SecureNet

Whitepaper

HTTP STRICT TRANSPORT SECURITY (HSTS)

Thomas Schreiber, SecureNet GmbH

Sven Schleier, SecureNet GmbH

1.11.2012

SecureNet GmbH - Frankfurter Ring 193a - D-80807 München
www.securenet.de - info@securenet.de - Phone +49/89/32133-600

Inhalt

1 Einleitung

Das Ausschalten der SSL-Verschlüsselung im Browser durch einen Man-in-the-Middle Angriff ist eine schwerwiegende Sicherheitslücke, die in ungeschützten Umgebungen (z.B. im WLAN) leicht ausgenutzt werden kann. Die Folge ist der komplette Verlust der Vertraulichkeit: Bankkonto-, Email- und Social-Network-Zugangsdaten gelangen dabei ebenso in die Hände des Angreifers wie Kreditkartendaten oder vertrauliche Geschäftsinformationen.

Mit dem *HSTS-Header* existiert ein Schutzmechanismus, mit dem der Serverbetreiber die Gefahr der erfolgreichen Durchführung eines solchen Angriffs auf ein Minimum reduzieren kann. Diese Schutzmaßnahme wird, wie eine Analyse von 1 Mio. Websites gezeigt hat (siehe [25]), aber bisher nur in wenigen Fällen angewandt.

In diesem Whitepaper erklären wir SSL-Strippung und geben konkrete Gegenmaßnahmen an. Dabei steht der HSTS-Header im Vordergrund. Da diese Schutzmaßnahme aber eine Lücke offen lässt und der Header auch noch nicht von allen Browsern unterstützt wird, werden weiterführende Empfehlungen gegeben.

In erster Linie richtet sich das Whitepaper an Serverbetreiber und die Ersteller von Webanwendungen. Aber auch aus Nutzersicht wird gezeigt, wie man der Gefahr, selbst zum Angriffsopfer zu werden, aus dem Weg gehen kann.

2 Was ist SSL-Stripping?

Ein Angreifer kann SSL-Stripping immer dann durchführen, wenn er das Gateway kontrolliert, über das Benutzer mit dem Internet verbunden sind. Am einfachsten geht das durch Aufstellen eines *Rogue Access Points*, also eines vertrauenswürdig aussehenden aber bösartigen WLAN-Hotspots, zu dem der ahnungslose Benutzer sich verbindet. Aber auch in kabelgebundenen Netzen, und möglicherweise noch anderweitig, existieren Techniken, den Benutzer auf das Angreifer-Gateway zu zwingen.

Jetzt ist es ein Leichtes, in jeder HTTP-Seite, die der Benutzer aufruft, die HTTPS-Links durch HTTP-Links auszutauschen. Klickt der Benutzer im Anschluss auf einen solchen Link und achtet nicht darauf, dass in der Adressleiste auch tatsächlich das Schloss auftaucht, so erkennt er nicht, dass seine Eingaben unverschlüsselt über die Leitung gehen und auf dem Gateway ausgespäht werden. Serverseitig kann dieser Angriff nicht erkannt werden, denn der Man-in-the-Middle (MitM) übersetzt jeden solchen HTTP-Request nun wieder zurück in einen HTTPS-Request und gibt diesen an den Server weiter..

Der Angriff ist ausführlicher hier beschrieben: [26]. Das Video [27] demonstriert, wie schnell man als Benutzer auf diesen Trick hereinfallen kann.

Der SSL-Stripping-Agriff wurde 2009 erstmals beschrieben (siehe [14]). Gleichzeitig wurde das Tool *sslstrip* vorgestellt, mit dem es auch weniger versierten Personen möglich ist, den Angriff auszuführen.

Wie wenig vom Benutzer abverlangt werden kann, dass er sich durch umsichtiges Verhalten selbst vor SSL-Stripping schützt, zeigt eine im August 2011 veröffentlichte empirische Studie des Computer Science Department der New Mexico Tech Universität (siehe [22]): Dabei wurde das Verhalten von Benutzern bei einer SSL-Stripping Attacke und dem Erscheinen von Hinweisen auf die Attacke untersucht. Als ein Ergebnis kam heraus, dass sich die meisten Benutzer selbst dann, wenn eine Rotfärbung der Eingabefelder die unverschlüsselte Übertragung kennzeichnet, nicht davon abhalten lassen, sensitive Daten einzugeben.

