Always-On SSL

"Always-On SSL" is more than just a configuration setting. **Always-On SSL** is an approach to securing end user security for the duration of each user's visit to your website—from beginning to end. The goal is to extend SSL hardening beyond your own server and to your user's browser so that you can both be assured of the security that SSL/ TLS is supposed to provide.

This paper explains how website operators can provide visitors with a positive user experience while providing better security. The configurations and settings recommended here take a holistic approach known as "**Always-On SSL**." Recommended measures include: use of secure cookies and HSTS headers and elimination of mixed (http-https) content; pre-launch review of code and configurations to ensure that no holes or backdoors are left open; implementing best protocol and ciphersuite combinations; and using Extended Validation certificates to provide your visitors with a positive and consistent experience.

Contents

The Current Threat Landscape.......... 1

What is Always-On SSL?.......... 2

What risks are we trying to address?.......... 3

  Firesheep: The Problem with Insecure Cookies.......... 3

  SSL Strip: ARP-spoofing to Launch Man-in-the-Middle Attacks.......... 3

Good Design.......... 5

  Remove Mixed Content.......... 5

  Implement HTTP Strict Transport Security (HSTS).......... 6

Use a High-Quality CA Service Provider and Extended Validation SSL.......... 6

Fix Server Protocol and Ciphersuite Settings.......... 5

SSL Tools.......... 7

Conclusion.......... 8

References.......... 8

The Current Threat Landscape

Many website operators believe that if they have installed an SSL certificate for use with Secure Socket Layer/Transport Layer Security (SSL/TLS), then they have done all that is necessary to protect communications with their web site. While SSL/TLS has become well-known over the past two decades for its ability to protect credit card transactions, it is now clear, but somewhat less appreciated, that online properties need to implement better security with SSL/TLS to prevent attempts to circumvent the protections of SSL/TLS. However, as discussed in this paper, installing an SSL/TLS certificate on your site is just one part of your web site's security.   Past practices are no longer good enough, and a more secure approach is needed—that approach is **Always-On SSL**.

How often do you recall seeing websites where the address being advertised by the provider begins "HTTPS://" ?   Not many, and until more recently, hardly anyone knew the difference between a web address beginning with HTTP and one beginning HTTPS. Some browsers no longer even display either of those sets of letters, which is not that relevant to the main point of this discussion—that SSL should be implemented everywhere. In fact, now benefitting from 20-20 hindsight, we could even say that the World Wide Web should have been built more secure from the start with "HTTPS://" (meaning "secure") rather than with a default of simply "http://".   Nevertheless, over time we have seen more and more attacks attempt to disrupt the security of end-to-end encryption.   Today, we go online and into an environment of attackers equipped with a variety of tools-- phishing, malware, toolkits, etc., designed to intercept and change not just normal, unencrypted traffic, but traffic that was supposed to be encrypted. This happens, for instance, when an attacker launches what is known as a "man-in-the-middle attack".

We now use online resources for lots of everyday activities such as shopping, banking, and communicating with others.   This is an information-rich stream that attackers want to break into in order to pull what they can for malicious use, so we need to escalate our security behaviors and begin to implement **Always-On SSL**. Many high-profile websites have come to the realization that if they don't do something, these attackers will drive away business. And so they are working diligently to improve the security of their websites and build upon their reputation of trust with their user communities. **Always-On SSL** is one key security approach among many others that these owners have identified as essential to the continued protection of their good will and intellectual property.

What is Always-On SSL?

Always-On SSL is an approach to web design and implementation that requires the use of HTTPS for all webpages-- beginning with the home page. Web giants such as Google, Facebook, PayPal, Twitter, and Yahoo have embraced **Always-On SSL**. The reasons are clear--they care about the security of visitors to their sites, they value the trust of Internet users who have placed value in their brands, and they have found that HTTPS with **Always-On SSL** is not that difficult to implement.

