Networking Primer - Layers 5-7

We have seen how lower-level networking functions, now let us look at some of the upper layer protocols that handle our applications. It takes many different applications and services to maintain a network connection and ensure that data can be transferred between hosts. This section will outline just a vital few.

HTTP

Hypertext Transfer Protocol (`HTTP`) is a stateless Application Layer protocol that has been in use since 1990. HTTP enables the transfer of data in clear text between a client and server over TCP. The client would send an HTTP request to the server, asking for a resource. A session is established, and the server responds with the requested media (HTML, images, hyperlinks, video). HTTP utilizes ports 80 or 8000 over TCP during normal operations. In exceptional circumstances, it can be modified to use alternate ports, or even at times, UDP.

HTTP Methods

To perform operations such as fetching webpages, requesting items for download, or posting your most recent tweet all require the use of specific methods. These methods define the actions taken when requesting a URI. Methods:

Method	Description
`HEAD`	`required` is a safe method that requests a response from the server similar to a Get request except that the message body is not included. It is a great way to acquire more information about the server and its operational status.
`GET`	`required` Get is the most common method used. It requests information and content from the server. For example, `GET http://10.1.1.1/Webserver/index.html` requests the index.html page from the server based on our supplied URI.
`POST`	`optional` Post is a way to submit information to a server based on the fields in the request. For example, submitting a message to a Facebook post or website forum is a POST action. The actual action taken can vary based on the server, and we should pay attention to the response codes sent back to validate the action.
`PUT`	`optional` Put will take the data appended to the message and place it under the requested URI. If an item does not exist there already, it will create one with the supplied data. If an object already exists, the new PUT will be considered the most up-to-date, and the object will be modified to match. The easiest way to visualize the differences between PUT and POST is to think of it like this; PUT will create or update an object at the URI supplied, while POST will create child entities at the provided URI. The action taken can be compared with the difference between creating a new file vs. writing comments about that file on the same page.
`DELETE`	`optional` Delete does as the name implies. It will remove the object at the given URI.
`TRACE`	`optional` Allows for remote server diagnosis. The remote server will echo the same request that was sent in its response if the TRACE method is enabled.
`OPTIONS`	`optional` The Options method can gather information on the supported HTTP methods the server recognizes. This way, we can determine the requirements for interacting with a specific resource or server without actually requesting data or objects from it.
`CONNECT`	`optional` Connect is reserved for use with Proxies or other security devices like firewalls. Connect allows for tunneling over HTTP. (`SSL tunnels`)

Notice that we have `required` or `optional` listed beside each method. As a requirement by the standard, GET and HEAD must always work and exist with standard HTTP implementations. This is true only for them. The methods trace, options, delete, put and post are optional functionalities one can allow. An example of this is a read-only webpage like a blog post. The client PC can request a resource from the page but not modify, add, or delete the resource or resources.

For more information on HTTP as a protocol or how it operates, see `RFC:2616`.

HTTPS

Cheat Sheet
Resources
Go to Questions

Table of Contents

Introduction
Network Traffic Analysis
Networking Primer - Layers 1-4
Networking Primer - Layers 5-7

Analysis
The Analysis Process
Analysis in Practice

Tcpdump
Tcpdump Fundamentals
Capturing With Tcpdump (Fundamentals Labs)
Tcpdump Packet Filtering
Interrogating Network Traffic With Capture and Display Filters

