Tryhackme: Network Security Protocols

Daniel Schwarzentraub · Follow
4 min read · Sep 28, 2023

Task 1: Introduction

A network protocol specifies how two devices, or more precisely processes, communicate with each other.

A network protocol is a pre-defined set of rules and processes to determine how data is transmitted between devices, such as end-user devices, networking devices, and servers. The fundamental objective of all protocols is to allow machines to connect and communicate seamlessly, regardless of any difference in their internal design, structure, logic, or operation. In analogy, a networking protocol is like a "common language" that helps make communication possible among people with different native languages and from various parts of the globe.

Learning Objective

In this room, we will learn primary protocols essential for network security at each OSI model layer.

Room Prerequisites

Understanding of following topics is recommended before starting the course:

- OSI Model
- HTTP Protocols and Methods
- Principles of Security

Let's begin!

Task 2: Application Layer

In this part, we will learn the core technical concepts of various protocols. However, to proceed further, we must be very clear about the following two ideas, which are the hallmark of any protocol regardless of its functionality:

- Each protocol represents specific layers of the OSI or TCP/IP model and operates as per the functionality of that layer.
- TCP and UDP-based protocols operate on specific network ports.

HTTPS Protocol

Technical Overview - HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is a client-server protocol; responsible for securely sending data between a web server (website) and a web browser (client side). It is an encrypted variant of HTTP which sends data in an unencrypted format.

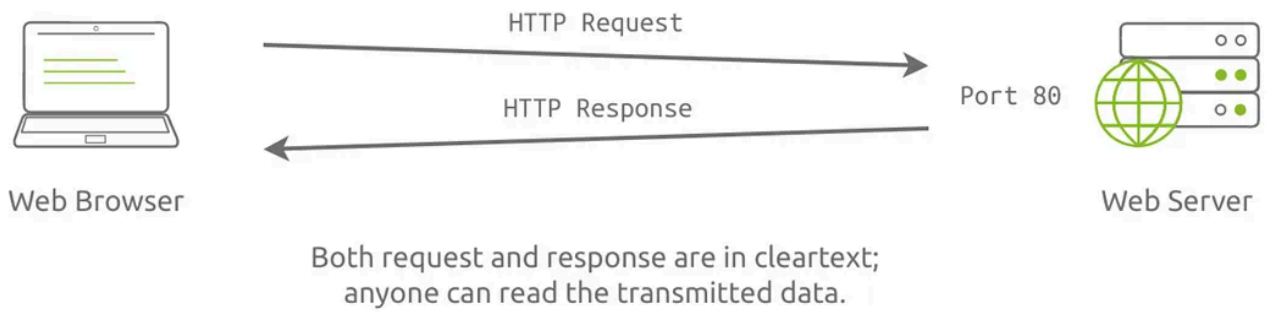
HTTP would be enough to browse a website to learn about a company product; however, HTTPS is a must if you want to provide your credit card details to place an online order. HTTPS was developed to securely share sensitive information, including passwords, contact information and financial information, between web browsers and websites. Without HTTPS, secure online banking and online payment wouldn't have been possible.

Workflow - HTTPS

HTTPS uses its unencrypted counterpart, i.e., HTTP, and adds a layer of encryption. In this case, it is SSL/TLS (Secure Sockets Layer/Transport Layer Security); the rest of the workflow remains the same. So before proceeding forward, we will review how HTTP requests and responses operate in a typical client and server environment.

Request and Response - HTTP

An HTTP request is made by a user agent (a browser or any other application sending requests through a web API (Application Programming Interface)). It is vice versa in the case of response. This request aims to access some resources on the remote web server, which is then responded to by the web server. The figure below shows a web browser sending an HTTP request to a web server, listening at TCP port 80.

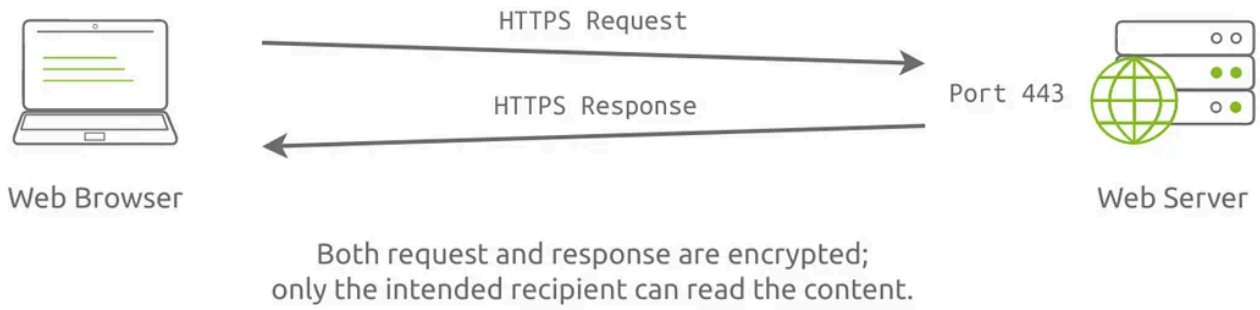


The request might be `GET` to request a web page, an image, or a file. Other HTTP requests include `PUT` and `POST`, which send data to the web server, such as a value or a file. You can read more about HTTP in the HTTP in Detail room.

If an attacker can capture the network packets between the client and the server communicating over HTTP, they will be able to read their content as it is sent in cleartext.

Request and Response - HTTPS

After our quick review of the HTTP request and response workflow, it is convenient to learn about HTTPS. Remember that the "S" in HTTPS is for the extra SSL/TLS layer of encryption added over HTTP.



Even if an attacker can capture the network packets between the client and the server communicating over HTTPS, they will fail to read the contents of the TCP data due to encryption.

Encryption Mechanism of HTTPS

As already mentioned, SSL/TLS provides the encryption layer of HTTPS. It relies on asymmetric encryption (public key cryptography) and symmetric encryption. Asymmetric encryption uses two keys, i.e., public key and private key; its rule is to negotiate the symmetric encryption algorithm and the secret key. The default port of HTTPS is 443. Encryption protects against interception and alteration of data, maintaining the confidentiality and integrity of exchanged traffic.

FTPS Protocol

Technical Overview - FTPS

File Transfer Protocol Secure (FTPS) is a communication protocol which is a refined and secure version of File Transfer Protocol (FTP). Initially, FTP was developed in 1971 and published as RFC 114. Additional improvements and various changes were published in RFC 765 and RFC 959.

FTP was designed as a client-server model; separate control/command and data connections between a client and a server are used, along with a username and password. In FTP, both authentication and data transfer take place in an unencrypted form between the client and the server; however, in FTPS, an encrypted channel is established.

Workflow

FTPS is an extension of FTP, which adds TLS security to commands and data connections. It is necessary to get an overview of FTP to understand FTPS.

Request and Response - FTP

As described earlier, FTP is based on the client-server model. It utilizes the following two communication channels between the client and the server.

- **Control Connection:** In this connection, an FTP client (such as Filezilla and CuteFTP) sends a connection request (authentication) to the remote FTP server at the default FTP port, TCP port 21. As the name implies, a control connection is used for sending and receiving commands and responses.
- **Data Connection:** After authentication, this connection is used for transferring data (files and folders).

FTP Connection Types

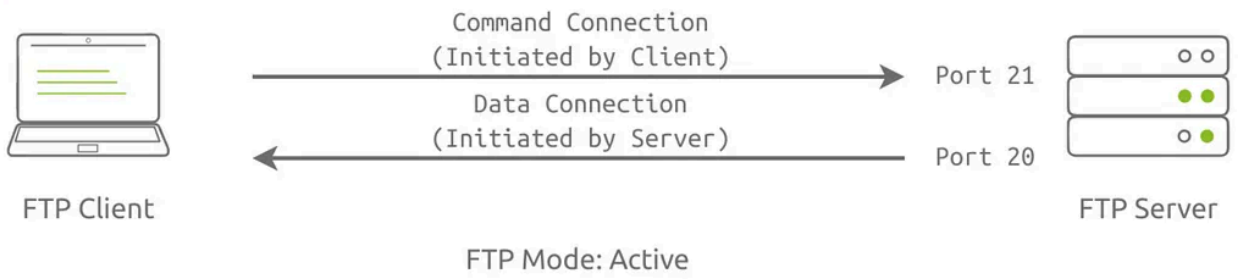
FTP has two modes:

1. Active modes
2. Passive mode

Active Mode

In active connection mode, the client establishes the control connection to send commands/authentication parameters to the server. After authentication and upon the client's request to initiate data transfer, the server establishes the data connection to the client to transfer the data. In brief:

1. The FTP client connects to the FTP server at TCP port 21 to establish a command connection.
2. The FTP server connects to the FTP client at TCP port 20 to establish a data connection.

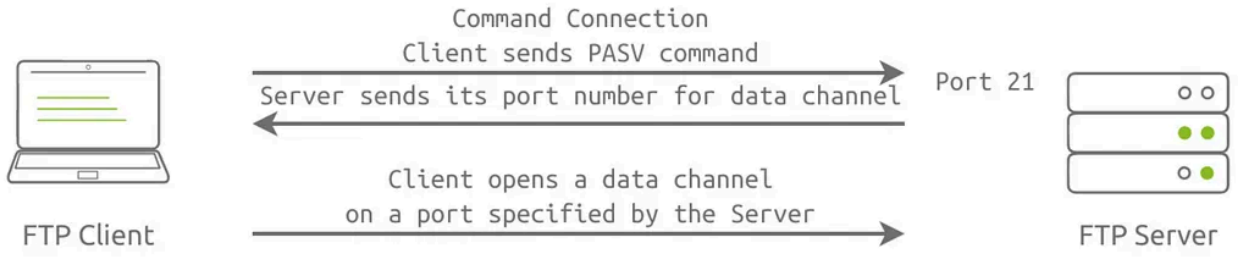


FTP Mode: Active

It is worth mentioning that this type of connection is unsuitable in an environment where the client is behind a firewall, as it will block incoming connections to the client. In the case of a client behind a firewall, a passive connection would be necessary.

Passive Mode

In passive connection mode, the client establishes the control and data connections. The client sends the `PASV` command to the server over the command channel; the server sends a random port to the client. As soon as the client receives the port number, the client establishes a connection to the provided port number so that the server can initiate the data transfer to the client.



This type of connection works well when the client is behind the firewall.

FTP Data Types

When data exchange between client and server takes place, the following type of data types are used:

- **ASCII/Type A**: This is the default type and is used for text file transfers. If necessary, data is converted into 8-bit ASCII before transmission and then converted back upon reception.
- **Image/Type I**: This is commonly referred to as the binary mode. It uses byte-by-byte transmission. The recipient stores the received bytes upon reception.
- **EBCDIC/Type E**: It is suitable for text communication using the EBCDIC character set.
- **Local Type L n:** It is typically used for file transfer among machines that do not support 8-bit bytes transfer. Here `n` is a second parameter that represents byte size.