Indeed, these key web sites are leading the way for others who have already implemented SSL, but have not yet tried **Always-On SSL**, let alone those who have implemented no security to protect visitors to their sites. Try opening a regular http connection with Twitter, Facebook, Twitter, Yahoo, PayPal, Dropbox, Tumblr, Wells Fargo, Chase, American Express, Standard Chartered, Credit Suisse, or any of several other websites with **Always-On SSL**, and you will see how it forces SSL even when you try to use "http://" in the address bar.

What risks are we trying to address?

Some of the vulnerabilities that can be exploited if you do not implement **Always-On SSL** include session hijacking, side-jacking, and similarly, man-in-the-middle attacks. Just two examples are Firesheep and SSL Strip, discussed in this section. Both attacks involve spoofing local traffic. Firesheep is a program that makes it easy to intercept unsecured authentication cookies and use them to side-jack secure sessions. SSL Strip is a man-in-the-middle attack facilitated by downgrading https to plaintext http. SSL Strip allows an attacker to capture and read all web traffic (including session logins) in plain text.

Firesheep: The Problem with Insecure Cookies

Cookies are often used as an authentication mechanism for client-server communications. A session cookie can be set with a flag that indicates it is supposed to be protected and kept "secure". One of the key vulnerabilities discovered in recent years is the insecure cookie. Theft of cookies that are not marked "secure" can lead to impersonation and the theft of credit card numbers and other personal data. This vulnerability is exploited when a website uses SSL only to establish the session cookie, and then reverts back to HTTP/port 80 for the remainder of the communication. There are attacks that can be launched against this insecure configuration. Without end-to-end encryption, it is very easy for a Man in the Middle to intercept the session cookie and use it to impersonate the site visitor. This leads to session hijacking and can also mislead the user with a false sense of security. Thus, by failing to mark authentication-related cookies as secure or failing to protect them when initiating communication, they are available for an attacker to grab.

For example, a few years ago some security researchers created a special tool to attack this vulnerability. They named it "Firesheep." (Firesheep is a play on words because it can be loaded as an add-on to the Firefox browser.) With Firesheep, the attacker simply launches the program, with no programming skills whatsoever, and hijacks user sessions over open Wi-Fi connections. This might happen, for instance, at your local cyber café. The program simply waits until someone tries to connect to a popular website stored in Firesheep's database, and then it uses that website's session protocols to grab the user's session. It allows an attacker to take over and impersonate the user on his or her active account, just as if the attacker were the authorized user—even allowing the attacker to change the user's complete profile, including passwords, in some instances. In response to Firesheep, the Electronic Frontier Foundation created its own Firefox extension, "HTTPS-everywhere." The name itself suggests the need for **Always-On SSL**.

SSL Strip: ARP-spoofing to Launch Man-in-the-Middle Attacks

Another security researcher, Moxie Marlinspike, created a program called "SSL Strip" to demonstrate how an attacker might trick users by "stripping" SSL from the user's session. Here is how SSL Strip works. The attacker prepares his machine to act as a router (Man in the Middle) on the local network: promiscuously monitor all traffic; change destination ports and forward packets; log all traffic to and from the victim's machine, etc. Then he launches the attack by changing routing tables and arp-spoofing the victim's machine by telling it that his machine is the local router/gateway. For instance, if the victim's IP address is 192.168.1.10 and the attacker's MAC address is 00-AB-CD-EF-12-34, he broadcasts a message using a command that essentially says, "tell 192.168.1.10 that 192.168.1.1 (the IP address for the gateway router) is at 00-AB-CD-EF-12-34." Once this deception is in place, SSL Strip running on the attacker's machine takes control and routes the victim's communications as if it were the network router. If the communications were supposed to occur via https or SSL/TLS, they are intercepted, and the victim's machine is forced to communicate via http on port 80. The browser's address bar suddenly switches over to http, usually without notice, and packets to and from the victim's machine are unencrypted and also logged to a port on the attacker's computer. These types of attacks work because most users do not know the difference between http and https and do not notice when an address bar changes from https to http. Many users will just type in the name of their bank and add ".com", and most websites will take the user's request and respond by telling the client machine to re-request the connection over port 443. SSL Strip forwards external packets to their destination and communicates via SSL with external websites. When responses are returned, the attacker's machine acts as a proxy, providing content to the victim in clear text via normal http. It can also be used to redirect users to look-alike HTTPS links with faked content.

