**Stripping away the encryption offered by HTTPS, called SSL Strip, is a serious cyber threat to many corporations since their employees are constantly on the move and require access to Internet on-the-go even through open non-secure Wi-Fi hotspots. Once attackers gain access to a network, they can act as a Man-in-the-Middle (MITM) to intercept connections over the network. These interception tactics can also be deployed against wired networks, provided that someone gains access to an Ethernet port.**

**The**[**KRACK Attack**](https://www.krackattacks.com/)**effectively**[**demonstrated**](https://www.youtube.com/watch?v=Oh4WURZoR98)**that corporate users can’t blindly trust the medium that connects them to the internet. It also illustrates how encryption can be completely stripped away even if the site supports HTTPS.**

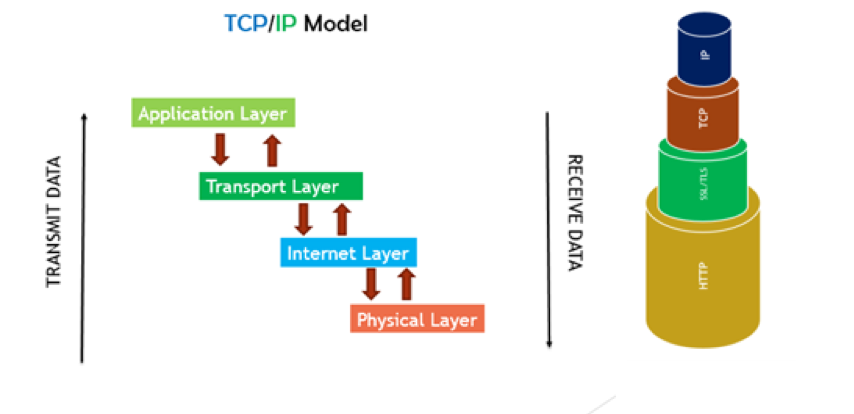
[**Uncover unknown, unintended or malicious TLS certificates. Try OutagePREDICT now.**](https://www.venafi.com/venaficloud/outagepredict?utm_source=blog&utm_medium=CTA&utm_campaign=DevOps-malicious-TLS-Graphical-CTA#form)

A bit of history

**The creator of SSL strip vulnerability is Moxie Marlinspike, a well-known American computer security researcher. In 2009, he**[**spoke**](https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf)**about this dangerous SSL weakness for the first time at the Black Hat information security event. According to Marlinspike’s**[**presentation**](https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf)**, the exploitation of this vulnerability is very serious threat for the privacy of our digital credentials since it can**[**happen in real time**](https://www.venafi.com/blog/what-active-attack-vs-passive-attack-using-encryption)**, undetected, and targets whatever secure sites people are browsing to at any moment. It doesn't require multiple certificates and once the attacker gets his “dirty work” done, he can switch the victims back to a normal traffic stream.**

A bit of theory

**HTTP and HTTPS are the application-layer protocols in a**[**TCP/IP model**](https://en.wikipedia.org/wiki/Internet_protocol_suite)**, as illustrated in the figure below. HTTPS uses a secure tunnel to transfer and receive data which is commonly called SSL/TLS (Secure Socket Layer / Transport Layer Security), and therefore the suffix ‘S’ is added to HTTPS.**

****

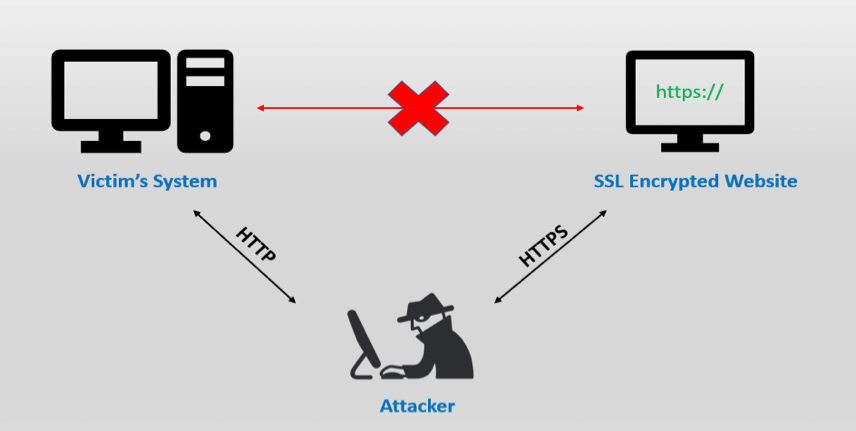
[**SSL/TLS**](https://www.venafi.com/solutions/encryption-and-authorization)**is a secure protocol used to communicate sensitive information. This protocol is used when exchanging sensitive data such as banking information and email correspondence for example. The protocol’s security is established by creating an encrypted connection between two parties (usually a client application and a server). Browsers and web servers regularly use this protocol when a secure connection is needed. In most scenarios the**[**following events**](https://blog.cloudflare.com/performing-preventing-ssl-stripping-a-plain-english-primer/)**take place when establishing a secure connection:**

1. **The user sends an unsecured HTTP request.**
2. **The server answers via HTTP and redirects the user to a secure protocol (HTTPS).**
3. **The user sends a secure HTTPS request, and the secure session begins.**

**This process provides a**[**reasonable guarantee of both privacy and integrity**](https://www.venafi.com/blog/what-session-hijacking)**. In other words, we don't just encrypt the messages we're sending, we make sure the message we receive isn't altered over the wire.**

