Application Security: HTTPS and HSTS

Hypertext transfer protocol secure (HTTPS)

What is HTTPS?

Hypertext transfer protocol secure (HTTPS) is the secure version of HTTP, which is the primary protocol used to send data between a web browser and a website.

HTTPS is encrypted in order to increase security of data transfer.

This is particularly important when users transmit sensitive data, such as by logging into a bank account, email service, or health insurance provider.

Any website, especially those that require login credentials, should use HTTPS. In modern web browsers such as Chrome, websites that do not use HTTPS are marked differently than those that are.

Look for a padlock in the URL bar to signify the webpage is secure.

Web browsers take HTTPS seriously; Google Chrome and other browsers flag all non-HTTPS websites as not secure.

How does HTTPS work?

HTTPS uses an encryption protocol to encrypt communications.

The protocol is called Transport Layer Security (TLS), although formerly it was known as Secure Sockets Layer (SSL).

This protocol secures communications by using what's known as an asymmetric public key infrastructure.

This type of security system uses two different keys to encrypt communications between two parties:

- The private key - this key is controlled by the owner of a website and it's kept, as the reader may have speculated, private. This key lives on a web server and is used to decrypt information encrypted by the public key.
- The public key - this key is available to everyone who wants to interact with the server in a way that's secure. Information that's encrypted by the public key can only be decrypted by the private key.

Why is HTTPS important? What happens if a website doesn’t have HTTPS?

HTTPS prevents websites from having their information broadcast in a way that’s easily viewed by anyone snooping on the network.

When information is sent over regular HTTP, the information is broken into packets of data that can be easily “sniffed” using free software.

This makes communication over the an unsecure medium, such as public Wi-Fi, highly vulnerable to interception.

In fact, all communications that occur over HTTP occur in plain text, making them highly accessible to anyone with the correct tools, and vulnerable to on-path attacks.

With HTTPS, traffic is encrypted such that even if the packets are sniffed or otherwise intercepted, they will come across as nonsensical characters.

In websites without HTTPS, it is possible for Internet service providers (ISPs) or other intermediaries to inject content into web pages without the approval of the website owner.

This commonly takes the form of advertising, where an ISP looking to increase revenue injects paid advertising into the webpages of their customers.

Unsurprisingly, when this occurs, the profits for the advertisements and the quality control of those advertisements are in no way shared with the website owner.

HTTPS eliminates the ability of unmoderated third parties to inject advertising into web content.

What port does HTTPS use?

HTTPS uses port 443. This differentiates HTTPS from HTTP, which uses port 80.

(In networking, a port is a virtual software-based point where network connections start and end. All network-connected computers expose a number of ports to enable them to receive traffic. Each port is associated with a specific process or service, and different protocols use different ports.)

How else is HTTPS different from HTTP?

HTTPS is not a separate protocol from HTTP.

It is simply using TLS/SSL encryption over the HTTP protocol. HTTPS occurs based upon the transmission of TLS/SSL certificates, which verify that a particular provider is who they say they are.

When a user connects to a webpage, the webpage will send over its SSL certificate which contains the public key necessary to start the secure session.

The two computers, the client and the server, then go through a process called an SSL/TLS handshake, which is a series of back-and-forth communications used to establish a secure connection.

To take a deeper dive into encryption and the SSL/TLS handshake, read about what happens in a TLS handshake.

Advantages of HTTPS

Following are the advantages or benefits of a Hypertext Transfer Protocol Secure (HTTPS):

- The main advantage of HTTPS is that it provides high security to users.
- Data and information are protected. So, it ensures data protection.
- SSL technology in HTTPS protects the data from third-party or hackers. And this technology builds trust for the users who are using it.
- It helps users by performing banking transactions.

Disadvantages of HTTPS

Following are the disadvantages or limitations of a Hypertext Transfer Protocol Secure (HTTPS):

- The big disadvantage of HTTPS is that users need to purchase the SSL certificate.
- The speed of accessing the website is slow because there are various complexities in communication.
- Users need to update all their internal links.