Request and Response - FTPS

As the name implies, FTPS is an extension of FTP. It adds an encryption layer to transmit command and data channels between client and server securely. The following two methods are used to invoke security:

- **Implicit Connection**: In this connection, FTPS client and server establish a link in which both command and data channels are secured automatically with SSL encryption.
- **Explicit Connection**: The FTP client explicitly requests the server to invoke an SSL/TLS secured session on port 21 and then continue data transfer based on a mutually agreed authentication mechanism. With explicit connection, you can choose which channel to encrypt by choosing among three modes of communication for control and data channel, i.e., control only encrypted, data only encrypted and both control and data encrypted.

The standard port for FTP and Explicit FTPS is 21, whereas it is 990 in the case of Implicit FTPS. Adding FTPS protects against sniffing attacks against login information and data.

SMTPS Protocol

Technical Overview

Simple Mail Transfer Protocol Secure (SMTPS) is an extension of SMTP, which is used for email communication. We should not confuse SMTP with POP3. Although both are used for email communication, SMTP is an "Email Push Protocol" used to transfer email messages from the client to the server. In contrast, POP3 is used to download email messages from the server to the client. SMTPS is an extension of SMTP; it uses TLS/SSL to provide authentication, integrity, and confidentiality for transferred data. First, let's review the SMTP protocol.

SMTP Protocol

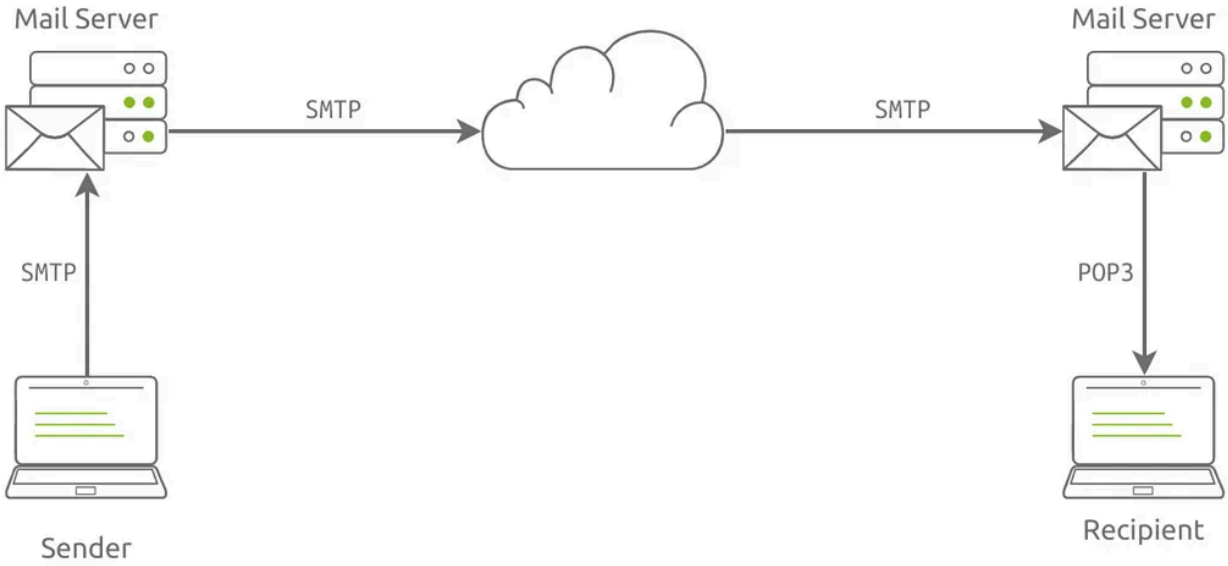
As described earlier, SMTP is an "Email Push Protocol" commonly used to transfer emails from an SMTP client to an SMTP server. SMTP is implemented in the following two models:

- **SMTP End-to-End:** This model is used for email communication between organizations. In this model, the sender-side SMTP client initiates an SMTP connection to the recipient's SMTP server.
- **SMTP Store-and-Forward:** This model is used for email communication within an organization. In this model, the SMTP server will maintain the copy of the mail within itself (i.e., store) until the copy is forwarded to the receiver.

SMTP Components

To understand the workflow of SMTP, we will study the following essential components of SMTP:

- **User Agent (UA):** UA is responsible for creating the email message and sending it to the Mail Transfer Agent (MTA).
- **Mail Transfer Agent (MTA):** MTA will transfer the email from the UA to the recipient MTA across the Internet (often, the MTA and Mail Delivery Agent are hosted on the same server).



TLS Process in SMTPS

SMTPS is not a proprietary protocol; instead, it wraps SMTP inside TLS. You can say that SMTPS is similar to SMTP on the application layer, with an extension of TLS encryption at the transport layer. For encryption, the `STARTTLS` command is used between the email client and the email server.

Port 587 and 465 are both frequently used for SMTPS traffic. Mails transmitted using SMTP are not encrypted, so they are prone to sniffing attacks. Therefore, SMTPS is used to encrypt emails through TLS before transmission. In addition, SMTPS also forbid attackers from sending spam messages from compromised/vulnerable domains, exfiltration sensitive information, and conducting phishing attacks.

POP3S Protocol

Technical Overview - POP3S

Post Office Protocol Secure (POP3S) is an extension of the POP3 protocol; it is used for the encrypted retrieval of email messages from the email server to the email client. So first, let's review the POP3 protocol.

POP3 Protocol

In the previous section, we explored how SMTPS is used for secure email transmission. SMTP is not responsible for retrieving email messages; here, POP3 comes into play. POP3 is the latest POP version; it retrieves email messages from a Mail Delivery Agent (MDA) to a Mail User Agent (MUA).

POP3 Components and Workflow

Like SMTP, POP3 has two components: client (MUA) and server (MDA). The steps are the following:

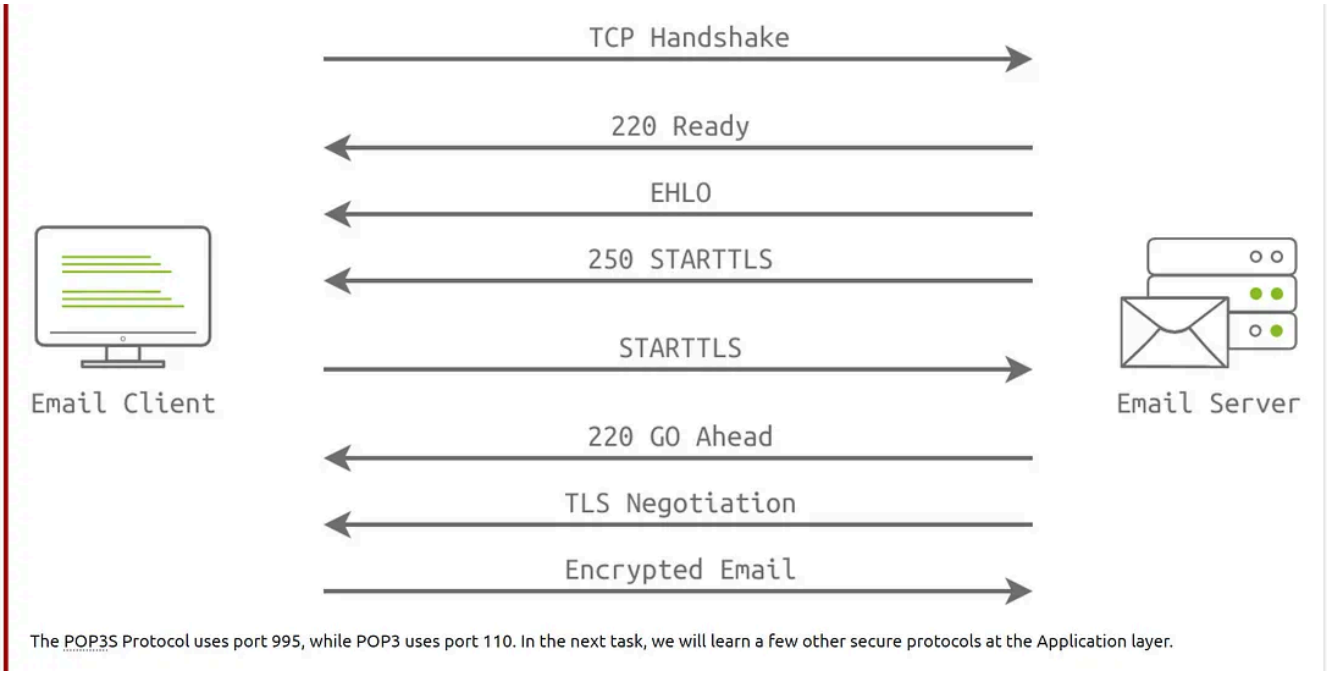
1. The email client establishes a connection to the email server.
2. The email client downloads all the queued emails from the email server. (This is a default option; however, the client can select only particular email messages to download.)
3. All emails are saved on the device that initiated the connection.
4. The email server deletes the email copy. (This is a default option; a client can choose not to download an email after it is retrieved.)

Limitations of POP3

- **Emails are Processed Locally:** No synchronization of email messages across multiple devices. Protocol downloads the emails on the currently logged-in device and usually deletes them from the server.
- **Transmission in clear text:** The username and password, along with the email messages, are sent in cleartext, which makes them vulnerable to sniffing attacks.

POP3S

As we conclude, POP3 is considered weak from a security point of view. This requires an added layer of security; hence, POP3S comes into play. POP3S is an extension of POP3, which wraps the communications related to email messages within TLS. For this purpose, the client and server initiate the `STARTTLS` command, as shown in the figure below. After the `EHLO`, the POP3S server will trigger the switch to TLS. Note that `EHLO` stands for Extended HELO, where `HELO` is the command used to identify to the server.



What is the default port for HTTPS?

Answer: **443**

In a passive FTP connection, what does the client send the first command over the command channel?

Answer: **PASV**

SSL Checker

Use our fast SSL Checker will help you troubleshoot common SSL Certificate installation problems on your server...

www.sslshopper.com

Network Tools: DNS,IP,Email

DNS and Network troubleshooting and diagnostic tools integrated into one sweet interface.

mxtoolbox.com

Task 3: **Application Layer — More Secure Protocols**

DNSSEC

As you would already know, DNS stands for Domain Name System. The DNS protocol is responsible mainly for resolving domain names. Instead of remembering the IP address, you need to focus on the domain name. For instance, at the time of this writing, `example.com` resolves to `93.184.216.34`; it is clear which one is easier for the human mind to remember.

DNS works by sending a DNS query. For instance, when browsing the web, your web browser might send a query for DNS record type `A` or `AAAA`, i.e., IPv4 or IPv6 addresses. In the following console output, we can see the host with IP address `192.168.0.102` sending two DNS queries to the DNS server `1.0.0.1` regarding the domain name `example.com`. We can see the responses for the `A` and `AAAA` queries.

