HSTS

Policy Mechanism for Practical End-to-End Web Security

Author: ADAM MIHAI GERGELY

ABSTRACT. This paper presents the current characteristics of the HSTS protocol, its limitations and aims to provide a theoretical new solution for dealing with the existing problems of the HSTS protocol. The key points of this article are the general concept behind HSTS, some improvements that can improve upon this concept. It also describes the time problem of HSTS and introduces a more secure way of dealing with the timing issues of HSTS.

KEYWORDS

Paper Keywords:

- Computer Science Security
- Cryptography & P.K.I.
- Digital Signatures & Certificates
- Privileged Management Infrastructure
- Attribute Certificates
- Distributed Computing

Paper domain classification:

- ACM:
  - Security and Privacy
    - Cryptography
    - Security
- AMS:
  - 68: Computer Science
    - 68N: Software
      - 68N01: General

1. INTRODUCTION

This paper deals with the concept of migrating the time checking sequence of HSTS to the server-side, by using a more secure mechanism to handle the “max-age” parameter of the HSTS policy enforcement.

In the current specifications of HSTS, the web clients receive the “max-age” attribute in the communication header message as a simple notification, not a requirement. After the timer expires, the connection may revert back to normal communication.

Also, there should be a the possibility of infinite time for HSTS policy enforcement. The current specifications of HSTS allow for a finite time, however there are a lot of websites that rely exclusively on HSTS communication for as long as they exist. It is only natural that there should be a possibility of “infinite time” defined in the HSTS policy as well.

The theoretical improvement this paper presents states that a client must receive a “hsts-stop” signal from the server before actually terminating the HSTS policy.

In the improvements presented by this paper, there is another improvement which should be implemented, for better security. It describes the server taking on the responsibility of maintaining a correct counting of time by using different server-side technologies to achieve this. Leaving the timing to the client could present a problem, as clients are vulnerable to time-bases attacks or manipulations.

2. STATE OF ART

2.1 Definition and proof of concept

HSTS stands for HTTP Strict Transport Security and is defined in IETF's RFC #6797.

This specification defines a mechanism enabling web sites to declare themselves accessible only via secure connections and/or for users to be able to direct their user agent(s) to interact with given sites only over secure connections. This overall policy is referred to as HTTP Strict Transport Security (HSTS). The policy is declared by web sites via the Strict-Transport-Security HTTP response header field and/or by other means, such as user agent configuration, for example. [1]

Basically, the web server running HSTS requires web clients to use and enforce HTTPS only connection, but for a limited period of time.

HSTS addresses multiple threats, such as:

- Passive Network Attackers
- Active Network Attackers
- Web Site Development and Deployment Bugs [1]

However, it does not address the following threats:

- Phishing
- Malware and Browser Vulnerabilities [1]

HSTS operates by supplying a response message in the HTTPS header, containing “Strict-Transport-Security” with a time period.

For example, a header message line containing:

“**Strict-Transport-Security: max-age=1000000; includeSubdomains;**” instructs the browser to operate on HTTPS only connection, for the duration of the max-age, and apply settings for subdomains as well.

Upon receiving this message, the browser acts as following:

- Browser automatically turns off any unsecured content, leaving only secured content;
- If there is a SSL error (like an untrusted certificate), the access is blocked.

Also, HSTS, by concept design, offers protection against man-in-the-middle-attacks, downgrade attacks or cookie hijacking.

2.2 Practical solutions of HSTS

The high-level use case is a combination of:

- Web browser user wishes to interact with various web sites (some arbitrary, some known) in a secure fashion.
- Web site deployer wishes to offer their site in an explicitly secure fashion for their own, as well as their users', benefit. [1]

The main problem that HSTS resolves: SSL-stripping man-in-the-middle-attacks.

The SSL Stripping attack is defined as transparently converting a secure (HTTPS) connection to a regular plain-text (HTTP) connection. HSTS resolves this by imposing a strict HTTPS only communication.

Also, another feature of HSTS is that it can resolve, by its design, the problem of stiling the login credentials of a cookie-based website.

2.3 Limitations of HSTS

One of the main problems of HSTS is that the initial request presented by a browser is performed unprotected if the request is being transmitted over an insecure channel such as HTTP. Also, the initial request, after the expiration of the max-age period is also unprotected.

Attempts to correct this problem have been made on the client side, by implementing preloaded list of HSTS sites, that can be accessed directly on HTTPS.

3. A more secure way: Server side timing for the current time problems of HSTS and DNS information delivery about HSTS

3.1 The current time problem of HSTS

The current timing of the “max-age” attribute is passed over to the browser as a recommendation, leaving it to the browser to enforce it. This approach might be problematic, because the browser (or the client's computer) can be manipulated in a harmful way. The browser calculates and applies the time in which it will communicate securely only. This could be a problem if the time of the client's machine is tampered with, resulting in an altered period of time for communicating securely.

Also, another fundamental problem is that there is no “infinite” value for the “max-age” attribute. In an ideal implementation, where there is possibility to use the “infinite” value, the HSTS protocol informs the browser that the server will communicate only on HTTPS for as long as it exists.

A lot of websites (e-banking, e-commerce, etc) will never use plain text communication (plain HTTP) ! These websites will always use HTTPS. If this is the case, it should only be natural that HSTS is available for as long as the the secure website exists.