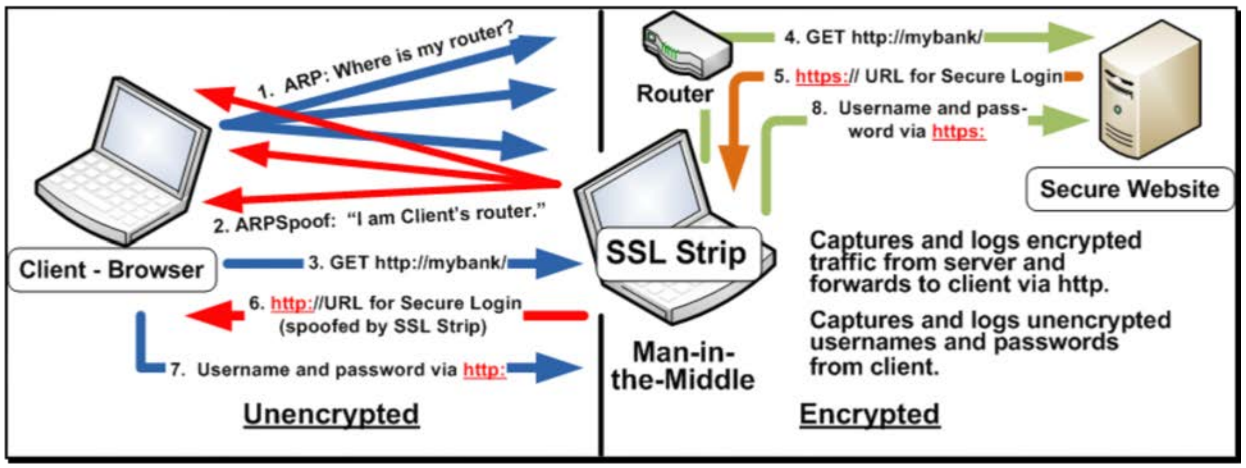

1. ARP: Where is my router?
Router
4. GET http://mybank/
5. https:// URL for Secure Login
8. Username and pass-word via https:
Secure Website
2. ARPSpoof: "I am Client's router."
Client - Browser
3. GET http://mybank/
SSL Strip
Captures and logs encrypted traffic from server and forwards to client via http.
6. http://URL for Secure Login (spoofed by SSL Strip)
Captures and logs unencrypted usernames and passwords from client.
7. Username and password via http:
Man-in-the-Middle
Unencrypted
Encrypted

Illustration of ARP Spoofing and SSL Strip

You might ask, how would **Always-On SSL** prevent this attack because a website cannot control whether a local proxy converts SSL traffic into clear text and vice versa? The answer is that this exploit takes advantage of the fact that the first step in the SSL handshake is the weakest one -- most client systems must take a big step from HTTP over to HTTPS, and SSL Strip exploits this transition by jumping in even before the SSL connection is established. Recalling that **Always-On SSL** is a collection of several, complementary security-related practices and measures, it includes HTTP Strict Transport Security (HSTS), which is discussed below. HSTS allows websites to pre-designate that all communications must be over HTTPS. [RFC 6797]

Action Items:

- Set the "Secure" Flag on your Session Cookies
- Implement HSTS (see discussion of this below)
- Configure your port 80 HTTP servers to always redirect all traffic to port 443 HTTPS

Good Design:

Remove Mixed Content

Always-On SSL is a holistic approach to website communication security. It means that the website owner must work toward an environment of continuous protection for site visitors. Visitors require a uniformly secure and trusted zone during their complete visit to a secure site. Of course, this means that the website owner must scan its code for mixed content—instances where “HTTP” (without the “s”) occurs—and replace those HTTP references with HTTPS pointers. It also means that all routing systems should specify port 443. Instances of port 80 (http) should be replaced with port 443, the port number for https. In other words, a site or domain implementing **Always-On SSL** should be configured to switch over to https when a user types “http” (or otherwise fails to specify SSL/TLS security). This is because without specifying “https” the browser will most often default to http on port 80 rather than try and communicate over port 443 via https. HTTP Strict Transport Security (HSTS), discussed below, should also be implemented in order to keep SSL “Always On.”