Terminal

```
user@TryHackMe$ sudo tshark port 53
    1 0.000000000 192.168.0.102 → 1.0.0.1      DNS 82 Standard query 0x2717 A example.com OPT
    2 0.012241216      1.0.0.1 → 192.168.0.102 DNS 98 Standard query response 0x2717 A example.com A 93.184.216.34 OPT
    3 0.013454645 192.168.0.102 → 1.0.0.1      DNS 82 Standard query 0xac05 AAAA example.com OPT
    4 0.018705620      1.0.0.1 → 192.168.0.102 DNS 110 Standard query response 0xac05 AAAA example.com AAAA
2606:2800:220:1:248:1893:25c8:1946 OPT
```

This name-to-IP address resolution is very convenient; however, anyone on the network could have responded with a forged response. Furthermore, the host that sent the query would have accepted "any" response. In other words, the host would connect to a rogue server. One way to avoid such a situation would be by using DNSSEC.

DNSSEC makes it possible to ensure that the DNS response we receive is from the domain owner. To achieve this, DNSSEC requires two main things:

1. The DNS zone owner should sign all DNS records using their private key.
2. The DNS zone publishes its public key so users can check the validity of the DNS records signatures.

In other words, the data to our DNS query is signed to ensure its integrity and authenticity; moreover, we can efficiently check the signature.

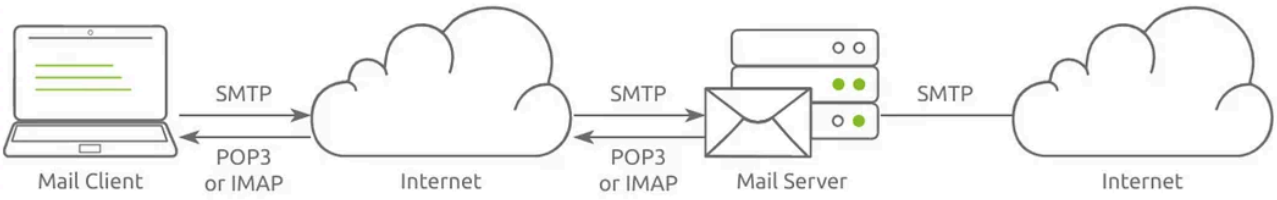
With signed records, DNSSEC provides the following:

- Authenticity: You can confirm that a certain DNS owner has authored and sent the record. Authenticity is possible because the received record is signed by the DNS owner's private key.
- Integrity: You can ensure that no changes have been made to the record on its way. Any changes to the record will render its signature invalid.

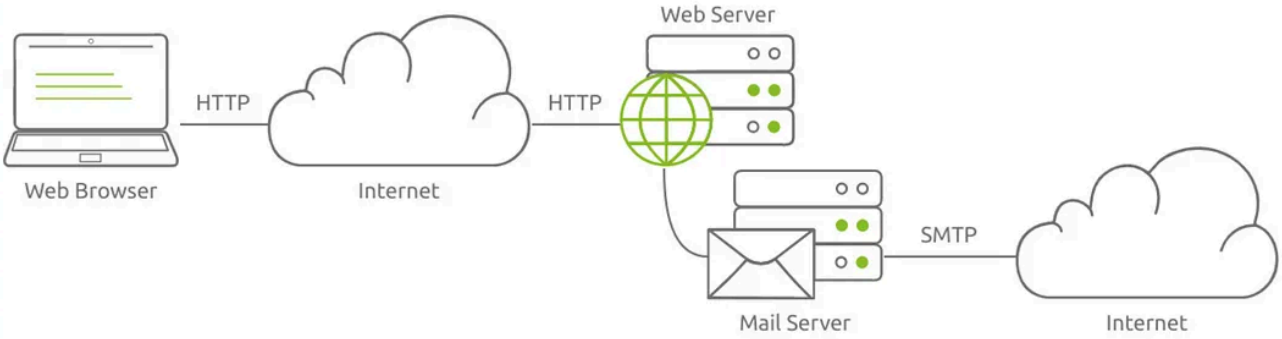
OpenPGP

When the first email was sent in 1971, we had a different cyber security landscape. Email protocols such as SMTP and POP3 are designed to send emails in cleartext. The same applies to IMAP, which allows synchronizing your mailbox with that on the server. All these protocols make your email no different than an exposed postcard open for everyone to see as it is handed from one server to another.

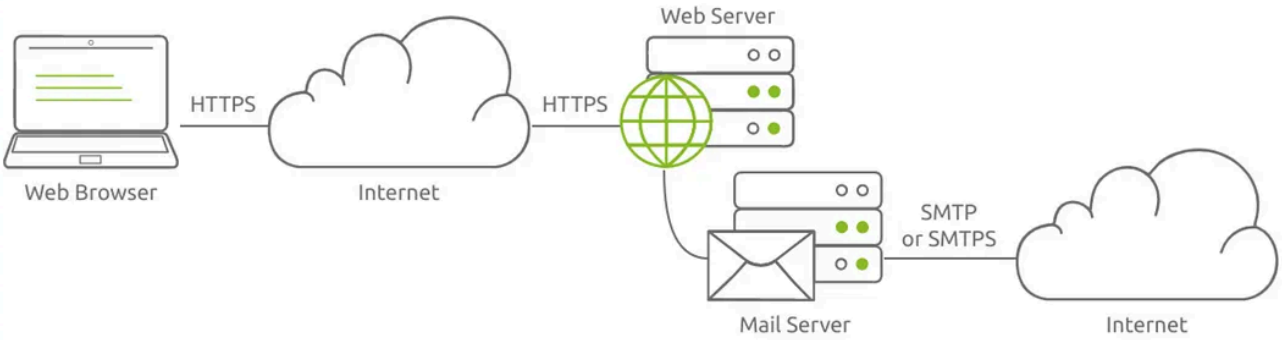
The image below shows a simplified example where a user uses an email client to send their email over SMTP and receive new email messages over POP3 or IMAP. The mail server uses SMTP to deliver the user's email messages to the intended recipient. Since all these protocols use clear text, an intruder can read the email messages as they travel across the Internet.



With the increased popularity of web-based email, users started to connect to a web server to read and compose their email messages. The image below shows an email message as it is written using a web browser. The web server, in turn, uses a mail server to send composed email messages and receive incoming ones. The connection was over HTTP, which meant that the same security issues related to confidentiality and integrity persisted.

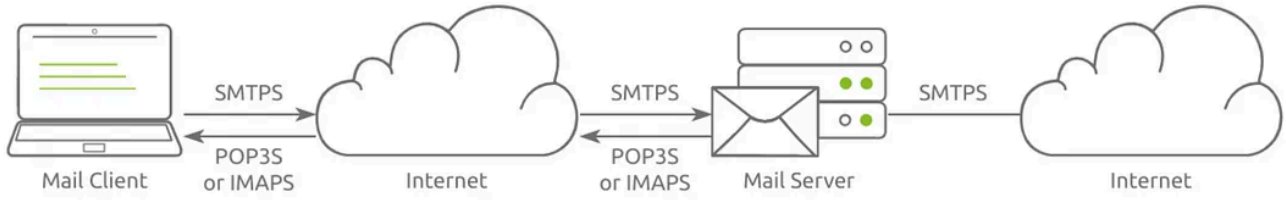


However, as service providers realized the need for SSL/TLS to secure web traffic, HTTPS became the new standard. Consequently, most web-based email systems migrated to HTTPS, causing the traffic between the web browser and the web server to be encrypted. However, the email traffic is not necessarily encrypted between the web server and the mail server(s). The web server and mail servers can read the contents of the messages; moreover, mail servers might use SMTP to transfer the messages, which means that email messages will traverse the Internet in cleartext.



Eventually, SSL/TLS started to find their way into all email protocols. SMTP, POP3, and IMAP became SMTPS, POP3S, and IMAPS, respectively. The "S" added to the protocol name refers to **secure**, indicating the addition of SSL/TLS on top of the existing protocol.

The image below shows a simplified example where a mail client uses SMTPS to send an email and uses POP3S or IMAPS to receive an email. The result is that email is sent encrypted between the client and the server; however, the mail server can read the email message contents.



The addition of SSL/TLS has dramatically enhanced the security of email messages. However, we must still trust the mail servers across the way. If this is not something you are comfortable with, you need to consider a standard such as OpenPGP. PGP (Pretty Good Privacy) is an encryption program created by Phil Zimmerman. OpenPGP is an **open standard** for signing and encrypting files and email messages and is detailed in RFC 4880. GnuPG (Gnu Privacy Guard), or simply GPG, is a free and open-source implementation of the OpenPGP standard. In brief, GnuPG allows you to sign and encrypt your data and communications.

GnuPG can easily integrate with your mail client to seamlessly sign, encrypt and decrypt email messages. Email messages encrypted using GnuPG (i.e., following OpenPGP standard) be only readable by the intended recipient. In other words, no one, including the mail servers, can read the contents of the messages except the intended recipient.

When used with email, GnuPG requires each user to generate a key pair: a private key and a public key. In simple terms, the sender's private key is used for signing, while the recipient's public key is used for encryption. From the recipient's perspective, the sender's public key is used to check the signature, while the recipient's private key is used for decryption. For more information about asymmetric encryption, we recommend you check the Introduction to Cryptography room.

OpenPGP implementations, such as GnuPG, offer a great solution to protect the confidentiality and integrity of email message contents. However, this does not include email message headers.

Below is an example of a message before and after being encrypted using OpenPGP. The original message is shown below.

```
Terminal

user@TryHackMe$ cat message.txt
Hello,

Please proceed with the transaction.

Best,
Strategos
```

To use OpenPGP, both parties need to generate a key pair using the command `gpg --gen-key`. This command will ask the user to provide their name and email address and create a private and public key. The private key should be stored securely, while the public key should be shared with the other parties we wish to communicate securely with.

Using an email client that supports OpenPGP will encrypt the message using the key of the recipient; however, if we want to accomplish this via the command line, the command would be something like the following:

```
gpg --encrypt --sign --armor -r strategos@tryhackme.thm message.txt
```

Notice the following options:

- `--encrypt -r recipient@tryhackme.thm` will encrypt `message.txt` using the public key associated with the recipient's email. This will provide confidentiality.
- `--sign` will sign our message (using our private key). This will prove authenticity.
- `--armor` is to produce the output using ASCII instead of binary.

Encrypting using `gpg` created the following message that can be sent seamlessly with an email client.

