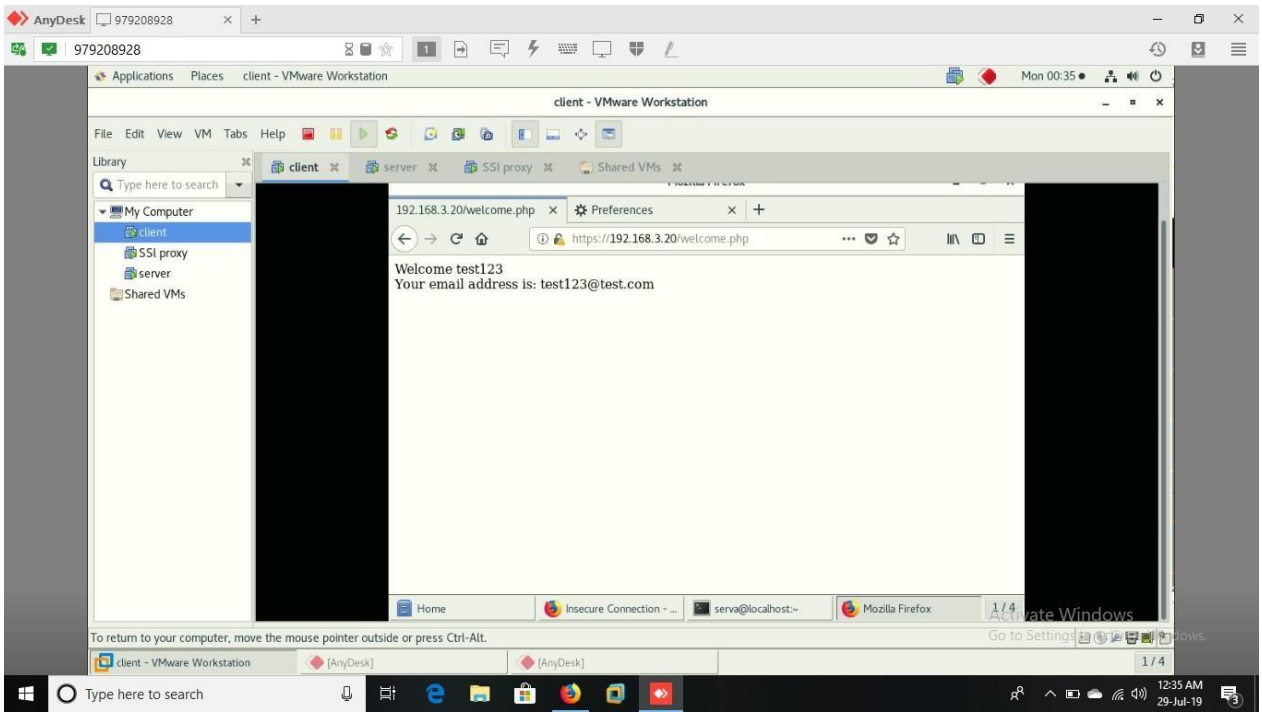


Forging cert with sslsplit



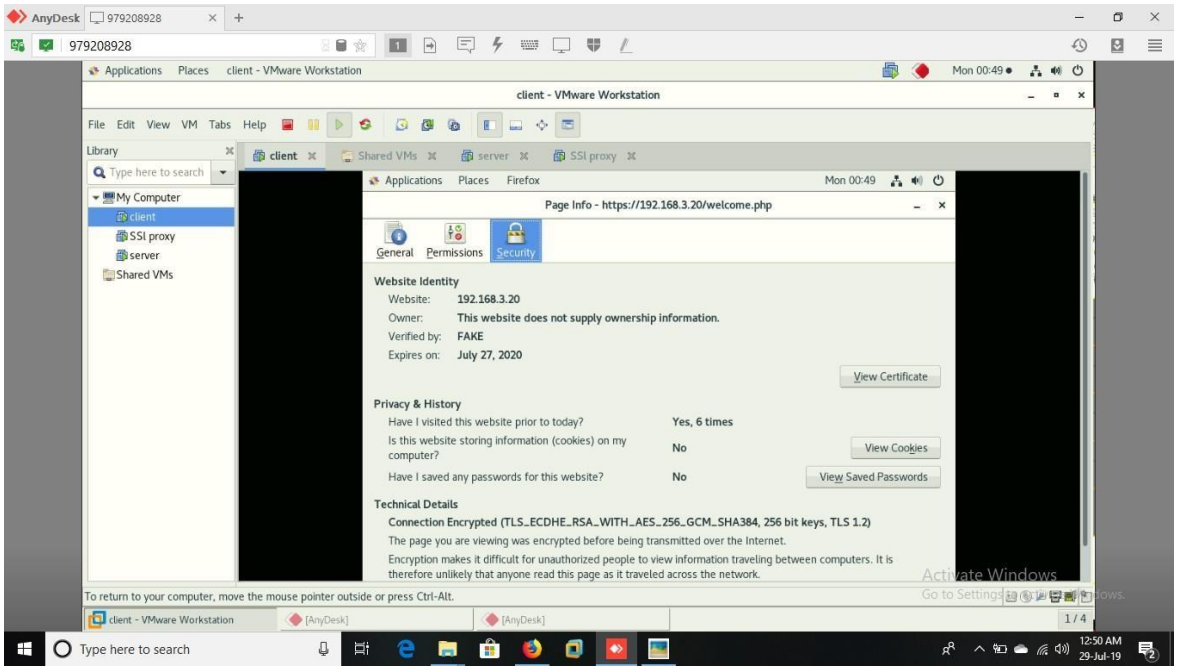


On client browser



SSL certificate by SSLsplit

->On client browser



6.CONCLUSION

Hence, we are able to decrypt the non-HTTPS and HTTPS traffic into plain text through SSL- split. The use of this proxy will also be useful to the industries who is seeking to monitor the encrypted traffic. Network administrators, researchers and security experts may find this usefulto detect future vulnerabilities in the implementation SSL/TLS in their organizations.

7.REFERENCES

https://link.springer.com/article/10.1186/s13635-016-0030-7

https://www.juniper.net/documentation/en_US/cso3.3/topics/concept/cp-ssl-proxy-overview.html

https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-ssl-proxy.html

https://github.com/sonertari/SSLproxy

https://origin-symwisedownload.symantec.com/library/SYMWISE/ENTERPRISE/

https://www.tecmint.com/install-apache-on-centos-7/

https://www.digitalocean.com/community/tutorials/how-to-create-an-ssl-certificate-on-apache-for-centos-7

https://www.digitalocean.com/community/tutorials/how-to-create-an-ssl-certificate-on-apache-for-centos-7

https://drive.google.com/file/d/12YaGIGs0-xfpqMNAY3rzUbIyed-Tso8W/

https://www.roe.ch/SSLsplit

https://blog.heckel.io/2013/08/04/use-sslsplit-to-transparently-sniff-tls-ssl-connections/