3 HSTS richtig einsetzen

3.1 Beschreibung

Am 18.9.2009 veröffentlichte das W3C den ersten Draft einer Spezifiaktion mit dem Namen Strict Transport Security (STS), um die oben beschriebene Form des Angriffes auf HTTPS zu verhindern. Dieser Entwurf wurde im Oktober 2012 von der IETF als vorgeschlagener Standard akzeptiert und in HTTP Strict Transport Security (HSTS) umbenannt.

HSTS wird vermittels des HTTP-Response-Headers `strict-transport-security` "eingeschaltet". Ist der Header gesetzt, d.h. wird er von einer Website in der Antwortseite zurückgeliefert, so wird der Browser von nun an alle HTTP-Zugriffe auf diese Site automatisch in HTTPS-Zugriffe umwandeln. Und zwar solange, wie es die Gültigkeitsdirektive im Header angibt.

HSTS sorgt weiterhin dafür, dass immer dann, wenn das Serverzertifikat vom Browser als nicht vertrauenswürdig eingestuft wird, die Verbindung zum Server komplett unterbunden wird. Ohne HSTS zeigt der Browser eine Warnung an, überlässt es aber dem Benutzer, sich darüber hinwegzusetzen und den Server dennoch anzusurfen. Die sogenannte "Click-Through Insecurity", also das mechanische Ignorieren derartiger Warnungen durch zumeist weniger erfahrende Benutzer, wird damit ebenfalls verhindert. Abschnitt 7 geht darauf näher ein.

Ein Nebeneffekt von HSTS ist, dass auch Cookie-Forcing-Angriffe (siehe [12]) verhindert werden, da Cookies nur noch über HTTPS verändert werden können.

HSTS ist wird schon seit geraumer Zeit von vielen Browsern unterstützt. Nicht dazu gehören jedoch Internet Explorer und Safari! Eine Übersicht gibt Abschnitt 4.

3.2 Header Syntax für HSTS

Der HSTS-Header hat folgende Syntax (siehe [1]):

```
strict-transport-security max-age=Gültigkeitsdauer;
includeSubDomains
```

`max-age` wird in Sekunden angegeben (z.B. 31.536.000 für 12 Monate, oder 86.400 für 1 Tag) und definiert, wie lange der sendende Host vom Browser als HSTS-Host geführt wird. Das Attribut ist verpflichtend, der Wert kann mit oder ohne Anführungszeichen angegeben werden. Da der HSTS-Header mit jeder Seite mitgeschickt wird, läuft die angegebene Gültigkeitsdauer ab dem Zeitpunkt, zu dem der Browser zum letzten Mal auf die Site zugegriffen hat. `max-age=0` signalisiert dem Browser, den angegeben Host nicht mehr als HSTS Host zu führen, so dass ab diesem Moment HTTP-Zugriffe wieder möglich sind.

`includeSubDomains` ist eine optionale Direktive, für die kein Wert gesetzt wird und die HSTS auf die Subdomains des Hosts erweitert.

Beispiel

Der folgende Header legt fest, dass die HSTS Policy für ein Jahr im Browser gültig ist:

```
Strict-Transport-Security: max-age=31536000
```

3.3 Empfohlene Gültigkeitsdauer für HSTS

HSTS hat prinzipbedingt eine Schwachstelle: Erfolgt der Erstzugriff - oder der erste Zugriff nach dem Ablauf von `max-age` - auf die jeweilige Website nicht über ein vertrauenswürdiges Netz, so kann der Angreifer dafür sorgen, dass der Header den Browser nicht erreicht, SSL-Stripping also weiterhin möglich ist.

Daraus ergibt sich unmittelbar die Anforderung einer sehr hohen Gültigkeitsdauer (`max-age`). Unter einem halben Jahr (`max-age=15768000`) sollte der Wert nicht liegen, besser bei einem Jahr (`max-age=31536000`) oder zwei Jahren (`max-age=63072000`).