Similarly, remove HTTP-based calls to external content. While the hyperlinked nature of Internet resources is great, exercise extreme care in grabbing widgets, frames, JavaScript, and other resources that have not been checked for their security and compliance with your own secure-design policies. One of the first steps in protecting against malicious code and man-in-the-middle attacks is to implement **Always-On SSL** with your content providers—after all, you are allowing them into the same secure zone that you’ve established for your customers.

One may ask, won’t this cost a lot, slow us down, and after all of this is done, won’t SSL encryption slow things down too much? The answer is that while these efforts might add time and cost at the front end of site development, they certainly pay off in the long run when you consider the risks and losses that they prevent. No additional expenditures for equipment or software are required either, and websites that have implemented **Always-On SSL** have noticed only negligible decrease in speed or performance. In fact, the speed of SSL can be optimized by using techniques like Elliptic Curve cryptography (ECDHE), keep-alives / session resumption (with caching, subsequent sessions can be established with an abbreviated handshake), and OCSP Stapling to provide certificate validity during the handshake.

Fix Server Protocol and Ciphersuite Settings

Good server configuration and patch management are key to the successful deployment of **Always-On SSL**. In previous publications, the CA Security Council has emphasized the importance of optimization, ciphersuite selection, and correct protocol settings. See https://casecurity.org/2013/09/19/its-time-for-tls-1-2/, https://casecurity.org/2013/03/08/rsa-recap-securing-your-site/, and https://casecurity.org/2013/09/13/encryption-still-works-its-about-how-you-implement-it/. If vulnerabilities exist that allow attackers to work around or downgrade your security, then **Always-On SSL** will only provide a false sense of security. Not only should system administrators disable weak ciphers and SSL ≤ version 3.0, but they should also install vendor-supplied security patches and critical updates quickly (e.g. within one month of release) and implement automated systems to scan for, identify, and address vulnerabilities. See sections 6.1 and 6.2 of the PCI Data Security Standards and NIST 800-40, Guide to Enterprise Patch Management Technologies.

Action Items:

- Scan for and replace all instances of "http" with "https"
- Scan for and remove mixed content and instances that lead to embedded vulnerabilities
- Disable null ciphers, short bit lengths, the MD5 algorithm, and SSL version 2
- Implement an automated vulnerability detection and patch management solution
- Review up-to-date information on other configuration vulnerabilities
- Take the recommended server configuration steps outlined by the CA Security Council here: **https://casecurity.org/2013/06/28/getting-the-most-out-of-ssl-part-2-configuration/**

Implement HTTP Strict Transport Security (HSTS)

HSTS is a way to tell a web browser to always connect to a domain over https, so that even if a page somewhere says, "connect by http", the browser will switch it over https. It is like an insurance policy that ensures the use of HTTPS for your web site in case of an intentional or unintentional attempt to direct the user to an unencrypted http resource. It is described in [RFC 6797] and is supported at this time by Google Chrome, Mozilla Firefox, and the Opera browser. To implement HSTS, "Strict-Transport-Security" entry is included in the HTTP response header. This is then cached by the browser to identify which sites are "HTTPS Only," and it is used by the browser when opening up subsequent sessions with the server's domain. The result is that it helps prevent subsequent man-in-the-middle attacks, like the one described for SSL Strip above.