Difference between HTTP and HTTPS

HTTPS is an abbreviation of Hypertext Transfer Protocol Secure. It is a secure extension or version of HTTP.

This protocol is mainly used for providing security to the data sent between a website and the web browser.

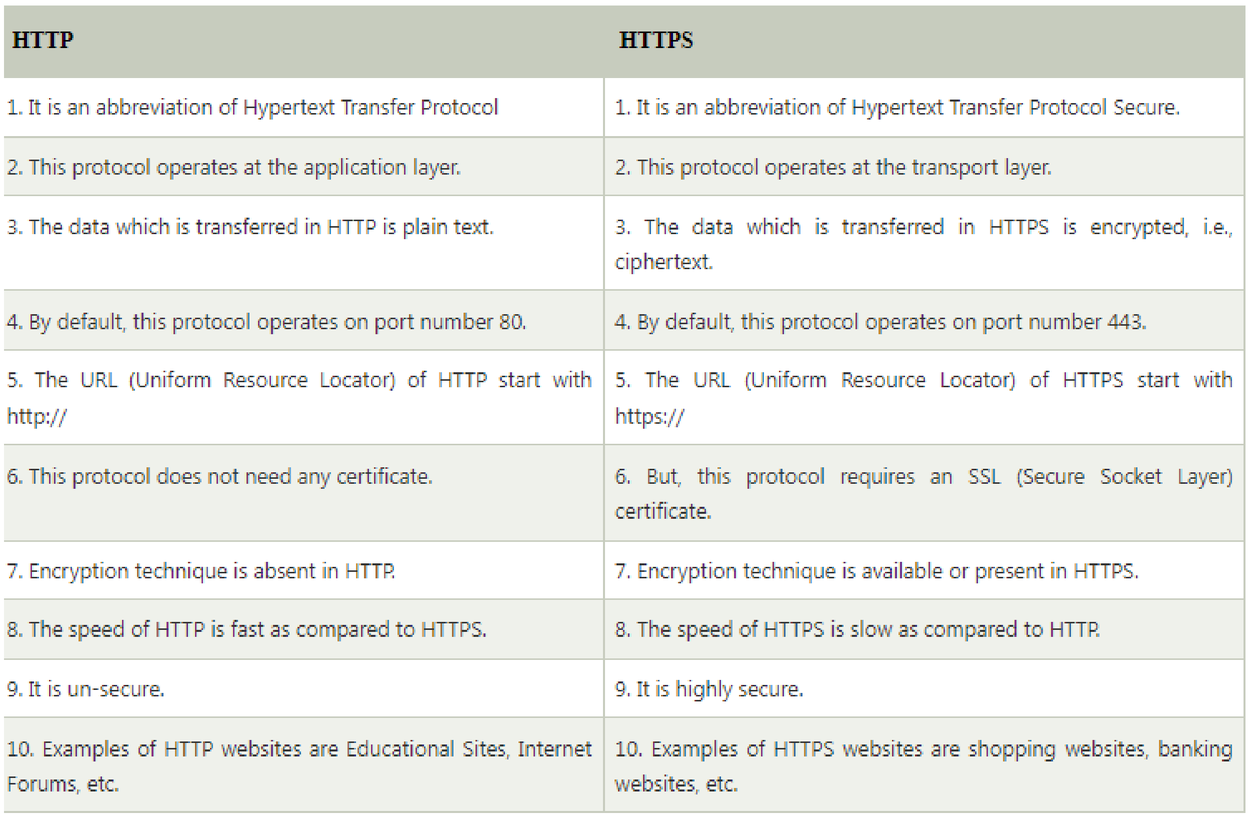
It is widely used on the internet and used for secure communications. This protocol uses the 443 port number for communicating the data.

This protocol is also called HTTP over SSL because the HTTPS communication protocols are encrypted using the SSL (Secure Socket Layer).