How the SSL strip attack works

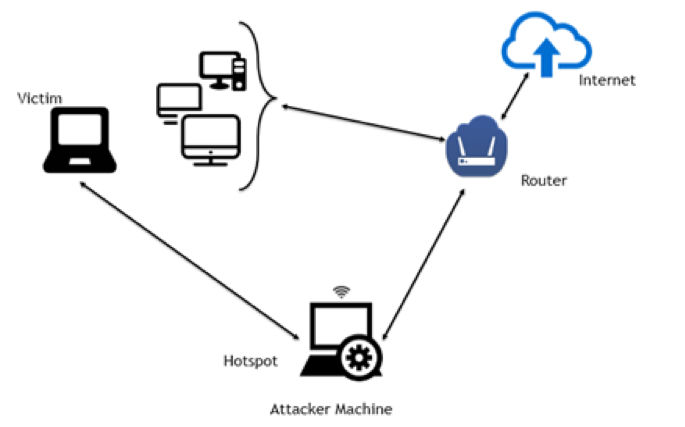
**In order to**[**“strip” the SSL**](https://avicoder.me/2016/02/22/SSLstrip-for-newbies/)**, an attacker intervenes in the redirection of the HTTP to the secure HTTPS protocol and intercepts a request from the user to the server. The attacker will then continue to establish an HTTPS connection between himself and the server, and an unsecured HTTP connection with the user, acting as a “bridge” between them.**

****

**How can the SSL Strip trick both the browser and the website’s server? The SSL Strip takes advantage of the way most users come to SSL websites. The majority of visitors connect to a website’s page that redirects through a 302 redirect, or they arrive on an SSL page via a link from a non-SSL site. If the victim wants, for instance, to buy a product and types the URL www.buyme.com in the address bar, the browser connects to the attacker machine and waits for a response from the server. In an SSL Strip, the attacker, in turn, forwards the victim’s request to the online shop’s server and receives the secure HTTPS payment page. For example: https://www.buyme.com. At this point, the attacker has complete control over the secure payment page. He downgrades it from HTTPS to HTTP and sends it back to the victim’s browser. The browser is now redirected to http://www.buyme.com. From now onward, all the victim’s data will be transferred in plain text format, and the attacker will be able to intercept it. Meanwhile, the website’s server will think that it has successfully established the secure connection, which indeed it has—but with the attacker’s machine, not the victim’s.**

Why are open Wi-Fi hotspots dangerous?

**SSL Strip attacks can be implemented in a number of ways. The most common method is by**[**creating a hotspot**](https://avicoder.me/2016/02/22/SSLstrip-for-newbies/)**and allowing the victims to connect to it. Many attackers establish fake hotspots with names similar to legitimate hotspot names, for example, “Starbucks Coffee” instead of “Starbucks”. Unaware, the user connects to the malicious hotspot. Once the user tries to connect to the server, the attacker uses his control over the hotspot and attacks the user.**

****

What is the threat of SSL stripping attacks?

**After the successful implementation of an SSL strip attack, the victim’s information is transferred in plain text format and can be easily intercepted by anyone, including the attacker. This results in a breach in the integrity and confidentiality of personal identifiable information (PII) such as login credentials,**[**bank accounts**](https://www.venafi.com/education-center/data-breach/cost-of-a-data-breach)**, sensitive business data, etc. Hence the threat of this vulnerability is easily understood and may have varying implications to your**[**digital presence**](https://www.venafi.com/solutions/encryption-and-authorization)**. Your business relies on encrypted communications to transact securely across the edge to the endpoint. But what if you can’t trust the identifying certificates on each end of the channel? Without this trust, you can’t engage in e-commerce web transactions and online banking that your consumers now rely on without having a second thought about security.**

**SSL stripping attacks**[**can work only**](https://www.ssldragon.com/blog/what-is-the-ssl-strip-and-how-you-can-prevent-it/)**on websites that encrypt only their login page. Hence, websites that use both HTTP and HTTPS in their setup are vulnerable to SSL stripping attacks. The question to be answered now is this: what can we do to**[**secure ourselves**](https://www.venafi.com/solutions/encryption-and-authorization)**against this threat? Is the adoption of HTTPS and the Chrome updates a panacea?**

Enable SSL site wide at all websites

**To mitigate this threat, financial institutions and technology firms have**[**already enabled**](https://www.venafi.com/blog/https-should-be-implemented-everywhereincluding-static-websites)**HTTPS on a site-wide basis. Enabling HTTPS encrypts the connection between a browser and the website, thereby securing sensitive data transmissions. Therefore it makes perfect sense for banks and high-profile technology firms to enable HTTPS on their dynamic websites because of the transaction of important and sensitive information.**

**We have also to realize that it is of equal importance to enable HTTPS across**[**static websites**](https://www.venafi.com/blog/https-should-be-implemented-everywhereincluding-static-websites)**, even if there aren’t any sensitive data transactions. A lot of corporations purchase an SSL certificate and they only configure the pages to be served over HTTPS that require a user to transmit personal information, such as login screens and checkout pages. That’s not a good way to operate.**

**Because of the abstract nature of internet connections, people think that a connection to a static website is secure over HTTP. However, the traffic travels through many points to get from your browser to a website. HTTP is insecure and allows anyone to manipulate traffic at any point between a laptop and a website. Attackers can intercept a lot of information by manipulating traffic on a static website protected only by HTTP. Some of it can be relatively harmless but other abuses are much more serious. But none of these abuses are possible if a site is protected by HTTPS. If there is any problem, web browsers like Chrome and Firefox display a message that warns visitors that they cannot verify the site’s TLS certificate.**

Benefits of Enabling SSL site wide

**So here is the first benefit for organizations.**[**Trust**](https://www.venafi.com/blog/https-should-be-implemented-everywhereincluding-static-websites)**. Encryption is like multifactor and security of the end user, but it also means that users can place greater trust in a website’s safety and authenticity. SSL/TLS is a solid way to endorse just how safe your platform is, adding a touch of professionalism to any site using it.**

**In addition, enabling HTTPS on a site-wide basis maintains compliance with all data privacy regulations, such as**[**GDPR**](https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679)**or**[**NIST SP 800-122**](https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-122.pdf)**. Especially, GDPR clearly states that organizations must be able to provide “*sufficient guarantees to implement appropriate technical and organizational measures*” to ensure that processing of personal data will comply with the GDPR and that data subjects’ rights are protected. It’s a win-win situation.**