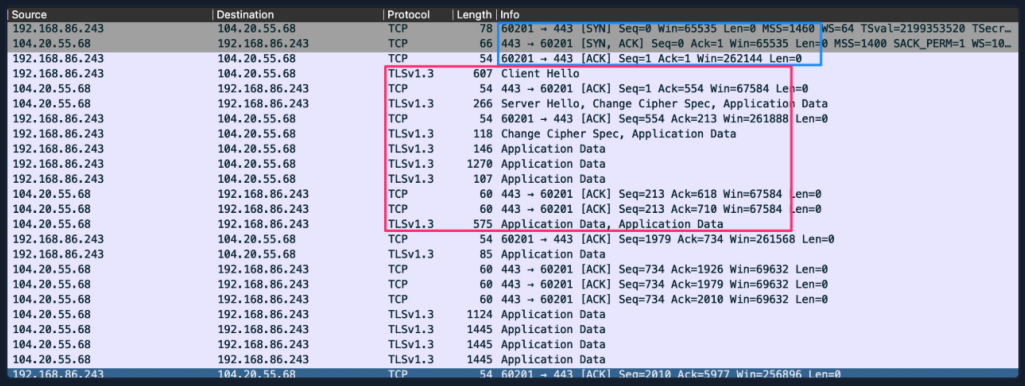
Wireshark
Analysis with Wireshark
Familiarity With Wireshark
Wireshark Advanced Usage
Packet Inception, Dissecting Network Traffic With Wireshark
Guided Lab: Traffic Analysis Workflow
Decrypting RDP connections

My Workstation
OFFLINE
Start Instance
∞ / 1 spawns left

HTTP Secure (HTTPS) is a modification of the HTTP protocol designed to utilize Transport Layer Security (TLS) or Secure Sockets Layer (SSL) with older applications for data security. TLS is utilized as an encryption mechanism to secure the communications between a client and a server. TLS can wrap regular HTTP traffic within TLS, which means that we can encrypt our entire conversation, not just the data sent or requested. Before the TLS mechanism was in place, we were vulnerable to Man-in-the-middle attacks and other types of reconnaissance or hijacking, meaning anyone in the same LAN as the client or server could view the web traffic if they were listening on the wire. We can now have security implemented in the browser enabling everyone to encrypt their web habits, search requests, sessions or data transfers, bank transactions, and much more.

Even though it is HTTP at its base, HTTPS utilizes ports 443 and 8443 instead of the standard port 80. This is a simple way for the client to signal the server that it wishes to establish a secure connection. Let's look at an output of HTTPS traffic and discern how a `TLS handshake` functions for a minute.

TLS Handshake Via HTTPS



Source	Destination	Protocol	Length	Info
192.168.86.243	104.20.55.68	TCP	78	60201 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2199353520 TSecr…
104.20.55.68	192.168.86.243	TCP	66	443 → 60201 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=10…
192.168.86.243	104.20.55.68	TCP	54	60201 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
192.168.86.243	104.20.55.68	TLSv1.3	607	Client Hello
104.20.55.68	192.168.86.243	TCP	54	443 → 60201 [ACK] Seq=1 Ack=554 Win=67584 Len=0
104.20.55.68	192.168.86.243	TLSv1.3	266	Server Hello, Change Cipher Spec, Application Data
192.168.86.243	104.20.55.68	TCP	54	60201 → 443 [ACK] Seq=554 Ack=213 Win=261888 Len=0
192.168.86.243	104.20.55.68	TLSv1.3	118	Change Cipher Spec, Application Data
192.168.86.243	104.20.55.68	TLSv1.3	146	Application Data
192.168.86.243	104.20.55.68	TLSv1.3	1270	Application Data
192.168.86.243	104.20.55.68	TLSv1.3	107	Application Data
104.20.55.68	192.168.86.243	TCP	60	443 → 60201 [ACK] Seq=213 Ack=618 Win=67584 Len=0
104.20.55.68	192.168.86.243	TCP	60	443 → 60201 [ACK] Seq=213 Ack=710 Win=67584 Len=0
104.20.55.68	192.168.86.243	TLSv1.3	575	Application Data, Application Data
192.168.86.243	104.20.55.68	TCP	54	60201 → 443 [ACK] Seq=1979 Ack=734 Win=261568 Len=0
192.168.86.243	104.20.55.68	TLSv1.3	85	Application Data
104.20.55.68	192.168.86.243	TCP	60	443 → 60201 [ACK] Seq=734 Ack=1926 Win=69632 Len=0
104.20.55.68	192.168.86.243	TCP	60	443 → 60201 [ACK] Seq=734 Ack=1979 Win=69632 Len=0
104.20.55.68	192.168.86.243	TCP	60	443 → 60201 [ACK] Seq=734 Ack=2010 Win=69632 Len=0
104.20.55.68	192.168.86.243	TLSv1.3	1124	Application Data
104.20.55.68	192.168.86.243	TLSv1.3	1445	Application Data
104.20.55.68	192.168.86.243	TLSv1.3	1445	Application Data
104.20.55.68	192.168.86.243	TLSv1.3	1445	Application Data
192.168.86.243	104.20.55.68	TCP	54	60201 → 443 [ACK] Seq=2010 Ack=5977 Win=256896 Len=0

In the first few packets, we can see that the client establishes a session to the server using port 443 `boxed in blue`. This signals the server that it wishes to use HTTPS as the application communication protocol.