3.4 Verbreitung von HSTS

Anhand der Alexa-Liste mit den 1 Mio. meistbesuchten Websites haben wir die Verbreitung von HSTS untersucht (siehe [25]). Dabei kam heraus, dass weniger als 0,5% der HTTPS-Websites weltweit und weniger als 0,1% der deutschen HTTPS-Websites HSTS verwenden. Unter den deutschen Banken ist nur eine einzige, die ihre Onlinebanking-Nutzer vermittels des HSTS-Headers ausreichend schützt.

Dabei wurde auch beobachtet, dass viele Websites mit HSTS den Wert viel zu niedrig eingestellt haben.

Die Erklärung für diesen niedrigen Wert könnte darin liegen, dass es den Administratoren nicht geheuer ist, durch Einstellung eines hohen Wertes die Kontrolle über die Rücknahme dieser Einstellung abzugeben, etwa dann, wenn sich doch Probleme zeigen, weil Auswirkungen übersehen wurden. Denn je höher der Wert, desto weniger Einfluss kann im Falle eines unvorhergesehenen Ereignisses auf all die Browser genommen werden, die den Wert bereits gesetzt haben. Es sei daher hiermit noch einmal darauf hingewiesen, dass die Einstellung durch Aussenden von `max-age=0` jederzeit wieder zurückgenommen werden kann.

Am Ende bleibt die Tatsache bestehen: Ein kurze Gültigkeitsdauer bringt die ganze Maßnahme in den Bereich der Nutzlosigkeit.

4 Browserunterstützung für HSTS

Wir haben die Browser mit der aktuell höchsten Verbreitung auf dem Desktop und auf Mobilgeräten auf ihre Unterstützung von HSTS hin untersucht.

4.1 Desktop

Die Tests fanden unter Windows 7 statt. Für den Internet Explorer 10 kam Windows 8 Enterprise Evaluation Build 9200 zum Einsatz.

Die folgende Tabelle gibt die Ergebnisse mit den zum Testzeitpunkt aktuellen Browserversionen wieder:

Browsername	Version	HSTS implementiert
Chrome	22.0.1229.79 m	Ja
Internet Explorer	9.0.8112.16421	Nein
Internet Explorer	10.0.9200.16384	Nein
Mozilla Firefox	15.0.1	Ja
Opera	12.02	Ja
Safari	5.1.7	Nein

„Preloaded HSTS Sites“ in Chrome

In Chrome ist ein zusätzliches Feature verfügbar: Eine fest eingebaute Liste mit Sites, die *immer* per HTTPS aufgerufen werden. Damit wird das kleine Fenster, in dem der Benutzer ungeschützt ist, effektiv geschlossen. Die Liste

enthält im Moment über 100 Domains (siehe [8]), die einzigen .de-Domains sind piratenlogin.de, entropia.de und www.entropia.de.

Firefox wird ab Version 17 ebenfalls eine HSTS Liste enthalten (siehe [21]). Allerdings wird die Liste sehr restriktiv sein, da dort nur Seiten eingepflegt werden sollen, die auch tatsächlich einen HSTS Header ausliefern. Der `max-age` Wert muss mindestens 10.886.400 Sekunden sein (18 Wochen).

4.2 Mobile

Android

Die Android Tests wurden auf einem Nexus 7 mit Android 4.1.1 (Jelly Bean) durchgeführt.

Browsername	Version	HSTS implementiert
Chrome	M18.1	Ja
Firefox	16.0	Ja
Opera Mini	7.5.31657	Nein
Opera Mobile	12.1	Ja

iOS

Die iOS Tests fanden auf einem iPad 3 unter iOS 6.0 statt.

Browsername	Version	HSTS implementiert
iOS Safari	6.0	Nein
Opera Mini	7.0	Nein
Chrome	21.0.1180.82	Ja

5 Tipps für den Serverbetreiber

Um die Wahrscheinlichkeit für das Gelingen von SSL-Stripping Angriffen weiter zu verringern, empfehlen sich zusätzliche Maßnahmen, die im Folgenden aufgeführt sind. Diese ersetzen nicht die Aktivierung von HSTS, sondern ergänzen sie vielmehr.