**Another (hidden) benefit is Google’s SEO ranking. Some companies spend a ton of resources on SEO without realizing that simply enabling SSL can give their site a**[**ranking boost**](https://www.hostinger.com/tutorials/ssl-benefits)**on Google Search. In 2014, when the browsers were still incentivizing SSL instead of mandating it,**[**Google announced**](https://comodosslstore.com/blog/why-you-should-implement-always-on-ssl.html)**it was making HTTPS a signal in its ranking algorithm. Experts estimate that having SSL/TLS can give your website up to a 5% boost. Now let’s think about what happens to that ranking signal after everyone starts to migrate to HTTPS. It becomes a standard, and the boost functionally begins to flip, to change from a benefit for sites that have it to a penalty for sites that don’t. When everyone ranks 5% higher than you, you’re at a disadvantage.**

Enable HSTS

**In addition to enabling HTTPS on a site-wide basis, corporations should weigh the benefits of enabling**[**HSTS**](https://www.globalsign.com/en/blog/what-is-hsts-and-how-do-i-use-it/)**(HTTP Strict Transport Security), which is a web security policy mechanism that helps to protect websites against SSL stripping attacks and cookie hijacking. It allows web servers to declare that web browsers should interact with them using only secure HTTPS connections, and never via the insecure HTTP protocol.**

**When a web application issues HSTS Policy to user browsers, conformant user browsers will**[**automatically redirect**](https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet)**any insecure HTTP requests to HTTPS for the target website. In addition, when a man-in-the-middle attacker attempts to intercept traffic from a victim using an invalid certificate, HSTS does not allow the user to override the invalid certificate warning message. By having a HSTS policy installed, it will be nearly impossible for the attackers to intercept any information at all!**

Manage SSL Machine Identities

**In addition to enabling HTTPS on a site-wide basis and enforcing HSTS policy, corporations need to activate one last line of defense. They need to take proper safeguards to defend their SSL/TLS certificates against bad actors who could misuse the certificate. An important part of this process involves investing in a machine identity management solution such as**[**Venafi Trust Protection Platform**](https://www.venafi.com/platform/trust-protection-platform)**that allows organizations to**[**continuously monitor**](https://www.venafi.com/solutions/encryption-and-authorization)**their digital certificates for signs of abuse.**[**Venafi Trust Protection Platform**](https://www.venafi.com/platform/trust-protection-platform)**gives you the visibility you need to block a new breed of hackers who misuse keys and certificates to hide in your encrypted traffic. Plus, you’ll have what it takes to act quickly, when needed, to keep avoid compromise or disruption caused by expired certificates.**

**Performing & Preventing SSL Stripping: A Plain-English Primer**

20/10/2017

* [](https://blog.cloudflare.com/author/junade-ali/)

[Junade Ali](https://blog.cloudflare.com/author/junade-ali)

Over the past few days we learnt about a new attack that posed a serious weakness in the encryption protocol used to secure all modern Wi-Fi networks. The [KRACK Attack](https://www.krackattacks.com/) effectively allows interception of traffic on wireless networks secured by the WPA2 protocol. Whilst it is possible to backward patch implementations to mitigate this vulnerability, security updates are rarely installed universally.

Prior to this vulnerability, there were no shortage of wireless networks that were vulnerable to interception attacks. Some wireless networks continue to use a dated security protocol (called WEP) that is demonstrably "totally insecure" [[1]](https://blog.cloudflare.com/performing-preventing-ssl-stripping-a-plain-english-primer/#fn1); other wireless networks, such as those in coffee shops and airports, remain completely open and do not authenticate users. Once an attacker gains access to a network, they can act as a proxy to intercept connections over the network (using tactics known as ARP Cache Poisoning and DNS Hijacking). And yes, these interception tactics can easily be deployed against wired networks where someone gains access to an ethernet port.

With all this known, it is beyond doubt that it is simply not secure to blindly trust the medium that connects your users to the internet. HTTPS was created to allow HTTP traffic to be transmitted in encrypted form, however the authors of the KRACK Attack presented a [video demonstration](https://www.youtube.com/watch?v=Oh4WURZoR98) of how the encryption could be completely stripped away on a popular dating site (despite the website supporting HTTPS). This blog post presents a plain-english primer on how HTTPS protection can be stripped and mechanisms for mitigating this.



**HTTP over TLS**

The internet is built on a patchwork of standards, with components being refactored and rebuilt in new published standards. When one standard is found to be flawed, it is later patched or replaced by a new standard. As a standard is falsified and replaced by a better one, the internet as a whole becomes better.

The HTTP protocol was originally specified to communicate data in the clear over the internet. Prior to the official introduction of HTTP 1.0, the first documented version of HTTP was known as HTTP V0.9 and was published in 1991. Netscape were the first to recognise the need for greater security assurance over the internet and in mid-1994 HTTPS was implemented into the Netscape browser. In order to implement greater security assurance, a technology called SSL (Secure Socket Layer) was created.

SSL 1.0 was short lived (and not even officially standardised) due to a number of security concerns and shortcomings. This protocol was incrementally updated in SSL 2.0 and SSL 3.0; this was then iteratively superseded by the TLS (Transport Layer Security) standards.

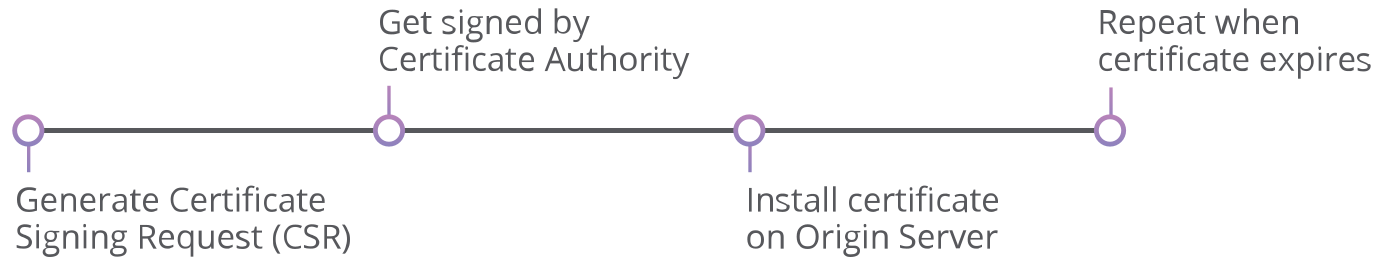
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Protocol** | SSL 1.0 | SSL 2.0 | SSL 3.0 | TLS 1.0 | TLS 1.1 | TLS 1.2 | TLS 1.3 |
| **Release** | N/A | 1995 | 1996 | 1999 | 2006 | 2008 | Draft |

Each of those versions come with different security constraints and support varies from browser-to-browser. Additionally, the encryption ciphers used can be configured somewhat independently of the overlying protocol. It is therefore vital to ensure any HTTPS-enabled web server is set-up to use a configuration that’s optimised to balance browser support against security. I won't go into detail on this in this post, however you can read about [SSL and TLS Deployment Best Practices](https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices) in documentation provided by SSL Labs.