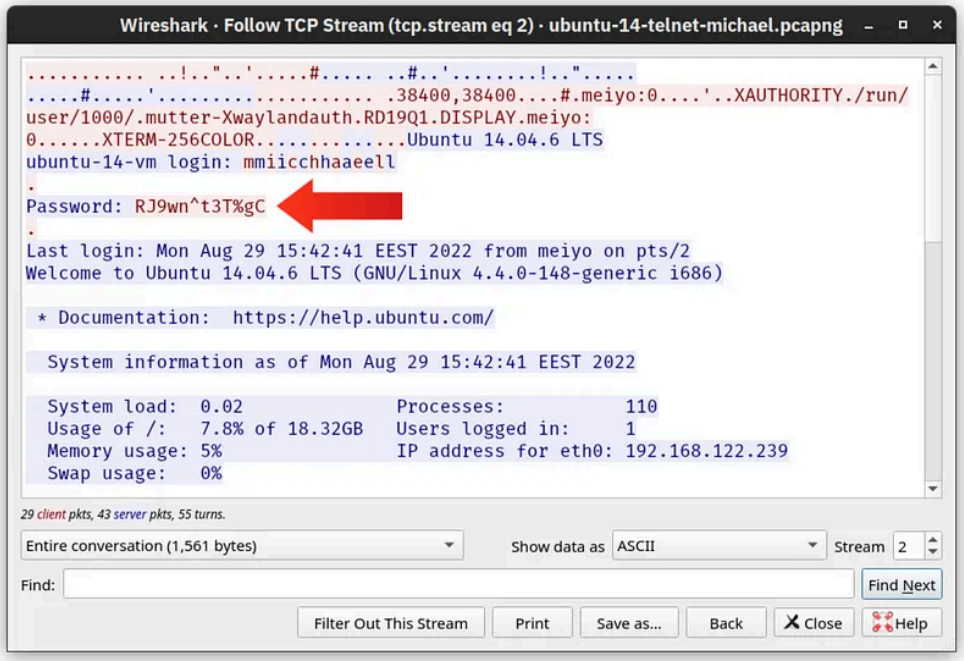
```
Terminal

user@TryHackMe$ cat message.txt.asc
-----BEGIN PGP MESSAGE-----

hF4Dt1Jduipo/LESAQdAmqgCLQRRQCFNOBWSF+dY64suA8xtty7ysfolfF7+fnUw
crwR2ioRTcXTe6c0dZl/sdmtjDJPZWGHI3XcD7XWA2hPDb+w4P46e9FJGsCE/JaO
1I4BCQIQZc91A79Ebli/41D0aVkBmDpjIgvpwjHdmomT7dghTcB+Qp80WbYDnV20
4qTgdgdAnLtQp3fnJCXlZ0BfecPB+ZfECdD1IAleBB3o14v5v/ntfPKfXPZwODUm
ELY7piC2GclBWbirrZsnzTLWeYCrABiKJ3Rb75VgJXdM1uKNBY0HLN06VdEuDy+L
=nRQE
-----END PGP MESSAGE-----
```

SSH

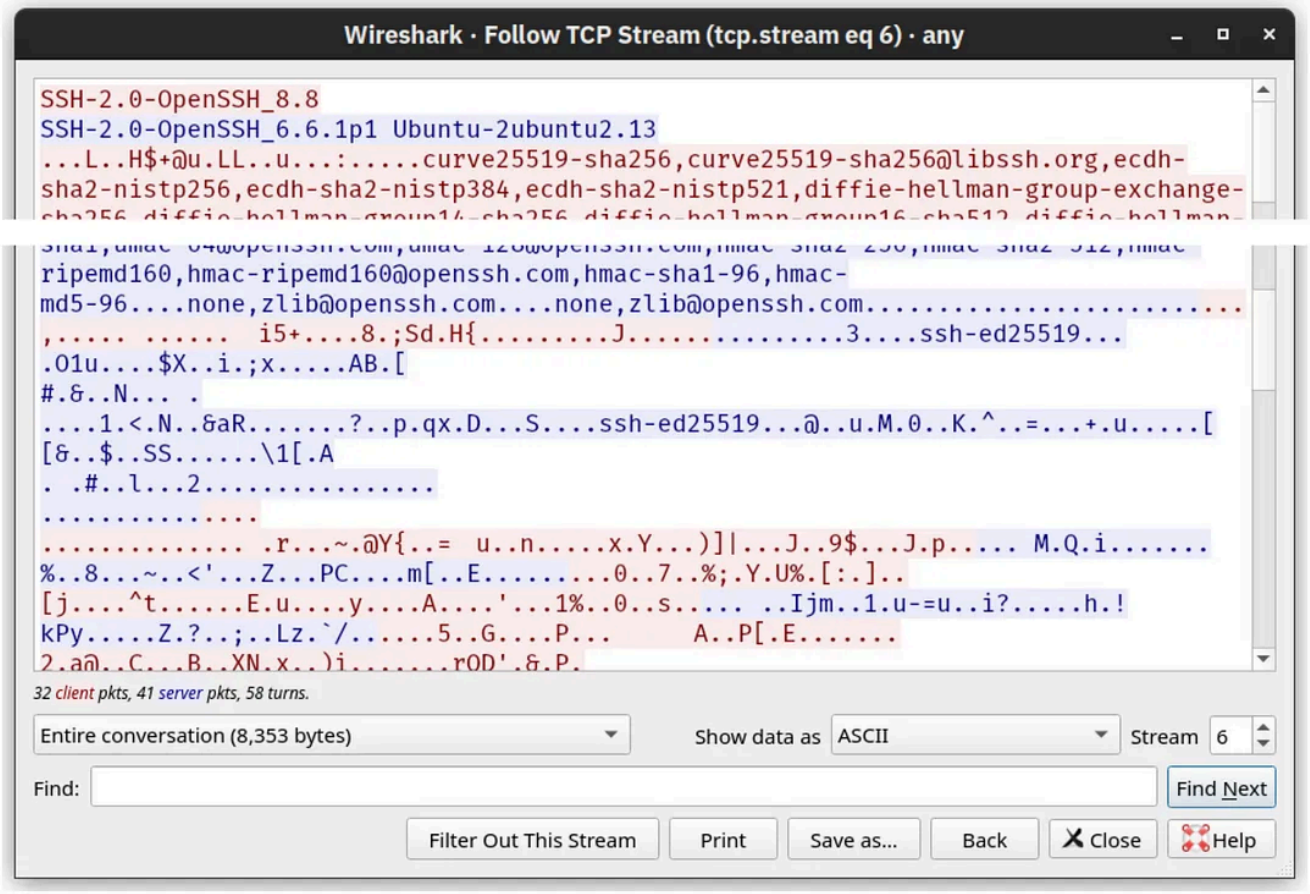
With the beginning of networked systems, there was a need to connect to a system over a network. For instance, a user needs to execute processes and administer the system. The aim was to do this remotely instead of being physically present at the computer.

Two of the earliest protocols were Telnet and remote login, with the clients `telnet` and `rlogin`, respectively. Both protocols made it possible to log in to a system over a network; however, neither focused on the security aspects. The confidentiality of the exchanged traffic, especially the login credentials, was not protected. Moreover, the integrity of the traffic and commands sent was not ensured. In other words, it was easy for an attacker monitoring the network to read the login credentials and modify the commands sent over the network.

The screenshot below is taken from Wireshark after using "Follow TCP Stream" against a Telnet session. We can see the user typing his username `michael` and pasting his password `RJ9wn^t3T%gC`. Although the password is secure by modern standards, it is sent in cleartext for any properly located network packet-capturing software to read. Note that in the image below, the text in red is sent by the Telnet client, while the text in blue is sent by the Telnet server.



The Secure Shell Protocol (SSH) provided the security requirements lacking in Telnet and remote login. With SSH, it is no longer feasible for the attacker to read the login credentials or modify the traffic. If we attempt to "Follow TCP Stream" on Wireshark after capturing the packets, we won't get any information regarding the username, password, or issued commands. The screenshot below shows that the only visible information is the version numbers and the supported protocols. (Please note that the red characters are sent by the client, while the blue characters are sent by the server.)



What does PGP stand for?

Answer: **Pretty Good Privacy**

What does GPG stand for?

Answer: **GNu Privacy Guard**

What command would you use to generate a key pair using `gpg` ?

Answer: **gpg --gen-key**

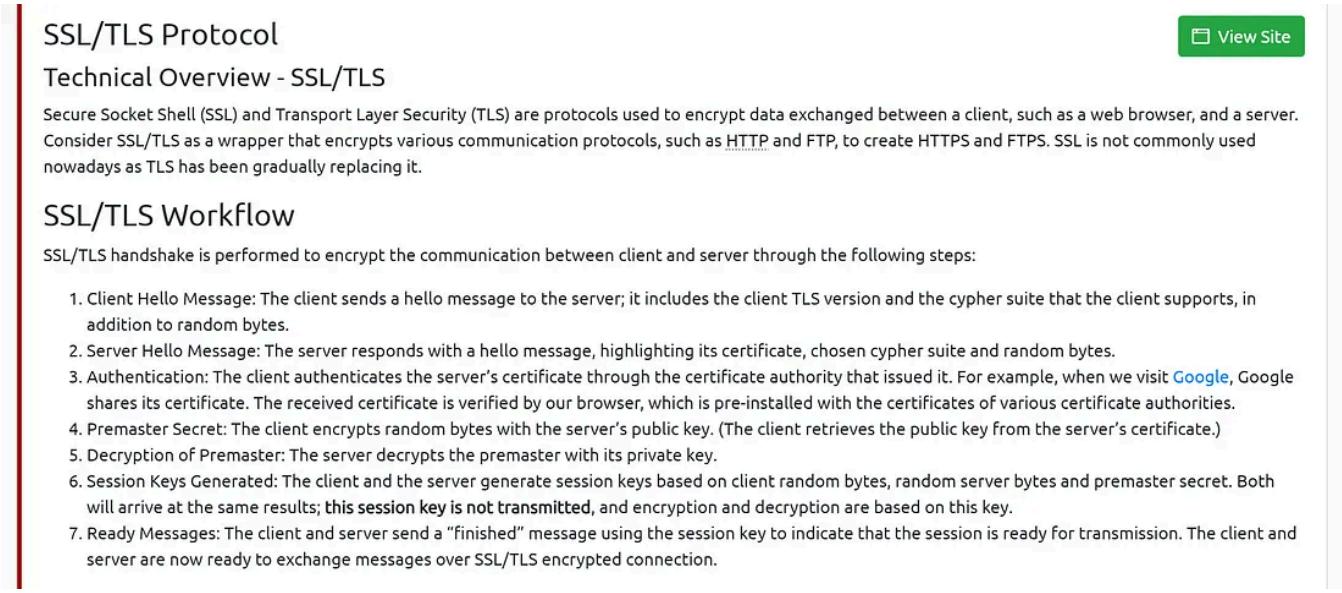
Consider the following three clients:

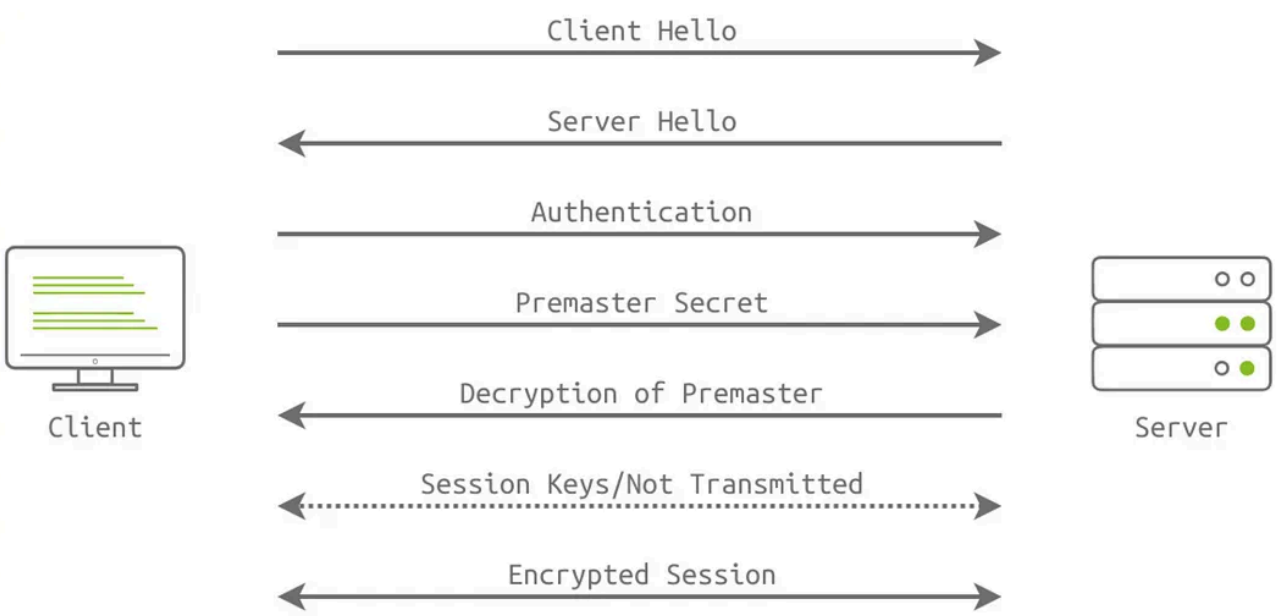
1. `rlogin`
2. `telnet`
3. `ssh`

Provide the number of the client that encrypts the traffic.

Answer: **3**

Task 4: **Presentation and Session Layers**





TLS is a wrapper that encrypts communication of communication protocols. It has port numbers for various protocols, such as 443 for HTTPS and 990 for FTPS.

SOCKS5 Protocol

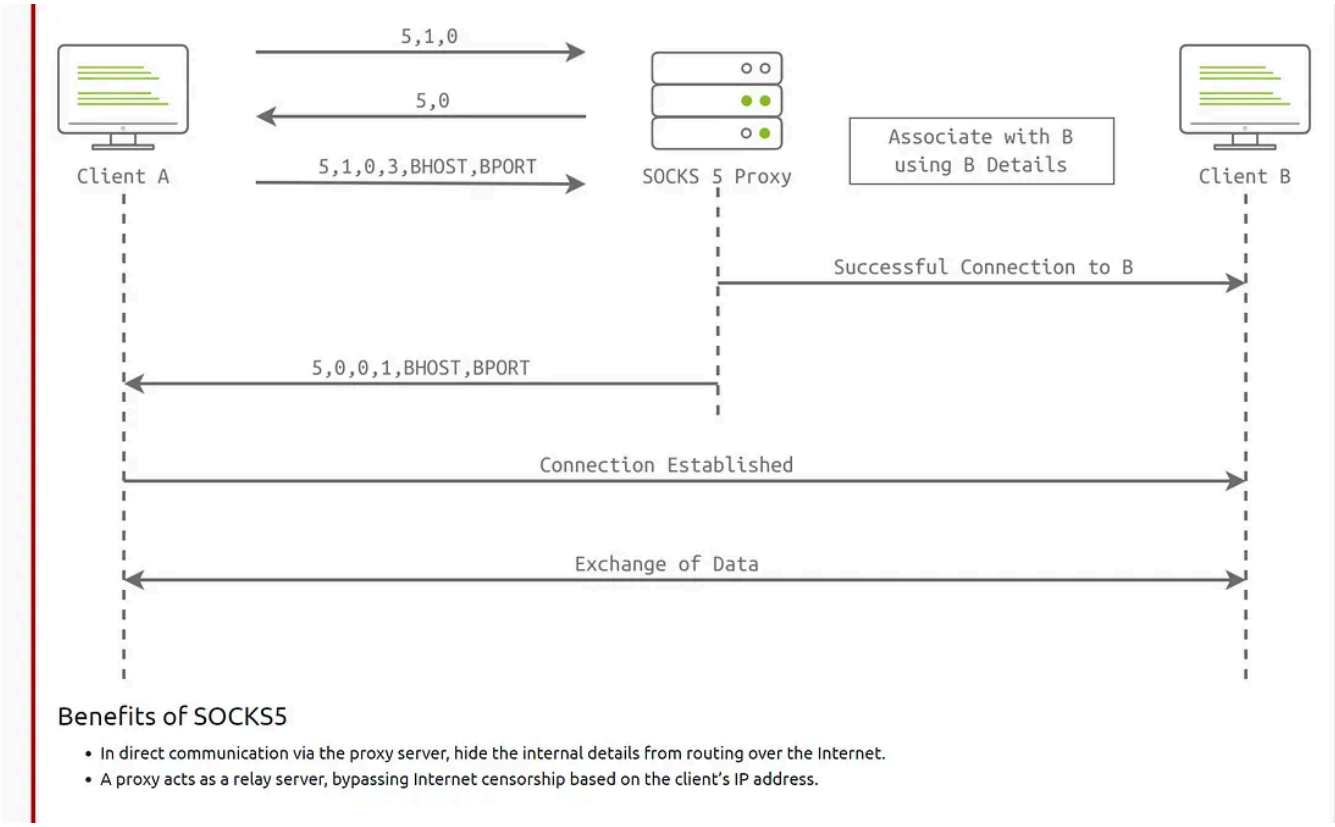
Technical Overview - SOCKS5

Socket Secure (SOCKS) is a proxy protocol for data exchange through a delegate server (SOCKS5 proxy). It is used to secure application layer protocols. For example, the Squid server implements the SOCKS5 protocol to transfer data via the HTTP protocol.

SOCKS5 Workflow

Consider a scenario when user A wants to connect with client B over the Internet, but a firewall is between them. The following handshake steps are involved:

- **Client Initiation**
  - Client A connects with the SOCKS5 proxy and sends the first byte (0x05) to the proxy where "5" is the SOCKS version.
  - Client A sends a second byte (0x01). One means authentication is supported.
  - Client A sends the third byte (0x00, 0x01, 0x02, or 0x03); these bytes denote the supported authentication methods and can be of variable length.
- **SOCKS5 Proxy Reply**
  - The proxy sends back a second byte, which is the chosen authentication method by the proxy server.
  - After the initiation packet, client A sends the request packet, which includes BHOST & BPORT numbers.
  - The successful session is established between client A and the proxy. The same steps are involved in the association of client B with the proxy.
- **Data Transfer**
  - After successfully associating both clients with a proxy server, both clients can exchange data and share information that will be routed through the proxy server.



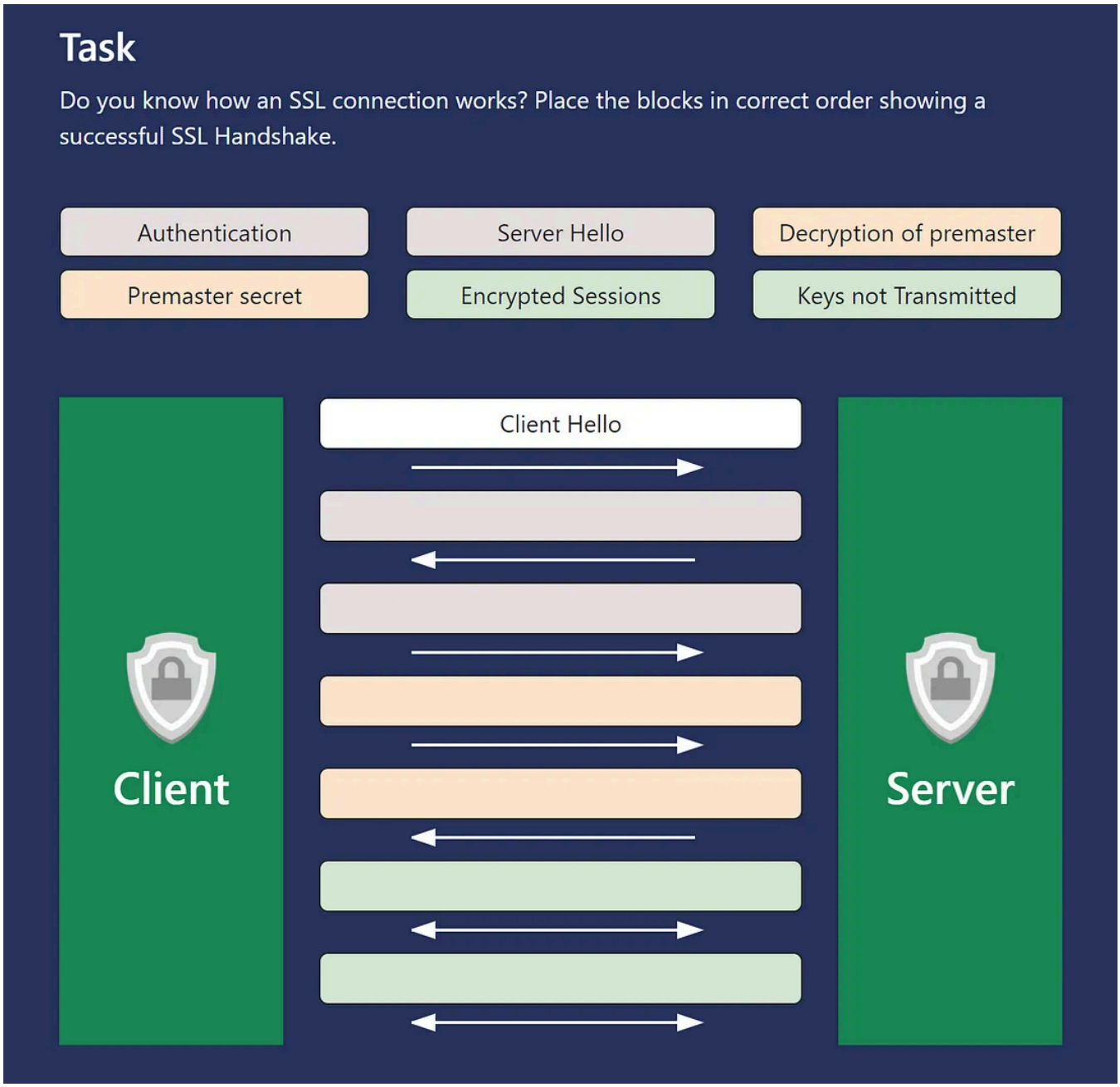
Does the hello message during the SSL handshake include the TLS version (yea/nay)?