5.1 „ sslstrip-Buster“

Das frei verfügbare Angriffstool *sslstrip* [14] macht es auch Programmierlaien leicht, SSL-Stripping zu betreiben. Um zumindest die auszuschalten (bzw. dafür zu sorgen, dass die sich ein leichteres Zeil suchen), hilft ein wenig JavaScript-Code in den schutzbedürftigen Webseiten:

Mit der Anweisung

```
window.location.protocol
```

kann auf dem Client überprüft werden, welches Protokoll zur aktuellen URI gehört und damit, ob die Seite per HTTP oder per HTTPS aufgerufen wurde. Bei Abweichung wird der Aufruf unterbunden gemäß dem hier gezeigten Muster:

```
if (location.protocol !== 'https:') {

      document.body.className="error";

      … z.B. Hintergrund abdunkeln und es erscheint ein Overlay
      mit einem Hinweis, dass diese Seite per HTTPS aufgerufen
      werden muss, aber nur über HTTP und damit unverschlüsselt
      aufgerufen wurde…

}
```

Generell ist dies natürlich nur ein rudimentärer Schutz, da ein zielgerichtet vorgehender Angreifer auch diese Anweisungen entfernen und so diese Mechanismen wieder deaktivieren kann. Aber wenigstens hält man solange, wie *sslstrip* eine Erkennung hierfür noch nicht anbietet, all die Gelegenheitsangreifer davon ab, die eigenen Kunden zu schädigen. In Kombination mit Quelltextverschleierung (Obfuscation) oder weiteren individuellen Maßnahmen kann der Aufwand für den Angreifer weiter erhöht werden.

5.2 Ausschließlich HTTPS verwenden

Am Beispiel einer Bank lässt sich das Problem verdeutlichen: Ein nicht geringer Teil der Nutzer geht zum Erreichen der Onlinebanking-Anwendung nach der Strategie vor, zunächst die Homepage der Bank aufzusuchen (vielleicht sogar danach zu googlen oder sich ein Bookmark auf die Startseite der Bank anzulegen), um von dort aus per Klick auf den entsprechenden Link zum eigentlichen Ziel zu navigieren. Ist die Homepage per HTTP erreichbar, so ist die Wahrscheinlichkeit hoch, dass dieser Einstiegslink ein HTTP-Link ist und damit auch für das SSL-Stripping den Einstieg darstellt. Daraus folgt:

Nicht nur die sensitiven Seiten selbst sollten ausschließlich per HTTPS erreichbar sein, auch die Seiten, auf denen sich Links zu den HTTPS-Seiten befinden, sollten ausschließlich per HTTPS erreichbar sein.

5.3 HTTPS-Links explizit machen

HTTPS-Links sollten für den Benutzer nicht erst nach dem Klick im URL erkennbar sein, sondern bereits anhand des Seiteninhalts, z.B. durch ein Schloss-Icon neben dem Link-Text.

Tabu sein sollten Formulare, die sich auf einer HTTP-Seite befinden und bei denen der Übergang zu HTTPS erst beim Absenden des Formulars, also beim Klick, erfolgt. Ganz besonders gilt das natürlich für das Login-Formular! Anders hat selbst der aufmerksame Benutzer keine praktikable Möglichkeit, das Nichtvorhandensein von HTTPS rechtzeitig zu erkennen.

5.4 Hinweise an den Benutzer

Der Benutzer ist auf die Gefahr hinzuweisen, insbesondere darauf, auf das Schloss im Adressfeld zu achten. Ihm sollten Empfehlungen wie die in Abschnitt 6 gegeben werden.

Auch das sollte man in Erwägung ziehen: Benutzer von Internet Explorer und Safari an den Übergängen zu den sensiblen Bereichen zusätzlich explizit darauf hinweisen, dass sie mit einem unsicheren Browser unterwegs sind.

6 Tipps für den Benutzer

Auch wenn wir der Ansicht sind, dass der Website-Betreiber alles dafür tun muss, dass der Benutzer sich beim Betreten der Website nicht durch Unwissenheit in Gefahr bringen kann, liegt am Ende die Verantwortung doch beim Benutzer selbst. Daher die folgenden Tipps zum Selbstschutz gegen HTTPS-Stripping.

6.1 Auf das Schloss achten!

Wann immer man sich außerhalb der vertrauten Netzwerkumgebung befindet: Auf das Schloss achten! Login und alle Formulare, die vertrauliche Informationen abfragen, nur dann absenden, wenn erkennbar ist, dass man sich auf einer Seite mit einem Schloss befindet!