At a high-level; the end result of HTTP over TLS, is that when a site is requested over https:// instead of http:// the connection is completed in an encrypted manner. This process provides a reasonable guarantee of both privacy and integrity; in other words, we don't just encrypt the messages we're sending, we make sure the message we receive isn't altered over the wire. When a secure connection is established, web browsers can indicate this to their users by lighting the browser bar green.



As SSL Certificates themselves are signed by Certificate Authorities, a degree of "domain validation" is carried out - the Certificate Authority makes sure they are only validating a certificate which is owned by someone who has the ability to make changes to the website. This provides a degree of assurance that certificates aren't being issued to attackers who can then seem legitimate when intercepting web traffic. In the event a certificate ends up in the wrong hands, a [Certificate Revocation List](https://en.wikipedia.org/wiki/Certificate_revocation_list) can be used to retract the certificate. Such lists are then automatically downloaded by modern Operating Systems to ensure that when an invalid certificate is served, it is marked as insecure in the browser. As there are a considerable number (>100) certificate authorities with the power to issue SSL certificates, it is possible to allowlist which Certificate Authorities should issue certificates for a given domain by [configuring CAA DNS records](https://support.cloudflare.com/hc/en-us/articles/115000310792-Configuring-CAA-Records-).



Nowadays this process can be simplified, for example; when your traffic is proxied through the Cloudflare network, we will dynamically manage renewing and signing your certificate for you (whilst using Cloudflare's [Origin CA](https://blog.cloudflare.com/cloudflare-ca-encryption-origin/) to generate a certificate to encrypt traffic back to the origin web server). Similarly, the EFF offer a tool called CertBot to make it relatively easy to install and [generate Let's Encrypt certificates from the command line](https://certbot.eff.org/).

When using HTTPS, it is important that the entire content of the website is then loaded over HTTPS - not just the login pages. It used to be common practice for websites to initially present the login page over a secure encrypted connection, then when the user was logged in, they would degrade the connection back to HTTP. Once logged into a website, a session cookie is stored on the local browser to allow the website to ensure the user is logged in.

In 2010, Eric Butler demonstrated how insecure this was by building a simple interception tool called FireSheep. By Eavesdropping wireless connections, FireSheep would capture the login session for common websites. Whilst the attacker would not necessarily be able to capture the password of the website, they would be able to capture the login session and perform behaviours on websites as if they were login. They would also be able to intercept traffic as the user was logged in.

When connecting to a website using SSL, the first request should usually redirect the user to a secure version of the website. For example; when you first visit http://www.cloudflare.com/ a HTTP 301 redirect is used to send you to the HTTPS version of the site, https://www.cloudflare.com/.

This raises an important question; if someone is able to intercept the unencrypted request to the HTTP version of the site, couldn't they then strip away the encryption and serve the site back to the user without encryption? This was a question explored by Moxie Marlinspike, which later led to the creation of HSTS.

**HTTP Strict Transport Security (HSTS)**

In 2009 at Blackhat DC, Moxie Marlinspike presented a tool known as SSLStrip. This tool would intercept HTTP traffic and whenever it spotted redirects or links to sites using HTTPS, it would transparently strip them away.

Instead of the victim connecting directly to a website; the victim would connect to the attacker, and the attacker would initiate the connection back to the website. This attack is known as an on-path attack.

The magic of SSLStrip was that whenever it would spot a link to a HTTPS webpage on an unencrypted HTTP connection, it would replace the HTTPS with a HTTP and sit in the middle to intercept the connection. The interceptor would make the encrypted connection to back to the web server in HTTPS, and serve the traffic back to the site visitor unencrypted (logging any interesting passwords or credit card information in the process).

In response, a protocol called HTTP Strict Transport Security (HSTS) was created in 2012 and specified in [RFC 6797](https://tools.ietf.org/html/rfc6797). The protocol works by the server responding with a special header called Strict-Transport-Security which contains a response telling the client that whenever they reconnect to the site, they must use HTTPS. This response contains a "max-age" field which defines how long this rule should last for since it was last seen.

Whilst this provided an improvement in preventing interception attacks, it wasn't perfect and there remain a number of shortcomings.

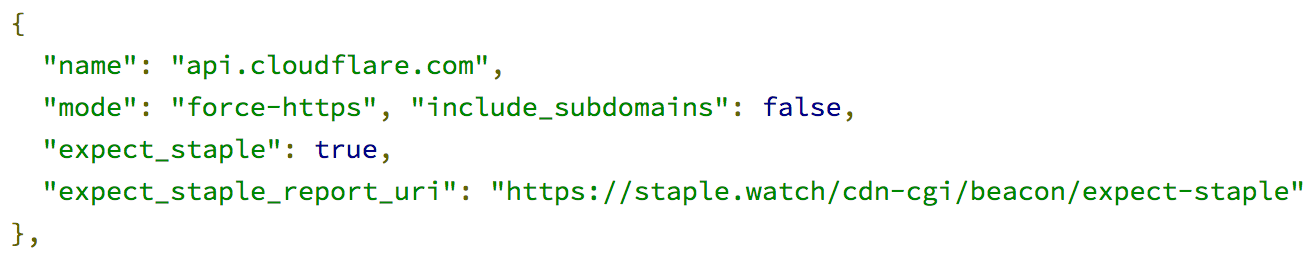
**HSTS Preloading**

One of the shortcomings of HSTS is the fact that it requires a previous connection to know to always connect securely to a particular site. When the visitor first connects to the website, they won't have received the HSTS rule that tells them to always use HTTPS. Only on subsequent connections will the visitor's browser be aware of the HSTS rule that requires them to connect over HTTPS.