Once a session is initiated via TCP, a TLS ClientHello is sent next to begin the TLS handshake. During the handshake, several parameters are agreed upon, including session identifier, peer x509 certificate, compression algorithm to be used, the cipher spec encryption algorithm, if the session is resumable, and a 48-byte master secret shared between the client and server to validate the session.

Once the session is established, all data and methods will be sent through the TLS connection and appear as TLS Application Data `as seen in the red box`. TLS is still using TCP as its transport protocol, so we will still see acknowledgment packets from the stream coming over port 443.

To summarize the handshake:

1. Client and server exchange hello messages to agree on connection parameters.
2. Client and server exchange necessary cryptographic parameters to establish a premaster secret.
3. Client and server will exchange x.509 certificates and cryptographic information allowing for authentication within the session.
4. Generate a master secret from the premaster secret and exchanged random values.
5. Client and server issue negotiated security parameters to the record layer portion of the TLS protocol.
6. Client and server verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

Encryption in itself is a complex and lengthy topic that deserves its own module. This section is a simple summary of how HTTP and TLS provide security within the HTTPS application protocol. For more information on how HTTPS functions and how TLS performs security operations, see `RFC:2246`.

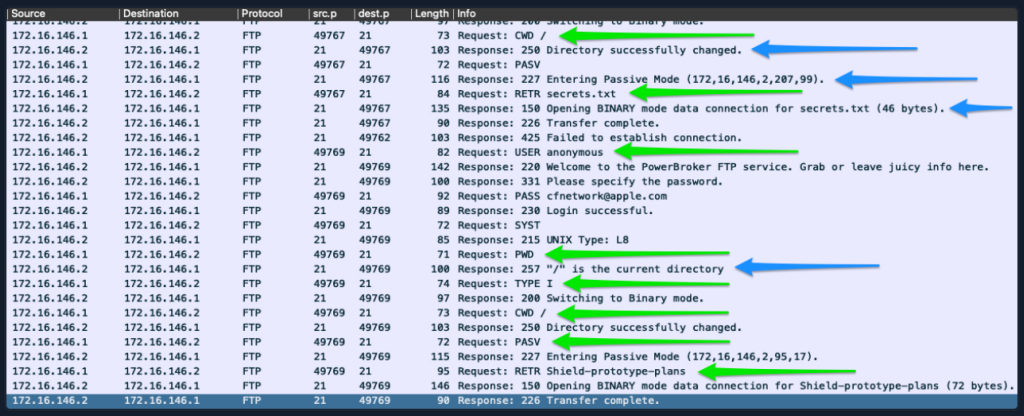
FTP

File Transfer Protocol (FTP) is an Application Layer protocol that enables quick data transfer between computing devices. FTP can be utilized from the command-line, web browser, or through a graphical FTP client such as FileZilla. FTP itself is established as an insecure protocol, and most users have moved to utilize tools such as SFTP to transfer files through secure channels. As a note moving into the future, most modern web browsers have phased out support for FTP as of 2020.

When we think about communication between hosts, we typically think about a client and server talking over a single socket. Through this socket, both the client and server send commands and data over the same link. In this aspect, FTP is unique since it utilizes multiple ports at a time. FTP uses ports 20 and 21 over TCP. Port 20 is used for data transfer, while port 21 is utilized for issuing commands controlling the FTP session. In regards to authentication, FTP supports user authentication as well as allowing anonymous access if configured.