Action Items:

- Add the Strict-Transport-Security: max-age=15768000 ; includeSubDomains header to your HTTPS responses
- Consider warning site visitors if they are using older versions of browsers (earlier than IE 8, Firefox 14, Safari 5, or Chrome 26)
- Take the recommended optimization steps outlined by the CA Security Council here: **https://casecurity.org/2013/07/29/getting-the-most-out-of-ssl-part-3-optimization/**

Use a High-Quality CA Service Provider and Extended Validation SSL

Another important step in implementing **Always-On SSL** is to obtain an Extended Validation (EV) SSL Certificate and ensure that it is properly installed. A publicly trusted EV SSL certificate from a reputable Certification Authority means that a site is entitled to enhanced trust. The CAs issuing EV certificates have their root certificates embedded in browser trust stores. Sites like PayPal, Twitter, and Facebook implement Extended Validation SSL certificates. Users visiting their websites are given positive reassurance—a green-colored box or lettering, along with the organization's name in the URL bar. These positive visual signals not only reinforce the reputation of the website operator, but also communicate identity and provide visitors with confirmation that they have returned to the right web site. While EV Certificates are highly recommended, websites that already have other types of publicly trusted SSL certificates issued by members of the CA Security Council do not need to buy anything new because **Always-On SSL** is a holistic solution, and not a new product to replace an existing SSL certificate. It is, however, important to ensure that the SSL certificate has been properly installed.

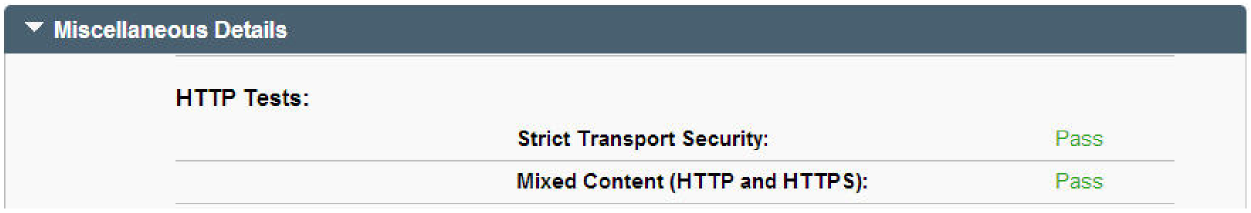
Missing intermediate CA certificates in a trust chain can cause problems, as can other configuration errors. Many of these problems can be checked and fixed by using a number of online tools. (*See **SSL Tools*** section on the following page.) Given today's security threats, a new mindset is needed. CAs, browsers, and website operators need to take on more responsibility for the entirety of the SSL communication—end-to-end and side-to-side. If Google, Facebook, PayPal, and other high-volume websites with diverse user systems are able to implement **Always-On SSL**, then many other websites should be able to do it as well.

Action Items:

- Install and configure an Extended Validation SSL Certificate
- Test Servers for other configuration issues, including server protocol settings, next

Use the CA Security Council's SSL Checking Tool

As a first step toward improving your site's security, use the CA Security Council's Certificate Checking Tool to see how well SSL/TLS is configured: https://sslcheck.casecurity.org Using this tool, you can test any SSL server for a number of security vulnerabilities, including SSL v.2, weak ciphersuites, and lack of HTTP Strict-Transport-Security. Specifically for this latter vulnerability, under "Miscellaneous Details," you should see a passing grade on "Strict Transport Security", as follows:



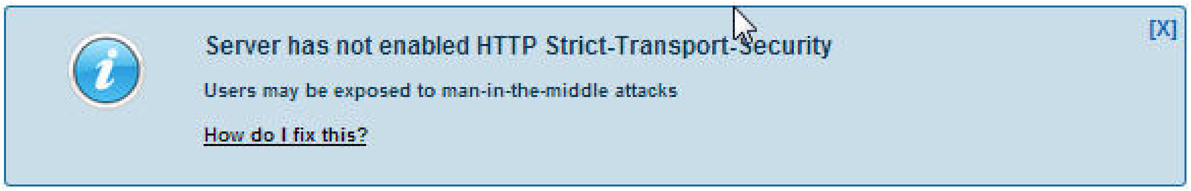
Miscellaneous Details

HTTP Tests:

Strict Transport Security:	Pass
Mixed Content (HTTP and HTTPS):	Pass

and **not**: **Strict Transport Security:** Fail.