Other mechanisms of attacking HSTS have been explored; for example by hijacking the protocol used to the sync a computer's time (NTP), it can be possible to set a computers date and time to one in the future. This date and time can be set to a value when the HSTS rule has expired and thereby bypassing HSTS[[2]](https://blog.cloudflare.com/performing-preventing-ssl-stripping-a-plain-english-primer/#fn2).

HSTS Preload Lists are one potential solution to help with these issues, they effectively work by hardcoding a list of websites that need to be connected to using HTTPS-only. Sites with HSTS enabled can be submitted to the Chrome HSTS Preload List at [hstspreload.org](https://hstspreload.org/); which is also used as the basis of the preload lists used in other browsers.

Inside the source code of Google Chrome, there is a file which contains a hardcoded file listing the HSTS properties for all domains in the Preload List. Each entry is formatted in JSON and looks something like this:



Even with preload, things still aren't perfect. Suppose someone is reading a blog about books and on that blog there is a link to purchase a book from an online retailer. Despite the fact the online retailer enforces HTTPS using HSTS it is possible to conduct an on-path attack, providing the blog linking to the online retailer does not use HTTPS.

**More Still to be Done**

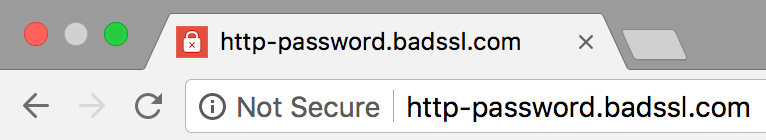
Leonardo Nve revived SSLStrip in a new version called [SSLStrip+](https://github.com/LeonardoNve/sslstrip2" \t "_blank), with the ability to avoid HSTS. When a site is connected to over an unencrypted HTTP connection, SSLStrip+ will look for links to HTTPS sites. When a link is found to a HTTPS site, it is rewritten to HTTP and critically the domain is rewritten to an artificial domain which is not on the HSTS Preload list.

For example; suppose a site contains a link to https://example.com/, the HSTS could be stripped by rewriting the URLs to http://example.org/; with attacker sitting in the middle, receiving traffic from http://example.org/ and proxying it to https://example.com/.

Such an attack can also be performed against redirects; suppose http://example.net/ is loaded over HTTP but then redirects to https://example.com/ which is loaded over HTTPS. At the point the redirect is carried out, the legitimate HSTS-protected site can be redirected to a phoney domain which the attacker uses to serve traffic over HTTP and intercept traffic.

As more and more of the internet moves to HTTPS, the surface area of this attack will get smaller as there is less unencrypted HTTP traffic to intercept.

In the latest newly-released version of Google Chrome (version 62), websites which serve input forms (such as credit card forms and password fields) on insecure connections are flagged as "Not Secure" to the user, instead of a neutral message. When in Incognito (private browsing) mode, Chrome will flag any website as insecure if it does not use HTTPS.



This change helps make it clearer to users when HTTPS has been stripped away from a webpage as they try to login. Additionally, in making this change, it hoped that more websites will adopt HTTPS - thereby improving the security of the internet as a whole.

**Final Remarks**

This blog post has discussed mechanisms of stripping HTTPS away from websites and in particular how HSTS can affect this. It is worth noting that there are other potential attack vectors within various HTTPS specifications and in certain ciphers; this blog post haven't gone into them.

Despite HTTPS offering a mechanism for encrypting web traffic, it is important to implement technologies such as HTTP Strict Transport Security to ensure it is enforced, and preferably submit your site to HSTS Preload lists. As more websites do this, the security of the internet overall is improved.

To learn how you can implement HTTPS and HSTS in practice, I'd highly recommend Troy Hunt's blog post: [The 6-Step "Happy Path" to HTTPS](https://www.troyhunt.com/the-6-step-happy-path-to-https/). His blog post goes into how you can enable strong HTTPS in practice, and additionally touches on a technology that I didn't mention here known as CSP (Content Security Policy). CSP allows you to automatically upgrade or block HTTP requests when loading pages over HTTPS, as this poses another attack vector.

1. Stubblefield, A., Ioannidis, J. and Rubin, A.D., 2002, February. [Using the Fluhrer, Mantin, and Shamir Attack to Break WEP](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.2068&rep=rep1&type=pdf). In NDSS. [↩︎](https://blog.cloudflare.com/performing-preventing-ssl-stripping-a-plain-english-primer/#fnref1)
2. Selvi, J., 2014. [Bypassing HTTP strict transport security](https://www.blackhat.com/docs/eu-14/materials/eu-14-Selvi-Bypassing-HTTP-Strict-Transport-Security-wp.pdf). Black Hat Europe.

Understand the con trick that is SSL Stripping or HTTP Downgrade

SSL Stripping or an SSL Downgrade Attack is an attack used to circumvent the security enforced by SSL certificates on HTTPS-enabled websites. In other words, SSL stripping is a technique that downgrades your connection from secure HTTPS to insecure HTTP and exposes you to eavesdropping and data manipulation. Think of it as wiretapping, but a bit more technical. When your web browser comes into contact with a web server, the first contact is made using ordinary TCP and then the user is redirected to TLS/SSL. Hackers take advantage of this small window using SSL strip or SSL downgrade attacks.

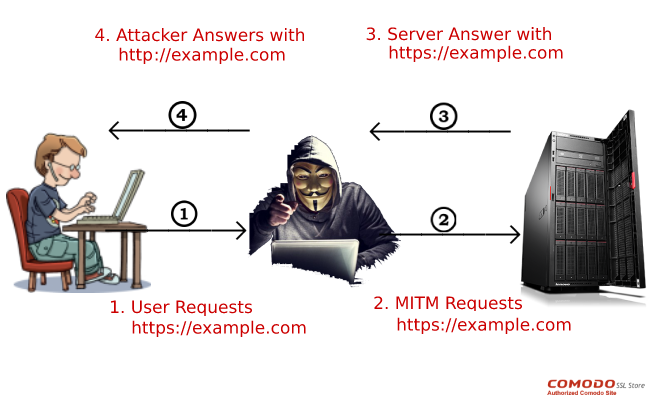
Moxie Marlinspike, American computer security researcher and cypher punk first demonstrated how one can bypass HTTPS security. It is a very dangerous technique that could prompt huge implications from a user’s point of view. Let’s see how it works.