By default, it is supported by various web browsers.

Those websites which need login credentials should use the HTTPS protocol for sending the data.

It allows users to create a secured encrypted connection and helps them to protect their information from being stolen.



HTTP	HTTPS
1. It is an abbreviation of Hypertext Transfer Protocol	1. It is an abbreviation of Hypertext Transfer Protocol Secure.
2. This protocol operates at the application layer.	2. This protocol operates at the transport layer.
3. The data which is transferred in HTTP is plain text.	3. The data which is transferred in HTTPS is encrypted, i.e., ciphertext.
4. By default, this protocol operates on port number 80.	4. By default, this protocol operates on port number 443.
5. The URL (Uniform Resource Locator) of HTTP start with http://	5. The URL (Uniform Resource Locator) of HTTPS start with https://
6. This protocol does not need any certificate.	6. But, this protocol requires an SSL (Secure Socket Layer) certificate.
7. Encryption technique is absent in HTTP.	7. Encryption technique is available or present in HTTPS.
8. The speed of HTTP is fast as compared to HTTPS.	8. The speed of HTTPS is slow as compared to HTTP.
9. It is un-secure.	9. It is highly secure.
10. Examples of HTTP websites are Educational Sites, Internet Forums, etc.	10. Examples of HTTPS websites are shopping websites, banking websites, etc.

HSTS (HTTP Strict Transport Security)

What is HSTS (HTTP Strict Transport Security)?

HTTP Strict Transport Security (HSTS) is a web security policy mechanism that enables web sites to declare themselves accessible only via secure connections.

This helps protect websites and users from protocol downgrade and cookie hijacking attacks.

Why Was HSTS Introduced?

HTTP is used over various transports, typically the Transmission Control Protocol (TCP).

However, TCP does not provide integrity protection, confidentiality or secure host identification.

This led to the development of Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS).

SSL/TLS provide an encryption layer between application protocols and TCP, commonly known as HTTPS.

In general, user agents (like web browsers) will employ various local security policies to decide how to interact with a host, based on a negotiation between the server, user preferences and their communication method (HTTP or HTTPS).

However, some user agents allow users to choose to continue to interact with a website when they are unable to establish a secure connection.

This could happen when a TLS certificate's trust chain is not validated, when it has expired or when the TLS host's domain name appears incorrectly in the TLS certificate.

This behavior is called click-through insecurity.

While giving users the option to continue to use a website despite a lack of HTTPS can keep users happy, it can introduce attack vectors that leave users open to certain types of cyber attacks, particularly man-in-the-middle attacks (MITM attacks), downgrade attacks and session hijacking attacks.

SSL-Stripping

As HSTS allows websites to declare they are only accessible through a secure connection, they can prevent users from connecting to them over any HTTP connection.

This prevents a security vulnerability known as SSL-stripping.

SSL-stripping is a downgrade attack that was introduced by Moxie Marlinspike in his 2009 BlackHat Federal talk New Tricks for Defeating SSL in Practice.

A downgrade attack is a form of cryptographic attack on a computer system or in this case, a communications protocol that makes it abandon its encrypted connection (HTTPS) in favor of an older, unencrypted connection (HTTP) that is typically provided for backwards compatibility with older systems.

SSL-stripping is implemented as part of a man-in-the-middle attack where web traffic is intercepted and redirected from the secure HTTPS version of the website to an unencrypted HTTP version.

The primary reason this attack continues to be successful is that many websites continue to not use TLS/SSL certificates.

This makes it impossible to know (without prior knowledge) whether a website's lack of HTTPS is due to an SSL-stripping attack or because they don't have a TLS certificate.

Additionally, there are no warnings to warn the user during the downgrade process, making the attack hard to detect even for the most vigilant user.

With the creation of a tool by Marlinspike to fully automate this type of attack, it represents a real cyber security risk.

Session Hijacking

Session hijacking or cookie hijacking is another vulnerability that is enabled through click-through insecurity.