FTP is capable of running in two different modes, `active` or `passive`. Active is the default operational method utilized by FTP, meaning that the server listens for a control command `PORT` from the client, stating what port to use for data transfer. Passive mode enables us to access FTP servers located behind firewalls or a NAT-enabled link that makes direct TCP connections impossible. In this instance, the client would send the `PASV` command and wait for a response from the server informing the client what IP and port to utilize for the data transfer channel connection.

FTP Command & Response Examples



Source	Destination	Protocol	src.p	dest.p	Length	Info
172.16.146.1	172.16.146.2	FTP	49767	21	73	Request: CWD /
172.16.146.2	172.16.146.1	FTP	21	49767	103	Response: 250 Directory successfully changed.
172.16.146.1	172.16.146.2	FTP	49767	21	72	Request: PASV
172.16.146.2	172.16.146.1	FTP	21	49767	116	Response: 227 Entering Passive Mode (172,16,146,2,207,99).
172.16.146.1	172.16.146.2	FTP	49767	21	84	Request: RETR secrets.txt
172.16.146.2	172.16.146.1	FTP	21	49767	135	Response: 150 Opening BINARY mode data connection for secrets.txt (46 bytes).
172.16.146.2	172.16.146.1	FTP	21	49767	90	Response: 226 Transfer complete.
172.16.146.2	172.16.146.1	FTP	21	49762	103	Response: 425 Failed to establish connection.
172.16.146.1	172.16.146.2	FTP	49769	21	82	Request: USER anonymous
172.16.146.2	172.16.146.1	FTP	21	49769	142	Response: 220 Welcome to the PowerBroker FTP service. Grab or leave juicy info here.
172.16.146.2	172.16.146.1	FTP	21	49769	100	Response: 331 Please specify the password.
172.16.146.1	172.16.146.2	FTP	49769	21	92	Request: PASS cfnetwork@apple.com
172.16.146.2	172.16.146.1	FTP	21	49769	89	Response: 230 Login successful.
172.16.146.1	172.16.146.2	FTP	49769	21	72	Request: SYST
172.16.146.2	172.16.146.1	FTP	21	49769	85	Response: 215 UNIX Type: L8
172.16.146.1	172.16.146.2	FTP	49769	21	71	Request: PWD
172.16.146.2	172.16.146.1	FTP	21	49769	100	Response: 257 "/" is the current directory
172.16.146.1	172.16.146.2	FTP	49769	21	74	Request: TYPE I
172.16.146.2	172.16.146.1	FTP	21	49769	97	Response: 200 Switching to Binary mode.
172.16.146.1	172.16.146.2	FTP	49769	21	73	Request: CWD /
172.16.146.2	172.16.146.1	FTP	21	49769	103	Response: 250 Directory successfully changed.
172.16.146.1	172.16.146.2	FTP	49769	21	72	Request: PASV
172.16.146.2	172.16.146.1	FTP	21	49769	115	Response: 227 Entering Passive Mode (172,16,146,2,95,17).
172.16.146.1	172.16.146.2	FTP	49769	21	95	Request: RETR Shield-prototype-plans
172.16.146.2	172.16.146.1	FTP	21	49769	146	Response: 150 Opening BINARY mode data connection for Shield-prototype-plans (72 bytes).
172.16.146.2	172.16.146.1	FTP	21	49769	90	Response: 226 Transfer complete.

The image above shows several examples of requests issued over the FTP command channel `green arrows`, and the responses sent back from the FTP server `blue arrows`. This is all pretty standard stuff. For a list of each command and what it is doing, check out the table below.

When looking at FTP traffic, some common commands we can see passed over port 21 include:

FTP Commands

Command	Description
`USER`	specifies the user to log in as.
`PASS`	sends the password for the user attempting to log in.
`PORT`	when in active mode, this will change the data port used.
`PASV`	switches the connection to the server from active mode to passive.
`LIST`	displays a list of the files in the current directory.
`CWD`	will change the current working directory to one specified.
`PWD`	prints out the directory you are currently working in.
`SIZE`	will return the size of a file specified.
`RETR`	retrieves the file from the FTP server.
`QUIT`	ends the session.

This is not an exhaustive list of the possible FTP control commands that could be seen. These can vary based on the

FTP application or shell in use. For more information on FTP, see RFC:959.