Other related error messages appear below.

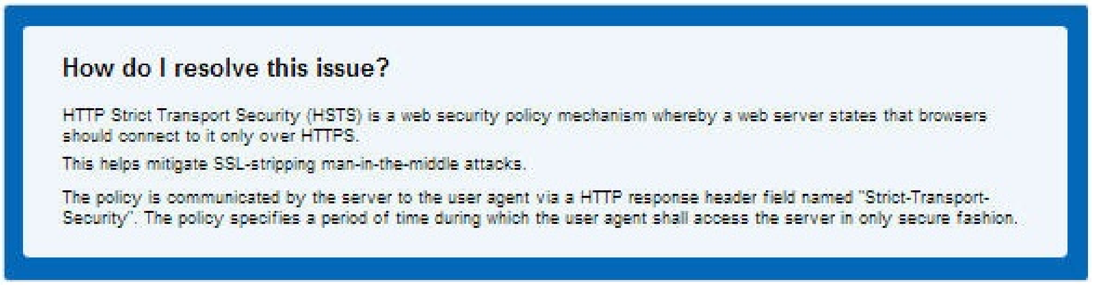


Server has not enabled HTTP Strict-Transport-Security

Users may be exposed to man-in-the-middle attacks

How do I fix this?

Users may be exposed to man-in-the-middle attacks
When we tested your webserver, we discovered the following issue:
Server has not enabled HTTP Strict-Transport-Security.



How do I resolve this issue?

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server states that browsers should connect to it only over HTTPS.

This helps mitigate SSL-stripping man-in-the-middle attacks.

The policy is communicated by the server to the user agent via a HTTP response header field named "Strict-Transport-Security". The policy specifies a period of time during which the user agent shall access the server in only secure fashion.

Conclusion

Over the last several years attackers have discovered new ways to attack weaknesses in the configuration of SSL. In response to SSL Strip, Firesheep, and other circumvention and man-in-the-middle attacks, SSL implementations must move to combination of defenses known as **Always-On SSL**. We can no longer conduct business on the Internet "as usual" by only defending the castle. Additional defensive measures are needed beyond those that protect against direct attacks on web servers. **Always-On SSL** protect client-side connections from coffee shops, libraries, airports, schools, and other public places around the world. **Always-On SSL** already protects the brand reputations of the likes of Google, Facebook, Twitter, and PayPal by securing billions of remote connections to their SSL servers every day. The CA Security Council supports **Always-On SSL** and encourages all service providers to embrace it as an essential component of every web-based service offering.

Take-Aways

- Visitors to your site are vulnerable to client-side, man-in-the-middle attacks
- Implement HTTP Strict Transport Security – HSTS
- Replace all "http" references with ones that use "https" instead
- Don't allow "mixed content" on your site
- Set your cookie flags to "secure"
- Obtain an Extended Validation SSL Certificate from a member of the CA Security Council
- Use https://sslcheck.casecurity.org to scan your site for protocol vulnerabilities

References

Online Trust Alliance - Always on SSL White Paper - https://otalliance.org/resources/AOSSL/index.html (Available in English, Spanish, German, Japanese, and French)

IETF RFC 6797 - HTTP Strict Transport Security (HSTS) – http://tools.ietf.org/html/rfc6797

PCI Data Security Standards – https://www.pcisecuritystandards.org/security_standards/

NIST 800-40, Guide to Enterprise Patch Management Technologies– http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf

About the CA Security Council

The CASC is an advocacy group comprised of leading global CAs committed to the exploration and promotion of best practices that advance the security of websites and online transactions.

For more information, visit us at:

https://casecurity.org