SSL Stripping: A con trick

Fundamentally, SSL strip attacks function the same way as a con trick does. In SSL Stripping, a user is made to believe that the connection is secure and the data he/she sends is encrypted. But in reality, the connection is insecure and data is sent in plain text, stripping off the encryption. That’s why the name SSL ‘strip’.

To execute an SSL strip attack, there must be three entities – victim’s system, secure web server, and attacker’s system. Had there been no attacker in between, the communication would only happen between victim’s system and the web server of website. An SSL stripping attack is employed to kill this secure communication without making the victim realize. The Attacker/hacker uses a proxy server and his/her coding skills to pull off this trick.

Let’s see how it’s done with an example.



Suppose Nick wants to send some money by accessing a secure, HTTPS-enabled website. But Bob, who is a hacker, wants to intercept this communication and see Nick’s credentials. To accomplish this, he establishes a connection with the victim, thereby cutting Nick’s communication with the secure server. When Nick requests to visit the banking site on his browser, Bob gets it and forwards it to the server of the banking site. A thing to note here is that all the communication taking place between Bob and the website is SSL protected.

The web server responds to Bob’s (originally Nick’s) request in form of an HTTPS URL. Now using his precarious and delicate coding skills, Bob downgrades this secure HTTPS URL to insecure HTTP URL and forwards it to Nick. The beauty (or fatality!) of the attack is such that poor Nick can’t have any clue what’s happening in the background. As the connection is over stripped HTTP protocol, everything Nick sends would be in plain text only. His passwords, banking details, credit card details…everything he sends to this URL can be seen by Bob. As the connection is downgraded from HTTPS to HTTP, stripping attacks are also known as HTTP DOWNGRADE attacks.

Primarily, there are three ways through which SSL stripping attacks can be executed. They are:

1. Using Proxy Server
2. ARP Spoofing
3. Using Hotspot

How to protect my website against SSL Strip attacks?

[SSL or HTTPS](https://comodosslstore.com/blog/what-is-ssl-and-how-do-ssl-certificates-work.html) is an idea, a wonderful idea indeed. Just like any other idea out there, there are loopholes in it. And such loopholes must be addressed. Here are the three things you can do to make your HTTPS website even more secure:

* Enable [HTTPS on pages](https://comodosslstore.com/blog/why-you-should-implement-always-on-ssl.html) of your website
* Implement HSTS policy – A Strict policy under which browser won’t open a page unless the site has HTTPS.

SSL STRIPPING

Another form of man-in-the-middle attack happens when a hacker manages to stage an SSL stripping scheme against the victim. As we mentioned previously, hackers can’t break into legitimate HTTPS traffic between a client and a server even if they manage to intercept and relay the communications.

In the case of SSL stripping, the attackers downgrade the communications between the client and server into an unencrypted format to be able to stage a MitM attack.

When a victim wants to connect to a server, the attacker intercepts the request and creates an independent, legitimate connection to the server through HTTPS protocol. When attackers receive the server’s response, they relay it to the victim in an unencrypted format, posing as the server. Thinking they’re communicating with the legitimate party, the victim will continue to send information to the attacker, who will then relay it to the server in HTTPS.

Wary users will notice that they’ve been targeted by an SSL stripping attack if they look in their browser’s address bar and see that they’re connected through the unencrypted HTTP protocol. You can also install HTTPS Everywhere, a browser extension that enforces HTTPS communication wherever possible. HTTPS Everywhere will prevent an uninvited party from downgrading your communications to HTTP.

Another measure to protect against SSL stripping is to make sure your local network is secure and unauthorized parties don’t have access to it. SSL hijacking requires access to your local network. At the corporate level, setting up strong firewalls will also prevent outside parties from gaining access to your local network and moving laterally to stage MitM attacks.

Frequently Asked Questions

HOW CAN SSL STRIPPING BE IMPLEMENTED?

The most common way of creating an SSL Striping man in the middle attack are:

1) Manually set the proxy of the browser to route all traffic  
2) Address Resolution Protocol (ARP) Poisoning  
3) Create a Hotspot and allow the victims to connect to it

WHAT CAN BE DONE TO PROTECT AGAINST SSL STRIPPING?

1) Enable SSL site-wide – HTTPS only  
2) Enable HSTS -HTTP Strict Transport Security  
3) Enable Cert Pinning  
4) Enable secure cookies,: ensure that all cookies are served with the secure attribute so that your user’s browsers will only send those cookies back over SSL-protected connections and never disclose them over any non-SSL (HTTP) link.  
5) Disable non-SSL access (HTTP) or redirect users to the SSL version of the website.

What is SSL stripping?

In a nutshell, SSL stripping downgrades an HTTPS (Hyper Text Transfer Protocol Secure) connection to one that is HTTP (the now out-of-date, less secure protocol).

Via a proxy, a hacker - the “man-in-the-middle” of a connection - intercepts all user requests made to a website’s server. Rather than connecting to a secure site, users are rerouted to the unsecure proxy server. Most users won’t even know that the redirect has occurred because they will end up on a page that looks virtually the same as the one they were searching for.

In this way, SSL stripping is more sophisticated than phishing. Phishing requires a user to log in to a fake page which allows the attacker to collect data like user name and password.

SSL stripping directs a user to an HTTP proxy that is related to a legitimate HTTPS-encrypted site. The attacker can collect logins and passwords via the HTTP connection without the victim noticing anything. They won't see an error or warning message in their browser alerting them to the fact they have been rerouted.

Having removed the Secure Sockets Layer (SSL) that protects a user’s confidential information, a hacker can eavesdrop and manipulate data at will.

How does SSL stripping work?

Users don’t typically come to SSL-secured sites by typing in a full URL or using a bookmarked *https://url*. Many arrive via a redirect (like the 302 redirect) or an HTTP site which provides a link to the secured site. Users are redirected or click on that link and it takes them where they want to go.