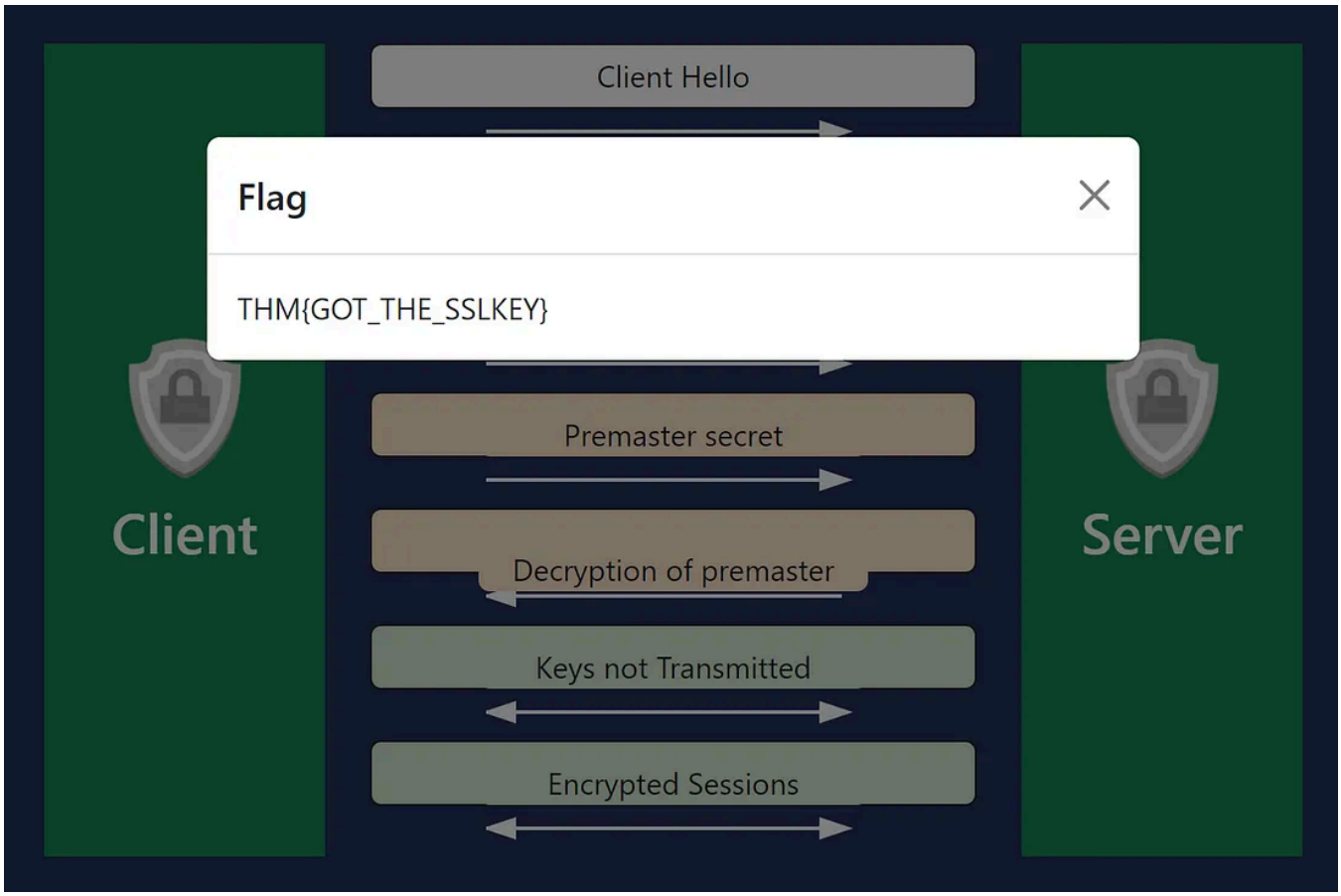
Answer: **Yea**

During the client initiation process of SOCKS5, what is the SOCKS version if the client sends the first 5 bytes (0x05)?

Answer: **5**

Click the **View Site** button at the top of the task to launch the static site in split view. What is the flag after completing the exercise?


Task
Do you know how an SSL connection works? Place the blocks in correct order showing a successful SSL Handshake.
Authentication
Server Hello
Decryption of premaster
Premaster secret
Encrypted Sessions
Keys not Transmitted
Client Hello
Client
Server



Answer: **THM{GOT_THE_SSLKEY}**

Task 5: **Network Layer**

IPsec

IPsec stands for Internet Protocol Security. In this room, we use IPsec to refer to IPsec-v3. IPsec provides security by adding authentication and protecting the integrity and confidentiality of the network traffic. IPsec uses the following protocols:

1. Authentication Header (AH): Provides authentication and integrity.
2. Encapsulating Security Payload (ESP): Provides authentication, integrity, and confidentiality.
3. Security Association (SA): Is responsible for negotiating the encryption keys and algorithms. One example is Internet Key Exchange (IKE). Discussing SA in more detail is outside the scope of this room.

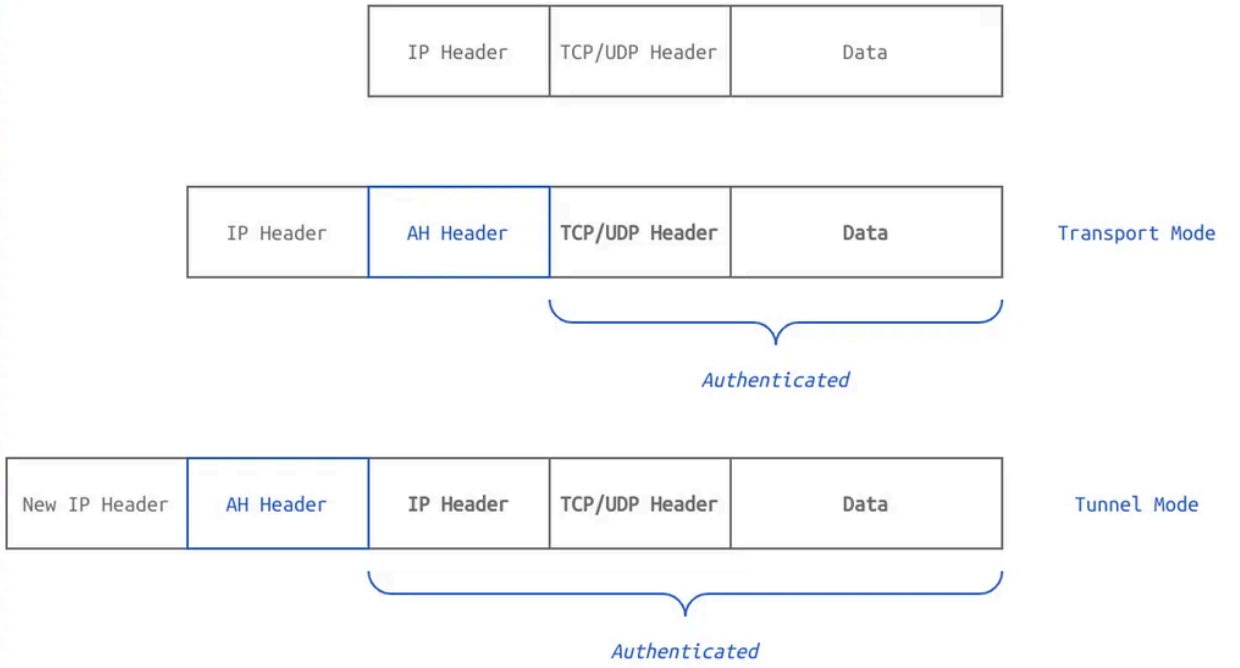
In the following sections, we discuss AH and ESP in more detail.

Authentication Header (AH)

Authentication Header (AH): The AH protocol is responsible for the authentication and the integrity of the traffic; however, it cannot protect the confidentiality of the data.

The AH protocol works in two modes, as shown in the figure below:

1. Transport Mode: Provides authentication for the TCP/UDP header and data.
2. Tunnel Mode: Provides authentication for the IP header, TCP/UDP header, and data.

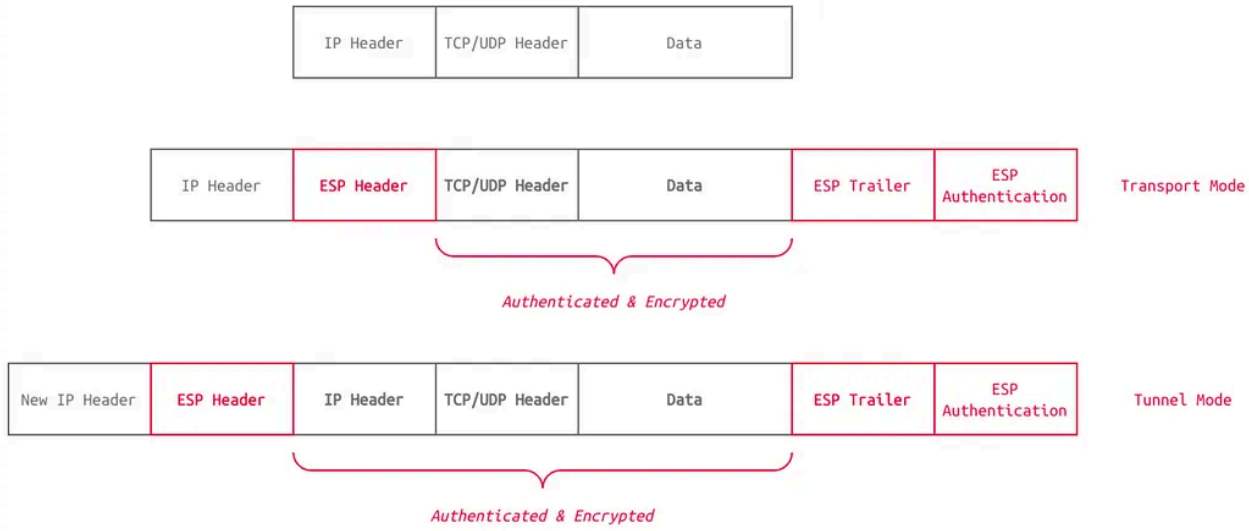


The AH protocol is suitable if providing authentication and integrity is enough without confidentiality. It is worth mentioning that AH is optional in IPsec-v3; however, it is mandatory to implement in IPsec-v2.

Encapsulating Security Payload (ESP)

Encapsulating Security Payload (ESP) provides encryption in addition to authentication and integrity. It works in two modes:

1. Transport Mode: Provides security (confidentiality and integrity) for the TCP/UDP header and data.
2. Tunnel Mode: Provides security (confidentiality and integrity) for the IP header, TCP/UDP header, and data.