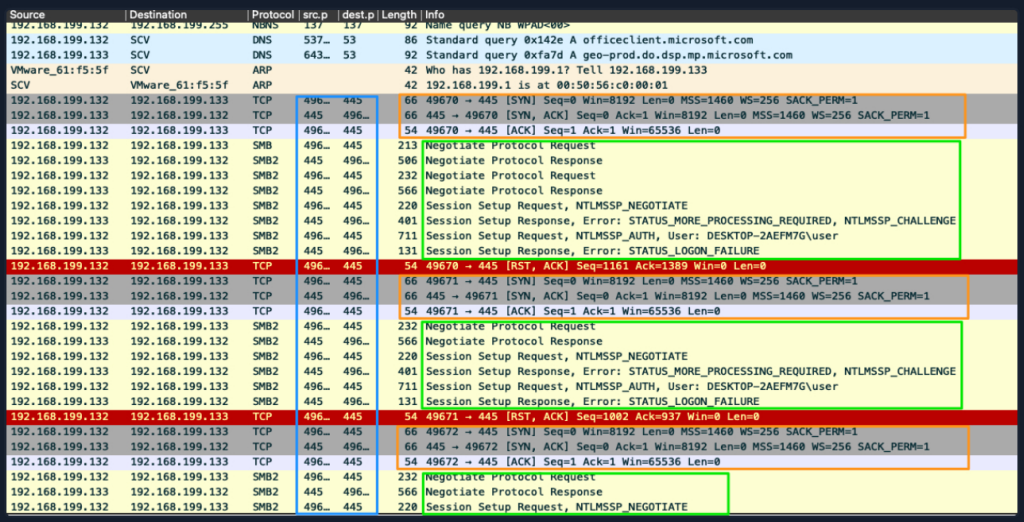
SMB

Server Message Block (SMB) is a protocol most widely seen in Windows enterprise environments that enables sharing resources between hosts over common networking architectures. SMB is a connection-oriented protocol that requires user authentication from the host to the resource to ensure the user has correct permissions to use that resource or perform actions. In the past, SMB utilized NetBIOS as its transport mechanism over UDP ports 137 and 138. Since modern changes, SMB now supports direct TCP transport over port 445, NetBIOS over TCP port 139, and even the QUIC protocol.

As a user, SMB provides us easy and convenient access to resources like printers, shared drives, authentication servers, and more. For this reason, SMB is very attractive to potential attackers as well.

Like any other application that uses TCP as its transport mechanism, it will perform standard functions like the three-way handshake and acknowledging received packets. Let us take a second to look at some SMB traffic to familiarize ourselves.