For example, you type into your browser *www .example.com*. The browser connects you to the hacker’s machine using HTTP and forwards your request to the server over HTTPS.

The hacker downgrades the connection from HTTPS to HTTP and sends it back to your browser. You will see *http://www .example.com*. The SSL has been “stripped”, your data is compromised, and the site’s sever continues to think that a secure connection has been made.

How can you prevent this form of attack?

SSL certificate

An SSL certificate alone won't protect you but you need to encrypt all of your connections with an HTTPS-configured [SSL certificate](https://www.eurodns.com/ssl-certificate). And you need to encrypt all elements of your site, not just the login page. Pictures, links - everything.

When you purchase an SSL certificate, you can (for an additional fee) add a Wildcard option which allows you to use your SSL on an unlimited number of subdomains and servers for greater security.

An Organisation Validation (OV) or Extended Validation (EV) SSL certificate will further improve your site’s level of security and confirm its authenticity. An EV SSL certificate shows your company’s name in a green URL bar as proof that your site is legitimate. [Compare SSL certificates](https://www.eurodns.com/ssl-certificates-comparison) to find the one that best suits your needs.

HSTS Preload List

Once you have your SSL certificate, add your domain name to the HSTS preload list, a global list that is used by Chrome, Firefox, and other search engines.

In a previous post, we covered how to add your site to the [HSTS preload list](https://www.eurodns.com/blog/hsts-preload-list-site-security) and why you should do it. But to recap:

* HSTS stands for HTTPS Strict Transport Security.
* The HSTS preload list contains a list of hostnames for which browsers automatically enforce HTTPS-connections
* Once a browser receives a site’s HSTS, it will be included on the preload list
* Inclusion on the list prevents future insecure HTTP connections from being made

The list includes domains, subdomains, and even entire TLDs. Some TLDs like .APP are already HSTS preloaded, meaning all websites under a [.APP domain](https://www.eurodns.com/domain-extensions/app-domain-registration) are HTTPS-encrypted by default. But as long as you are using an SSL certificate on your site, you can submit any domain to the list.

Educate your users

Last, not least, keep your users informed about a few basic precautions they can take to avoid falling victim to SSL stripping.

**HTTPS Everywhere:**Encourage users to download the [HTTPS Everywhere](https://www.eff.org/https-everywhere) browser extension which will force their browsers to only send information over HTTPS websites.

**Virtual Private Networks (VPN):**A VPN provides users with a layer of secure encryption no matter what site they are on. Even if a site is downgraded to HTTP, data will remain encrypted.

**Wi-Fi:**Avoid using public Wi-Fi networks, especially when sending sensitive data (like credit card information when making a purchase)

**HTTPS:**If those FIVE letters - HTTPS - aren't in front of the URL, don’t click on it.

**Links:**Don’t click on malicious-looking links or emails.

The Secure Sockets Layer (SSL) transfer protocol and its successor, Transport Layer Security (TLS), are amongst the most important building blocks for a secure web presence. They **encrypt the information** exchanged between browser and server via HTTP, before the information is sent – even when switching between an encrypted HTTPS and an unprotected page. This does not just prevent standard data transfer in plaintext, but also prevents a cookie set under SSL from being sent with an unencrypted connection. These useful SSL and TLS certificates also guarantee the **authenticity of the server hostname** to the requesting client. The TLS protocol provides security in numerous ways, which makes it indispensable whenever sensitive information is being transmitted.

TLS is generally one of the safest protocols, and has so far dealt well with attack attempts. However, under certain circumstances, special tools (such as the **sslstrip** programmed for demonstration purposes) are able to gain access to the data transfer before the encryption has begun. This type of unauthorized, third-party access is referred to as **SSL stripping.**

**SSL certificates from IONOS**

Protect your domain and gain visitors' trust with an SSL-encrypted website!

*Easy activation*

*Proven safety*

*24/7 assistance*

Top of Form

Continue

Bottom of Form

Contents

1. [What is SSL stripping?](https://www.ionos.com/digitalguide/hosting/technical-matters/ssl-stripping-how-to-protect-your-web-project/#c85522)
2. [How is an SSL-strip implemented?](https://www.ionos.com/digitalguide/hosting/technical-matters/ssl-stripping-how-to-protect-your-web-project/#c85525)
3. [Can you recognize SSL stripping as a user?](https://www.ionos.com/digitalguide/hosting/technical-matters/ssl-stripping-how-to-protect-your-web-project/#c85526)
4. [What can you do to safeguard against it?](https://www.ionos.com/digitalguide/hosting/technical-matters/ssl-stripping-how-to-protect-your-web-project/#c85527)
5. [How can HSTS help against SSL stripping?](https://www.ionos.com/digitalguide/hosting/technical-matters/ssl-stripping-how-to-protect-your-web-project/#c85528)

What is SSL stripping?

As early as 2002, the developer Moxie Marlinspike used**sslsniff** to program a tool that could eliminate SSL encryption. The proxy software made it possible to infiltrate SSL data streams and to **exchange the server certificate with any of its own certificates.** Marlinspike wanted to use the application to show the weaknesses of Internet Explorer, which was vulnerable to [man-in-the-middle attacks at the time of publication](https://www.ionos.com/digitalguide/server/security/man-in-the-middle-attack-an-overview-of-attack-patterns/).

Microsoft was able to secure the vulnerability, and other popular clients are widely protected from this kind of attack, provided they have the most **up to date version** and the correct configuration.

Marlinspike presented the sslstrip program in 2009 as part of the **security conference Black Hat DC.** Like his previous tool, sslstrip is a proxy that is positioned between client and server and tries to bypass certification on browser pages. To that end, the tool searches the web pages delivered by web servers for embedded links and referrals, which he redirects an SSL-protected log-in page, such as the following link:

<a href="https://example.com/login.php">

If the proxy finds this kind of link, it modifies it to an equivalent HTTP link. The user sends the registration through his browser, rather than the supposedly encrypted ordinary data in the plaintext. A potential hacker can **easily read through sslstrip as an intermediate station** and reach confidential information. Since the SSL stripping does not create an invalid connection, no warning messages are displayed. Generally, the user does not receive a warning that the information he is transmitting is unencrypted.