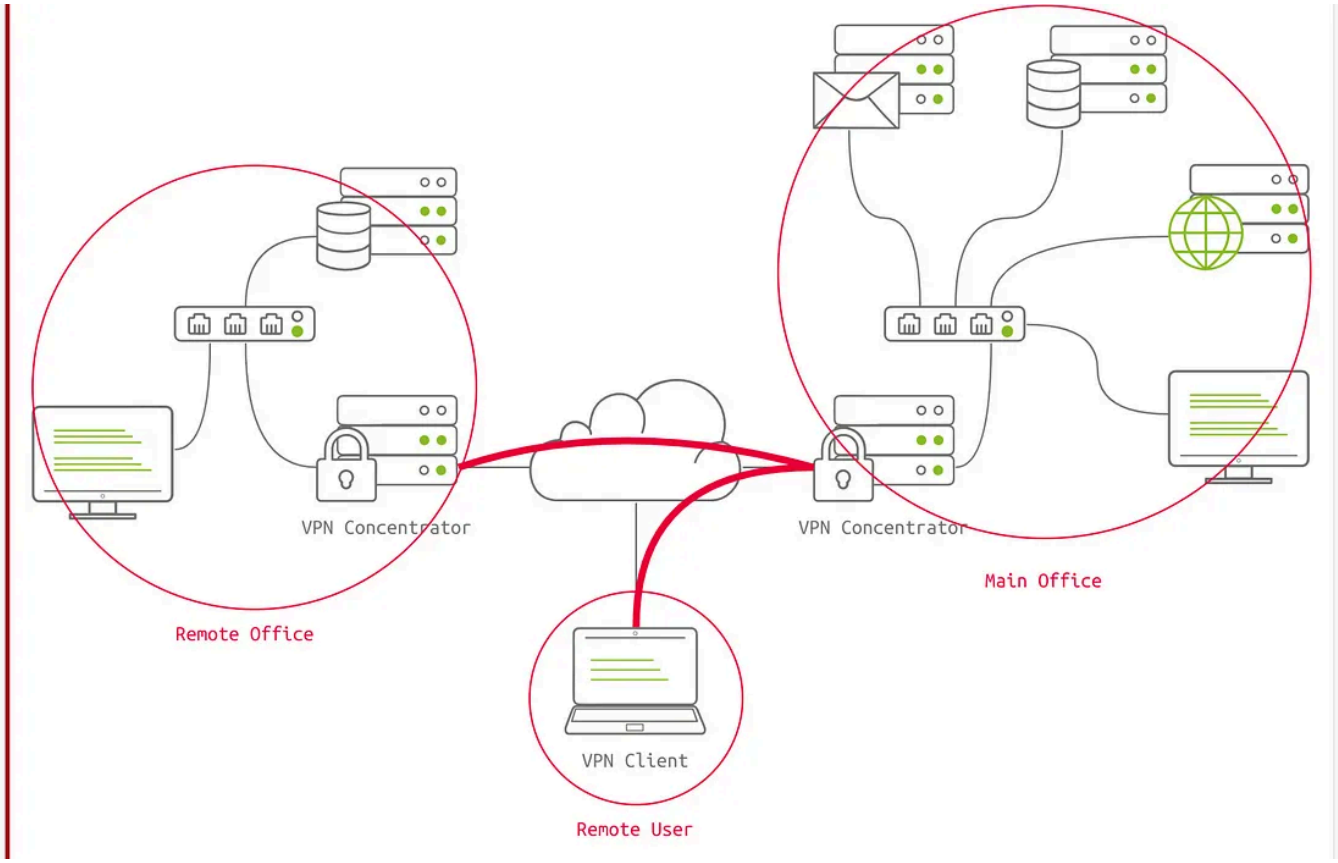


VPN

When the TCP/IP protocol was designed, security requirements such as confidentiality and integrity were not a design target. In contrast, availability was the priority as one of the purposes of the Internet is to withstand a nuclear attack, as is evident by the routing protocols adapting quickly when a link goes down. But we need to allow a corporation to use the existing Internet infrastructure to connect its offices securely. The answer lies in setting up a VPN.

A Virtual Private Network (VPN) makes it possible to establish a private connection over a public network. In other words, we can establish a secure connection over an insecure infrastructure.

For instance, in the figure below, we can see a remote office and a remote user connected over a VPN to the main office. A VPN connection requires a VPN client and a VPN server or concentrator. All the traffic between the VPN client and server is encrypted.



The two most common protocols used to establish VPN connections are:

1. IPsec
2. SSL/TLS

IPsec's ESP is a perfect protocol for setting up secure tunnels between different networks or a computer and a network. ESP can provide security and integrity of all data transmitted between two points; moreover, even the IP address can be hidden in tunnel mode. Note that the system must be behind a VPN concentrator for the IP address to be hidden. Cisco VPN systems offer IPsec.

Although SSL was created to secure HTTP traffic, SSL/TLS has found its way to establish secure VPN connections with OpenVPN. Using various tools and libraries built around TLS, OpenVPN offers different authentication and encryption mechanisms to establish VPN connections.

Some older protocols that can be used to establish VPN connections are no longer considered secure. One example is Point to Point Tunneling Protocol (PPTP), which is no longer considered secure.

What does ESP stand for?

Answer: **Encapsulating Security Payload**

Which protocol does the Cisco VPN client use to establish a VPN connection?

Answer: **IPSec**

Which protocol does the OpenVPN project use for encryption and authentication?

Answer: **SSL/TLS**

Task 6: **Conclusion**

> In this room, we have covered various network security protocols essential for the security of transferred data. The most widely used mechanism for securing data over an insecure channel is to add an **SSL/TLS wrapper**, as we saw in HTTPS, FTPS, POP3S, and SMTPS. The security protocols protect against Man In the Middle (MITM), replay, and eavesdropping attacks.
>
> If you want to study more and gain a deeper understanding of network protocols and their related (in)securities, we recommend you check the Wireshark room.

Tryhackme Walkthrough Tryhackme Writeup

Follow

Written by Daniel Schwarzentraub

116 Followers · 4 Following

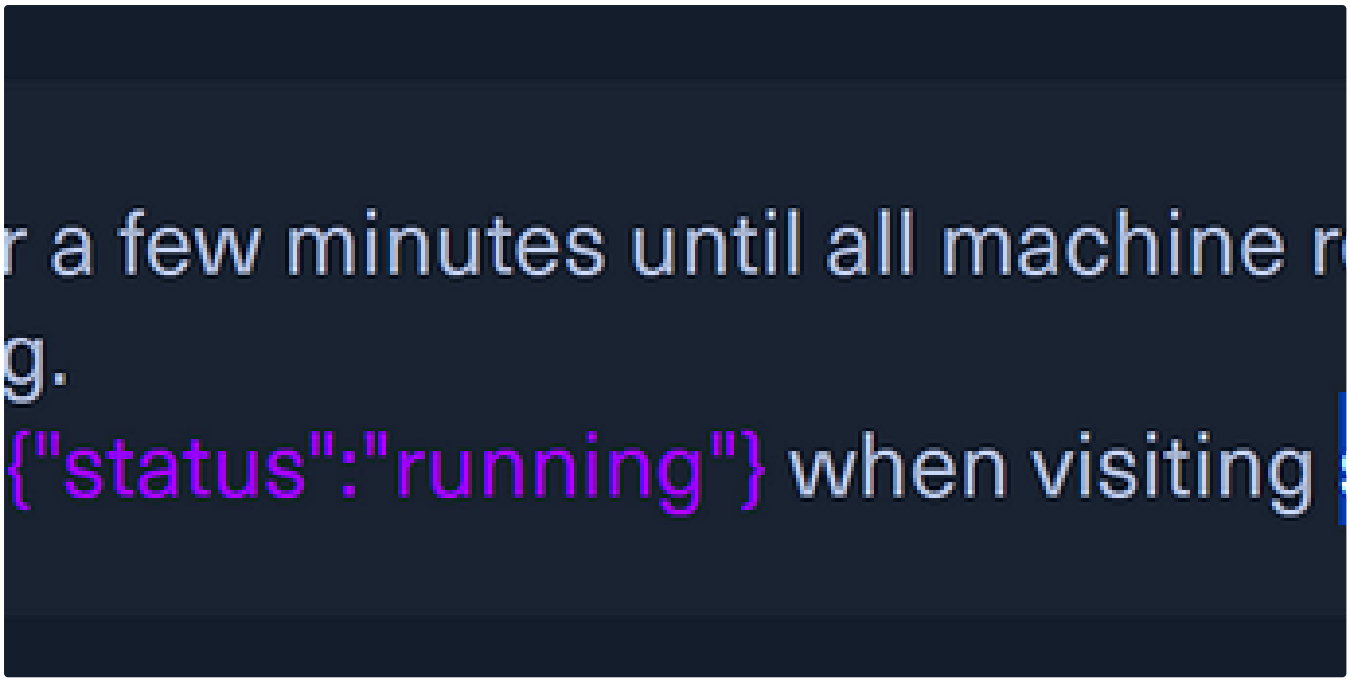
PNW_Hacker

No responses yet

What are your thoughts?