SMB On The Wire



Source	Destination	Protocol	src.p	dest.p	Length	Info
192.168.199.132	192.168.199.255	NBNS	137	137	92	Name query NB WPAD<00>
192.168.199.132	SCV	DNS	537…	53	86	Standard query 0x142e A officeclient.microsoft.com
192.168.199.133	SCV	DNS	643…	53	92	Standard query 0xfa7d A geo-prod.do.dsp.mp.microsoft.com
VMware_61:f5:5f	SCV	ARP			42	Who has 192.168.199.1? Tell 192.168.199.133
SCV	VMware_61:f5:5f	ARP			42	192.168.199.1 is at 00:50:56:c0:00:01
192.168.199.132	192.168.199.133	TCP	496…	445	66	49670 → 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.199.133	192.168.199.132	TCP	445	496…	66	445 → 49670 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.199.132	192.168.199.133	TCP	496…	445	54	49670 → 445 [ACK] Seq=1 Ack=1 Win=65536 Len=0
192.168.199.132	192.168.199.133	SMB	496…	445	213	Negotiate Protocol Request
192.168.199.133	192.168.199.132	SMB2	445	496…	506	Negotiate Protocol Response
192.168.199.132	192.168.199.133	SMB2	496…	445	232	Negotiate Protocol Request
192.168.199.133	192.168.199.132	SMB2	445	496…	566	Negotiate Protocol Response
192.168.199.132	192.168.199.133	SMB2	496…	445	220	Session Setup Request, NTLMSSP_NEGOTIATE
192.168.199.133	192.168.199.132	SMB2	445	496…	401	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
192.168.199.132	192.168.199.133	SMB2	496…	445	711	Session Setup Request, NTLMSSP_AUTH, User: DESKTOP-2AEFM7G\user
192.168.199.133	192.168.199.132	SMB2	445	496…	131	Session Setup Response, Error: STATUS_LOGON_FAILURE
192.168.199.132	192.168.199.133	TCP	496…	445	54	49670 → 445 [RST, ACK] Seq=1161 Ack=1389 Win=0 Len=0
192.168.199.132	192.168.199.133	TCP	496…	445	66	49671 → 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.199.133	192.168.199.132	TCP	445	496…	66	445 → 49671 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.199.132	192.168.199.133	TCP	496…	445	54	49671 → 445 [ACK] Seq=1 Ack=1 Win=65536 Len=0
192.168.199.132	192.168.199.133	SMB2	496…	445	232	Negotiate Protocol Request
192.168.199.133	192.168.199.132	SMB2	445	496…	566	Negotiate Protocol Response
192.168.199.132	192.168.199.133	SMB2	496…	445	220	Session Setup Request, NTLMSSP_NEGOTIATE
192.168.199.133	192.168.199.132	SMB2	445	496…	401	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
192.168.199.132	192.168.199.133	SMB2	496…	445	711	Session Setup Request, NTLMSSP_AUTH, User: DESKTOP-2AEFM7G\user
192.168.199.133	192.168.199.132	SMB2	445	496…	131	Session Setup Response, Error: STATUS_LOGON_FAILURE
192.168.199.132	192.168.199.133	TCP	496…	445	54	49671 → 445 [RST, ACK] Seq=1002 Ack=937 Win=0 Len=0
192.168.199.132	192.168.199.133	TCP	496…	445	66	49672 → 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.199.133	192.168.199.132	TCP	445	496…	66	445 → 49672 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.199.132	192.168.199.133	TCP	496…	445	54	49672 → 445 [ACK] Seq=1 Ack=1 Win=65536 Len=0
192.168.199.132	192.168.199.133	SMB2	496…	445	232	Negotiate Protocol Request
192.168.199.133	192.168.199.132	SMB2	445	496…	566	Negotiate Protocol Response
192.168.199.132	192.168.199.133	SMB2	496…	445	220	Session Setup Request, NTLMSSP_NEGOTIATE

Looking at the image above, we can see that it performs the TCP handshake each time it establishes a session `orange boxes`. When looking at the source and destination ports `blue box`, port 445 is being utilized, signaling SMB traffic over TCP. If we look at the `green boxes,` the info field tells us a bit about what is happening in the SMB communication. In this example, there are many errors, which is an example of something to dig deeper into. One or two auth failures from a user is relatively common, but a large cluster of them repeating can signal a potential unauthorized individual trying to access a user's account or use their credentials to move. This is a common tactic of attackers, grab an authenticated user, steal their credentials, utilize them to move laterally, or access resources they typically would be denied access to.

This is just one example of SMB use. Another common thing we will see is file-share access between servers and hosts. For the most part, this is regular communication. However, if we see a host access file shares on other hosts, this is not common. Please pay attention to who is requesting connections, where to, and what they are doing.

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

+ 0 What is the default operational mode method used by FTP?

active

Submit Hint

+ 0 FTP utilizes what two ports for command and data transfer? (separate the two numbers with a space)

20 21

Submit Hint

+ 0 Does SMB utilize TCP or UDP as its transport layer protocol?

TCP

Submit Hint

+ 0 SMB has moved to using what TCP port?

445

Submit Hint

+ 0 Hypertext Transfer Protocol uses what well known TCP port number?

80

Submit Hint

+ 0 What HTTP method is used to request information and content from the webserver?

GET

Submit Hint

+ 0 What web based protocol uses TLS as a security measure?

HTTPS

Submit Hint

+ 0 True or False: when utilizing HTTPS, all data sent across the session will appear as TLS Application data?

True

Submit Hint