How is an SSL-strip implemented?

Regardless of whether sslstrip or another similarly programmed application is used, the first thing an attacker does is switch the **proxy between the browser and the web server**. The software also has the ability to encode modified URLs by SSL stripping, but only if it can intercept or forward data flows. The following three methods are common in the implementation:

1. **Incorrect entry of the proxy in the browser options**: When your system is targeted, the goal is often just the browser, rather than the whole computer. Malware then ensures that an external proxy server is automatically entered in the settings without the user being aware.
2. **ARP or NDP Spoofing**: Within a subnet, an attacker can fall back on [ARP spoofing](https://www.ionos.com/digitalguide/server/security/arp-spoofing-attacks-from-the-internal-network/) (IPv4) or NDP-Spoofing (IPv6) to bring their proxy into play. The purpose of both protocols is to resolve the IP addresses into corresponding hardware addresses (also known as MAC addresses). Using manipulated messages from these protocols, the attacker can replace requested hardware addresses with his own system address, and then intercept the transmitted data packets.
3. **Providing your own hotspot**: The third option is that the device that the server proxy runs on can also act as a router. As a standard gateway, including a DHCP server, it can assign IP addresses to users, and read and forward packets that are sent beyond the boundaries of the subnet. This provides the perfect basis for SSL stripping.

After he has positioned the proxy, the attacker does not have to do much more for the SSL strip: he runs the tool, which sends out altered links when needed. If successful, is also sends out unencrypted information such **as bank- or user log-in data.**

Can you recognize SSL stripping as a user?

Servers and browsers have no way of detecting an SSL strip. Both applications assume that they are communicating with the real contacted partner, which is why they do not doubt the integrity of the transmitted data. The situation is quite similar for users, because **at first glance**, visiting the website seems to go as normal.  SSL stripping can only be seen in a few exceptional cases, through technical or design details. Unless a strikingly faulty layout is presented, or considerable delays occur when loading the page, there are very few signs that **SSL encryption** is missing.

However, for quite a while now browser address lines have been providing hints in different ways: In order to identify websites with secure connections, the address bar was **completely green** in older versions of Microsoft Internet Explorer. Other browsers just highlighted the company’s previous name, until this type of identification – commonplace with the first web-enabled mobile devices – was replaced by today’s common symbols, such as the **typical security lock**. However, these visual hints do not always guarantee that the site being visited has not been compromised by tools like sslstrip. Since an attacker **controls the whole data transfer**, he is able to deliver a similar symbol to the favicon to perfect his deception.

What can you do to safeguard against it?

The difficulty of detecting malicious pages makes SSL stripping attacks so dangerous to users: the encryption certificates which should be used by every careful website operator signify **security and trustworthiness**, and they do not give the visitors concerns about disclosing confidential information. In principle, SSL provides the necessary protection, since the ability to read and intercept data packets does not result from a security gap in the protocol, but from the fact that the encryption itself is prevented. To protect against SSL strips, each user should force the construction of encrypted HTTPS connections. Ways of doing this can be seen in the following examples:

1. **Enter the URL manually**: A cumbersome, but effective measure is entering the HTTPS URL into the browser
2. **Browser Extension**: There are several browser extensions that can help you access encrypted versions, if they exist. For example, the HTTPS Everywhere extension uses domain and rule lists **to handle any page calls through HTTPS connections**. Versions for Firefox, Android, Chrome, and Opera can be found on the Electronic Frontier Foundation website, which develops and supports the [expansion](https://www.eff.org/https-everywhere) together with the Tor project
3. **Save secure URL’s as bookmarks**: If you usually use an SSL protected web service (online banking, cloud storage etc.), you can save the HTTPS version as a bookmark and always access it that way. The condition for this is that you are in a secure network when you create the bookmark, otherwise you may add an already-manipulated URL to the Favorites list

You can also combat SSL stripping as the operator of a web project. For example, a basic step can be to enable encryption for all current pages and **force incoming HTTP connections to secure their pages**. The same thing applies to cookies: If you do not want to use practical data records for web analysis, make sure they are not sent back through unsecured HTTP connections. To do this, just record the **cookies** with the **‘secure’ attribute**, ensuring that your server is only receiving feedback via HTTPS. A further safety measure is the IETF standard HSTS, which is described in more detail in the following section.

How can HSTS help against SSL stripping?

Three years after Marlinspike pointed out the vulnerability of SSL certified websites using his sslstrip software, the IETF (Internet Engineering Task Force) specified the security mechanism [HSTS (http Strict Transport Security)](https://www.ionos.com/digitalguide/websites/web-development/hsts-reliably-secure-your-https-connections/) in [RFC 6797](https://tools.ietf.org/html/rfc6797). This allows web servers to alert connection-building clients that they are accessing the website **exclusively through a HTTPS connection** for a specific amount of time. To this end, the server uses the ‘Strict Transport Security’ field in an ordinary HTTP response header, plus the ‘max age’ directive, which defines the validity period of the statement in seconds. To secure a domain and make it reachable only by encrypted connection for one year, the web server’s http response must contain something like the following line:

Strict-Transport-Security: max-age=31536000

The ‘includeSubDomains’ parameter can be used to extend the command to all subdomains of the website, so that the use of SSL/TLS is enforced. If a browser receives a message from a contracted web server with a ‘Strict Transport Security’ statement, **all unencrypted requests are automatically converted to encrypted** in future connections to the relevant domains. If the connection is not secure, an error message is displayed and the requested page is not called.

HSTS is a permanent solution to protect a website and potential visitors from SSL strips and similar attacks. However, as previously mentioned, there is always a **very first connection structure** which can be manipulated before the safety mechanism can intervene. To counter this problem, Google has introduced a **preload list** for its Chrome browser, containing web projects only available via HTTPS. Other browser vendors have adopted the principle and implemented HSTS preload lists based on the Chrome list. To add your website to the list, you can submit a request to the project page set up by [Google.](https://hstspreload.org/)