Respond

More from Daniel Schwarzentraub

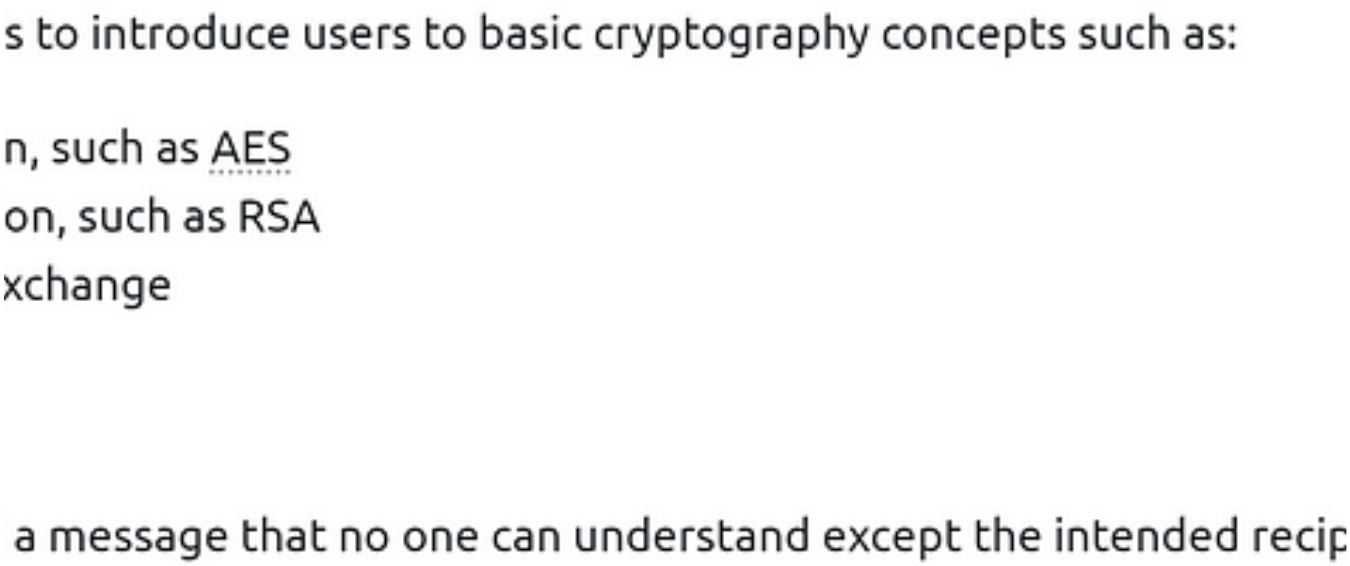


Daniel Schwarzentraub

HTB—Tier 1 Starting Point: Three

HTB—Tier 1 Starting Point: Three

Jul 20, 2023 4 2

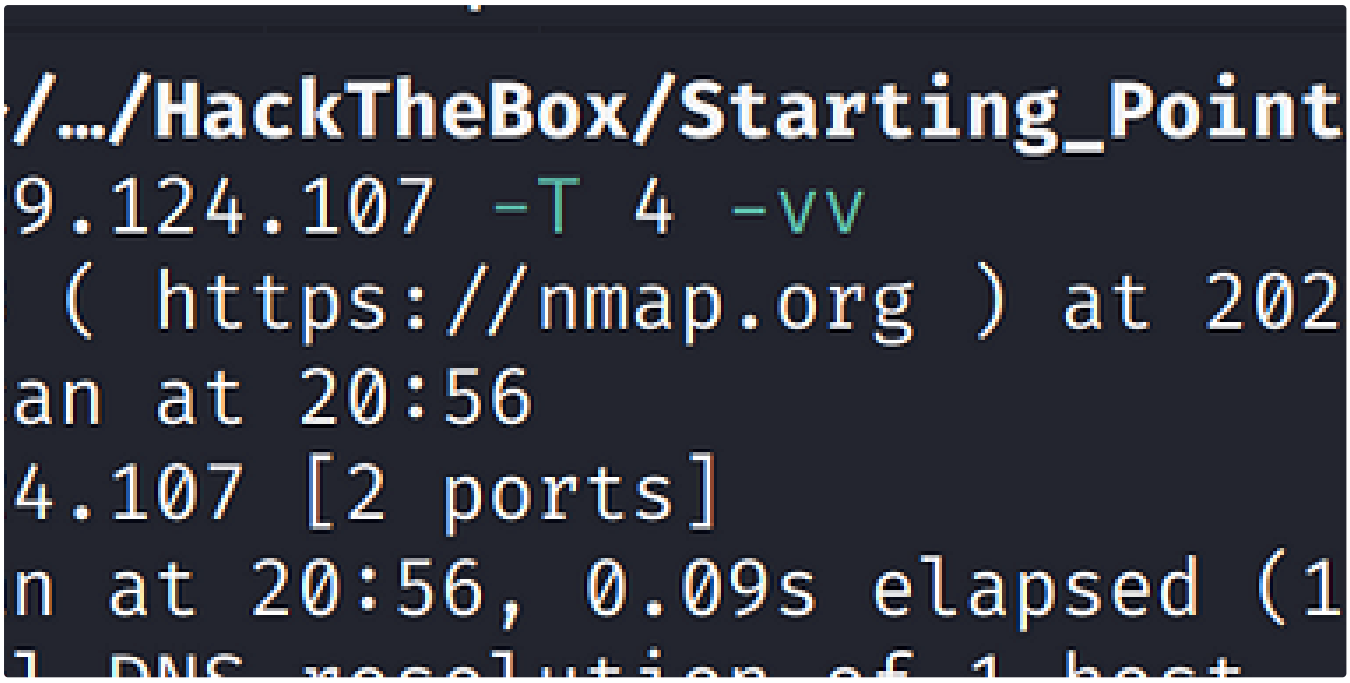


Daniel Schwarzentraub

Tryhackme: Introduction to Cryptography

Tryhackme: Introduction to Cryptography

Sep 26, 2023 2



Daniel Schwarzentraub

HTB—Tier 2 Starting Point: Archetype

HTB—Tier 2 Starting Point: Archetype

Jul 21, 2023

erative that we understand and can protect against common attacks.

mon techniques used by attackers to target people online. It will also teach some of the best wa

Daniel Schwarzentraub

Tryhackme Free Walk-through Room: Common Attacks

Tryhackme Free Walk-through Room: Common Attacks

Sep 13, 2024

See all from Daniel Schwarzentraub

Recommended from Medium

Open in app ↗

Medium Search

Discover the impact of training on teams and organisations

Nov 5, 2024 60

Network Secure Protocols—TryHackMe Walkthrough

Task 1: Introduction

Oct 23, 2024 5

Lists

Staff picks

796 stories · 1560 saves

Stories to Help You Level-Up at Work

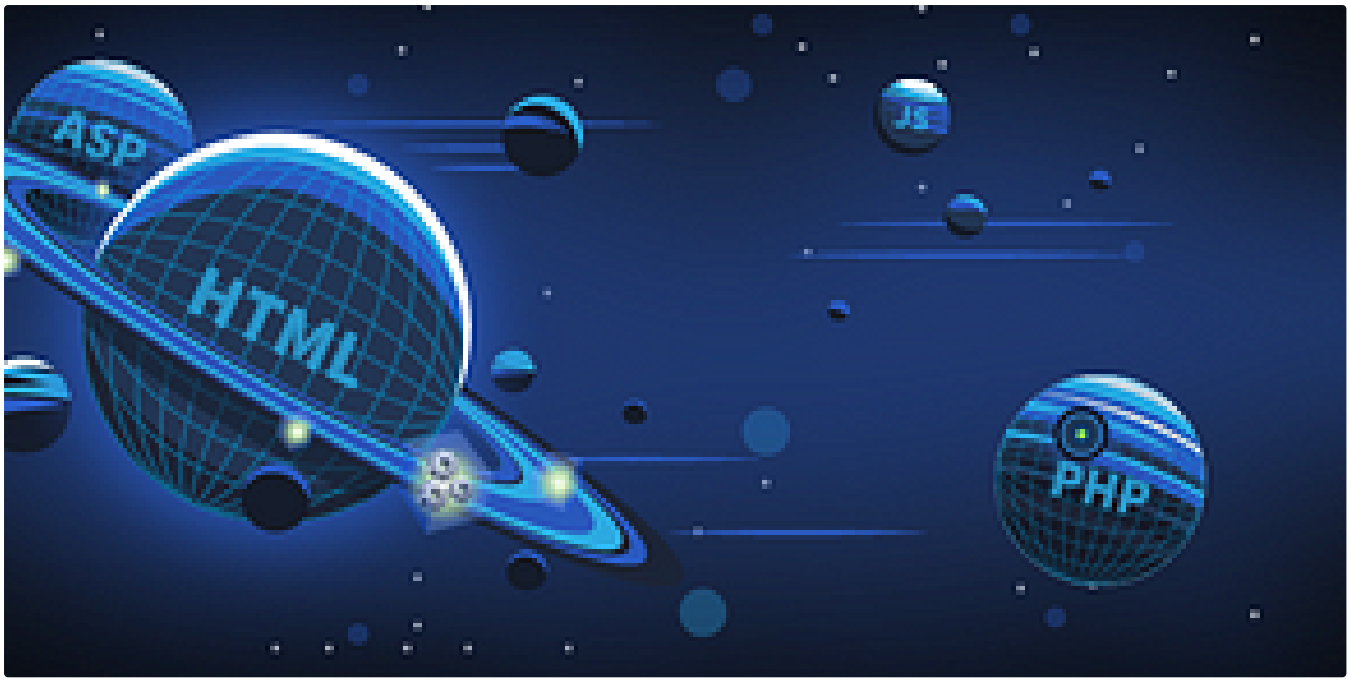
19 stories · 912 saves

Self-Improvement 101

20 stories · 3196 saves

Productivity 101

20 stories · 2707 saves

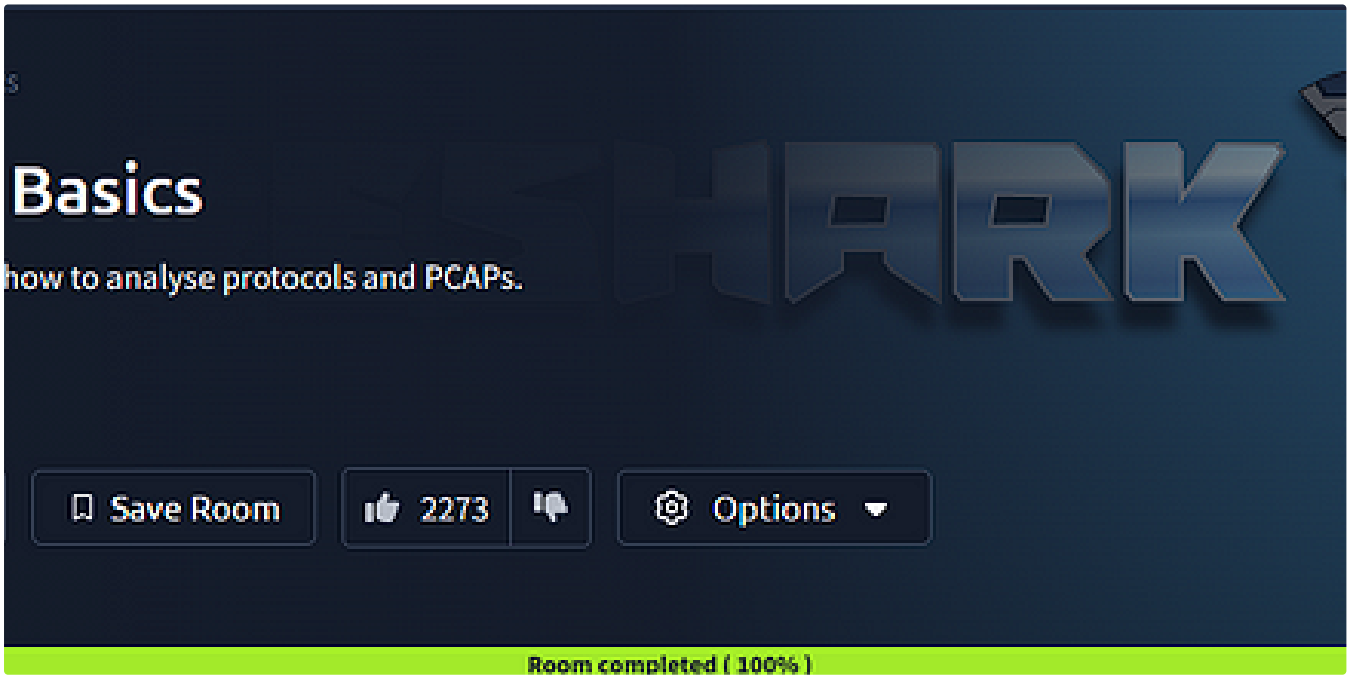


In T3CH by Axoloth

TryHackMe | Web Application Basics | WriteUp

Learn the basics of web applications: HTTP, URLs, request methods, response codes, and headers

Oct 26, 2024 56



TRedEye

Wireshark: The Basics — Tryhackme Walkthrough

Tryhackme Walkthrough

Oct 29, 2024 50

Sunny Singh Verma [SuNnY]

Linux Shells [Cyber Security 101] Learning Path TryHackMe Writeup | Detailed Walkthrough

The Room : Linux Shells is a Part of Command Line Path from the CyberSecurity 101 Learning Path on TryHackMe

Oct 27, 2024 50

IritT

Nmap — TryHackMe Insights &Walkthrough

An in depth look at scanning with Nmap, a powerful network scanning tool.

Dec 23, 2024