6.2 Browser-Extensions einsetzen

Dieser Tipp dient dazu, die Gefahr des versehentlichen Verstoßes gegen die vorstehende Regel zu minimieren.

Mit *NoScript* (siehe [16] und [17]) und *HTTPS Everywhere* (siehe [10] und [11]) existieren zwei Browser-Extensions, die HTTPS erzwingen. Bei NoScript kann man eine eigene Liste von Websites erstellen, HTTPS Everywhere bringt eine solche Liste mit rund 3.000 Einträgen bereits mit. Beide sorgen sie dafür, dass die enthaltenen Websites nur per HTTPS aufgerufen werden können.

Diese Erweiterungen können im Moment jedoch nicht als Workaround oder Übergangslösung für die Browser dienen, die noch keine HSTS Unterstützung haben, da sie nur für Firefox und Chrome verfügbar sind. An einem Port von HTTPS Everywhere für den Internet Explorer wird gearbeitet (siehe [11]).

6.3 Einen sicheren Browser verwenden

In Abschnitt 4 sind die Browser aufgeführt, die HSTS unterstützen und somit vor SSL-Stripping Angriffen auf HSTS-geschützte Websites schützen. Da Internet Explorer und Safari nicht dazu gehören, ist von ihrer Verwendung in ungesicherten Umfeldern abzuraten. Wann Internet Explorer und Safari HSTS unterstützen werden, ist nicht bekannt.

6.4 VPN erzwingen

Über das Internet erreichbare Unternehmensportale für Außendienstmitarbeiter, Lieferanten, etc. sind häufig durch eine stärkere Authentisierung geschützt, z. B. durch Einmaltoken. Diese stellen keinen Schutz vor SSL-Stripping dar!

Die Best-Practice für Firmen, die verhindern wollen, dass ihre Mitarbeiter durch Fehlverhalten in der in diesem Papier beschriebenen Weise zum Sicherheitsrisiko werden, lautet: Die Unterwegs-Laptops und Homeoffice-PCs der Mitarbeiter sind per VPN ins Unternehmens-LAN zu zwingen. Dann geht jeder Verbindungsaufbau des Browsers, egal ob HTTP oder HTTPS, unangreifbar durch den geschützten Kanal ins Firmennetz und von dort aus zum jeweiligen Webserver.

7 Verhinderung von "Click-Through Insecurity"

Einen nicht zu unterschätzenden Sicherheitsgewinn bringt auch die oben schon angesprochene Ablehnung nicht vertrauenswürdiger Zertifikate bei vorliegendem HSTS.

Liefert eine Website ein solches aus, so reagieren alle Browser im nicht-HSTS Fall ähnlich wie das hier gezeigte Chrome-Beispiel:

D. h. sie ermöglichen dem Benutzer, die gewünschte Site trotzdem aufzurufen.

Die Praxis zeigt, dass Benutzer sich der Gefahr häufig nicht bewusst sind und nur bestrebt sind, so schnell wie möglich weiterzumachen. Sie wählen dann "Trotzdem fortfahren" und haben damit im Falle eines Angriffs das Zertifikat des Man-in-the-Middle als vertrauenswürdig angenommen. Jetzt kann dieser die Kommunikation trotz SSL-Verschlüsselung belauschen.

Durch HSTS verändert sich das Verhalten des Browsers. Er erlaubt es dem Benutzer nun nicht mehr, sich über die Warnung hinwegzusetzen und schützt ihn so vor sich selbst:

Dieses Verhalten ist von der IETF im HSTS-Standard lediglich als Kann-Kriterium angegeben (siehe [18]), wird aber von Chrome und Firefox erzwungen.

Das manuelle Eintragen von selbstsignierten oder anderweitig nicht vertrauenswürdigen Zertifikaten in die Liste vertrauenswürdiger Zertifikate (alle Browser bieten eine solche Möglichkeit an oder stützen sich auf die diesbezüglichen Möglichkeiten des Betriebssystems) funktioniert auch mit HSTS

weiterhin. Die gängige Praxis, innerhalb der Firma den Browser mit bereits vorbelegtem selbstsigniertem Firmenzertifikat auszurollen, erfährt also keine Einschränkungen.