Session hijacking exploits a valid computer session to gain unauthorized access to information or services.

This is particularly relevant for web developers as cookies are used to maintain a session on many websites.

If a website does not flag their cookies as Secure, telling user agents to only send cookies over HTTPS, they can be easily stolen by an attacker.

As non-Secure cookies are returned to the host regardless of transport security, leaving them open to man-in-the-middle attacks.

Once an attacker has access to the cookies, they can then impersonate the user on a legitimate website.

How Does HSTS Work?
HSTS enables web servers to declare that any interactions by web browsers and other user agents must be conducted over HTTPS connections and not insecure HTTP connections.

A server can implement an HSTS Policy by supplying a response header over an HTTPS connection (HSTS headers sent over HTTP response headers are ignored).

The HSTS header is name "Strict-Transport-Security and also specifies a period of time during which the user agent should only access the service via HTTPS requests.

This means the first time a site is accessed using HTTPS it returns the Strict-Transport-Security header, the browser records this information, so future attempts to load the site using HTTP automatically use HTTPS.

When the expiration time specified by the Strict-Transport-Security header elapses, the next attempt to load the site via HTTP will proceed as normal instead of automatically using HTTPS.

However, whenever the Strict-Transport-Security header is delivered to the user agent, it will update the expiration time for that site, so sites can refresh this information and prevent the timeout from expiring.

Should it be necessary to disable HSTS, web servers can set the max-age to 0 (over a HTTPS connection) to immediately expire the HSTS header, allowing access via HTTP requests.

For example, a server could send a header that requests that future requests for the next year only use HTTPS via Strict-Transport-Security: max-age=31536000

When a web application issues a HSTS Policy to user agents, conforming user agents behave as follows:

- Any insecure links are automatically turn into secure links (e.g. http://example.com/ will be modified to https://example.com before accessing the server)
- If a secure connection cannot be ensured (e.g. the server does not have a valid certificate), the user agent will terminate the connection and not allow the user to access the website.

The most important thing to understand is that a HSTS Policy prevents click-through insecurity by not allowing the end user to use the insecure connection.

What is an Example Situation Involving HSTS?

Imagine your staff member uses a free WiFi access point at a cafe and starts surfing the web, visiting your organization's payroll system.

Unfortunately, the access point they are using is actually an attacker's laptop and they're intercepting the original HTTP request and redirecting your employee to a clone of your payroll system instead of the real thing, exposing your employees' personally identifiable information (PII).

If your payroll system uses HSTS and your employee has visited it once using HTTPS, then their browser will know to only use HTTPS, preventing this type of man-in-the-middle attack.

What are the Limitations of HSTS?

A key limitation of using HSTS is that a user that cannot connect through HTTPS will be unable to use the site.

Additionally, as the HSTS Policy is communicated in a response header, it requires the user agent to first visit the website to learn that it uses HSTS.

This means the initial request remains unprotected from active attacks if it uses an insecure protocol such as plain HTTP or if the URI for the initial request was obtained over an insecure channel.

This will also apply to the first request after the activity period specified in the HSTS max-age (sites generally set a period of several days or months depending on user activity and behavior).

There is widespread browser support for HSTS including Google Chrome, Mozilla Firefox, Internet Explorer, Microsoft Edge, Opera, and Safari address this limitation by preloading HSTS Policies from the HSTS Preload list.

The HSTS list contains known websites that support HSTS and are distributed with the browser so it uses HTTPS for the initial request for the listed websites.

As this approach can never scale across the entire Web, there have been discussions to be able to declare HSTS in DNS records and be access them securely via DNSSEC, which could ensure validity.

Additionally, HSTS is ineffective against typosquatting domains, DNS-based attacks and man-in-the-middle attacks that serve traffic from an artificial domain that is not on the HSTS Preload list.

And as HSTS relies on TLS itself, it also relies on the security of TLS.

Read RFC 6797 for a deeper discussion of the overall HSTS security considerations.