8 Servereinstellung

Die HSTS-Befähigung einer Website erfolgt im Webserver. Alle gängigen Webserver bieten hierzu entsprechende Konfigurationsmöglichkeiten, die im Folgenden einzeln aufgeführt werden.

8.1 Apache

Um Apache entsprechend konfigurieren zu können, muss das Header-Modul in Apache aktiviert sein (`a2enmod headers`). Die folgende Einstellung muss in der Apache Virtual-Host Konfiguration für Seiten die über HTTPS erreichbar sind, aufgenommen werden (siehe auch [5]):

```
Header always set Strict-Transport-Security "max-age=31536000;
includeSubDomains"
```

8.2 lighttpd

Die folgende Einstellung muss in der Konfiguration von lighttpd übernommen werden, um HSTS zu aktivieren (siehe auch [5]):

```
server.modules += ( "mod_setenv" )
$HTTP["scheme"] == "https" {
    setenv.add-response-header  = ( "Strict-Transport-Security"
=> "max-age=31536000;includeSubDomains")
}
```

8.3 nginx

Die folgende Einstellung muss in der Konfiguration von nginx übernommen werden, um HSTS zu aktivieren (siehe auch [5]):

```
add_header Strict-Transport-Security „max-age=31536000;
includeSubDomains“;
```

8.4 IIS 6

Die folgende Einstellung muss in der Konfiguration von IIS 6.x übernommen werden, um HSTS zu aktivieren (siehe auch [2]):

```
cscript adsutil.vbs set w3svc/HttpCustomHeaders "Strict-
Transport-Security: max-age=31536000;includeSubDomains"
```

`cscript` kann unter `c:\inetpub\AdminScripts` aufgerufen werden.

8.5 IIS 7

Die folgende Einstellung muss in der Konfiguration von IIS 7.x übernommen werden, um HSTS zu aktivieren (siehe auch [4]):

```
appcmd set config /section:httpProtocol
/+customHeaders.[name='Strict-Transport-Security',value='max-
age=31536000;includeSubDomains']
```

Siehe unter [3] für Informationen zu `appcmd`; `appcmd` befindet sich unter `%windir%\system32\inetsrv`.

8.6 Generische Variante

Der englische Wikipedia-Eintrag gibt Auskunft über die Einstellungen für eine Reihe weiterer Umgebungenen, darunter PHP, Perl, ASP. Siehe [5].

Stellt die eingesetzte Systemtechnik keine solche Konfiguration zur Verfügung, muss direkt in den Response-Header eingegriffen werden:

Der Eintrag lautet:

```
strict-transport-security: max-age=Gültigkeitsdauer;
includeSubDomains
```

Der Header wirkt sich nur aus, wenn er via HTTPS übertragen wird.

Als Best Practice sollten für HSTS alle nicht verschlüsselten Anfragen über Port 80 direkt auf Port 443 umgeleitet werden. Im Apache z. B. würde das diese Rewrite-Regel im Kontext des Virtual Hosts erledigen:

```
RewriteEngine On
RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [redirect=301]
```

9 Fazit

SSL-Stripping ist eine Angriffstechnik, die schwerwiegende Auswirkungen auf die Sicherheit des Benutzers hat. Mit HSTS existiert eine serverseitige Abwehrmaßnahme, die die Chance einer erfolgreichen Ausführung dieses Angriffs signifikant herabsetzt. Bei so gut wie allen gängigen Webserverplattformen ist der Schutz leicht einzustellen. Er muss jedoch einhergehen mit einem Website-weit wohl konzipierten Umgang mit SSL.

Systembedingt bleibt die Unsicherheit, dass das erstmalige Betreten der HSTS-geschützten Website bereits über ein ungeschütztes Netzwerk erfolgt und der HSTS-Header vom Angreifer herausgefiltert wird. Um die Wahrscheinlichkeit, dass dieser Fall eintritt, so weit wie möglich herabzusetzen, ist die Gültigkeitsdauer für HSTS auf einen sehr hohen Wert zu setzen. Zur weiteren Erhöhung des Schutzes sind zusätzliche Maßnahmen zu ergreifen, insbesondere auch, um

den Benutzern von Browsern, die HSTS nicht unterstützen, ein Mindestmaß an Schutz zukommen zu lassen.

Durch den Einsatz von HSTS bewahrt man seine Benutzer ganz nebenbei auch davor, dass sie unbedacht die Browserwarnung eines ungültigen Zertifikats wegklicken und so trotz verschlüsselter Verbindung vom Angreifer abgehört werden können.

10 Quellen

[1] Syntax von HSTS
http://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-14#section-6

[2] IIS 6.x Header Konfiguration
http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/ce111dee-a127-47e1-aa33-4009d692585e.mspx?mfr=true

[3] Appcmd.exe unter IIS7
http://technet.microsoft.com/en-us/library/cc772200(v=ws.10).aspx

[4] IIS 7.x Header Konfiguration
http://technet.microsoft.com/en-us/library/cc753133(WS.10).aspx

[5] Wikipedia Artikel zu HSTS mit Servereinstellungen:
http://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security

[6] Chromium STS:
http://dev.chromium.org/sts

[7] HTTP Strict Transport Security in Chromium:
http://blog.chromium.org/2010/01/security-in-depth-new-security-features.html

[8] Liste in Chrome von Domains die nur per HTTPS erreichbar sind:
http://src.chromium.org/viewvc/chrome/trunk/src/net/base/transport_security_state_static.json

[9] HSTS Proposed Standard von IETF:
http://www.ietf.org/mail-archive/web/ietf-announce/current/msg10748.html

[10] HTTPS Everywhere nicht für Internet Explorer und Safari verfügbar:
https://www.eff.org/https-everywhere/faq

[11] HTTPS Everyhwere für IE:
http://research.zscaler.com/2012/08/zscaler-safe-shopping-11-for-internet.html

[12] Cookie Forcing:
http://michael-coates.blogspot.de/2010/01/cookie-forcing-trust-your-cookies-no.html
http://osvdb.org/show/osvdb/74453

[13] Vorteile von HSTS:
http://scarybeastsecurity.blogspot.de/2011/02/some-less-obvious-benefits-of-hsts.html

[14] sslstrip
http://www.thoughtcrime.org/software/sslstrip/

[15] Debugging UI für HSTS in Chrome

http://www.imperialviolet.org/2011/02/17/hstsui.html

[16] HSTS FAQ für NoScript
http://noscript.net/faq#qa6_6

[17] HSTS in NoScript
http://hackademix.net/2009/09/23/strict-transport-security-in-noscript/

[18] No insecure „clicking through“ anymore when certificate errors are presented
https://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-14#section-8.4
https://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-14#section-12.1

[19] HSTS Pre-Loaded Liste
https://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-14#section-12.3

[20] Qualys – State of SSL
http://blog.ivanristic.com/Qualys_SSL_Labs-State_of_SSL_2010-v1.6.pdf

[21] HSTS Preload List ab FF Version 17
https://wiki.mozilla.org/Privacy/Features/HSTS_Preload_List

[22] Studie über SSLStrip
https://www.acsac.org/2011/openconf/modules/request.php?module=oc_program&action=view.php&a=&id=189&type=3

[23] OWASP HSTS Beschreibung
https://www.owasp.org/index.php/HTTP_Strict_Transport_Security

[24] Alexa Liste der "Top 1,000,000 Sites“
http://www.alexa.com/topsites

[25] Untersuchung der Verbreitung von HTTP Strict Transport Security, SecureNet
http://www.securenet.de/fileadmin/papers/HTTP_Strict_Transport_Security_HSTS_Verbreitung.pdf

[26] „SSL-Stripping, die ignorierte Gefahr“
http://blog.securenet.de

[27] „SSL-Stripping, die ignorierte Gefahr“, Live-Demo
http://www.youtube.com/watch?v=79vBEohGtDM

Über SecureNet

Die SecureNet GmbH ist Softwarehaus und Rundum-Dienstleister für Web Application Security. Seit 8 Jahren führt SecureNet Penetrationstests, Codeanalysen sowie umfassende Beratung zum Aufbau der Softwaresicherheit durch und gibt in Best-Practice-Seminaren die Erfahrungen an Entwickler, Sicherheits- und Fachverantwortliche weiter.

www.securenet.de