3.2 Server-side timing

Server-side timing can be implemented for remembering each browsers' time values, regardless of a fixed time period or the “infinite” time for websites that require strict HTTPS for as long as they exist.

This paper proposes the following solution for implementing server-side timing: the identification of a browser by using a new special key that can be a checksum of the computer's IP address and the browser's User Agent information.

The server stores the browser's key, along with the time it allocates for this browser. The browser is only notified of the allocated “max-age” time, but the enforcement is made by the server.

If the browser does not obey, secure communication is terminated with the message “**HSTS violation**”.

Also, the browser may revert to normal communication (exit from HSTS) only after receiving a “**Strict-Transport-Security: hsts-stop**” signal.

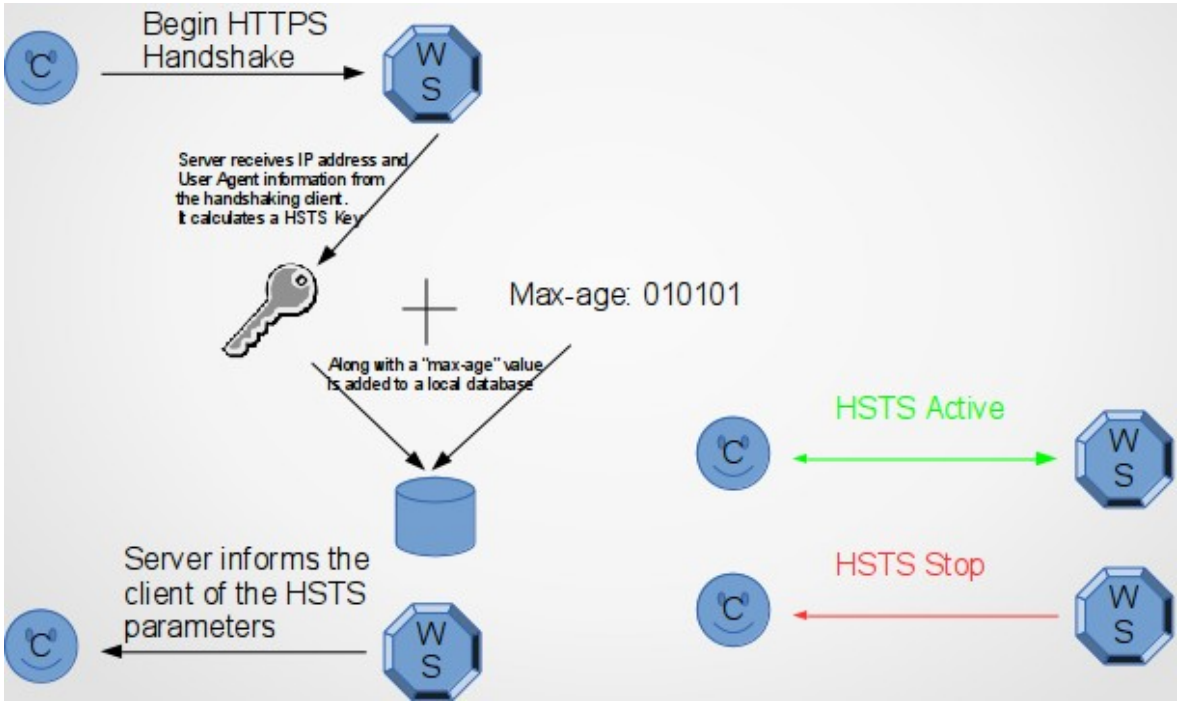

Fig. 1 – Server-side timing sequence

3.3 Informing of HSTS via DNS

The DNS system is already implemented in today's Internet. It is being used on over 90% of the world-wide-web HTTP traffic. Its infrastructure is ground-solid and well tested.

If the DNS system is so well proven, why not use the DNS system to propagate HSTS preloaded list ? It could help resolve the problem of distributing preloaded lists of HSTS sites, thus resolving the problem of the initial request delivered on a non-secure channel.

The DNS system is already used for transmitting other information, not quite related to DNS itself, like DNS-Sec, TXT Spoofing Records, etc.

With the creation of another new type of record, or attribute, like “HSTS”, we could have a DNS entry like:

“(**www.)example.com IN HSTS True**”

which could indicate to the web-browser, beside the IP address (which is the DNS' primary function), the fact that the domain / sub-domain in question is using HSTS.

The web-browser automatically initiates a HTTPS communication, based on the fact that the DNS system told it that the domain / sub-domain is capable of HSTS.

4. CONCLUSIONS

The moving to the server-side of the timing problem is, in my opinion, an important improvement to HSTS because it is more secure to have the server maintain the time rather than the client.

Also, the possibility of “infinite” time is a required feature in today's Internet. There are many websites that require HSTS “for lifetime”, like E-commerce, Internet Banking, Secure portals, etc. Websites that should never ever allow insecure elements running in their backyard.

This should require minor rewrites of web server software like Apache, IIS, nginx, etc... or adding an HSTS plugin. Also, this should require minor rewrites of web client software like Mozilla Firefox, Internet Explorer, Google Chrome, etc.

Also, the distribution of the preloaded HSTS list can be achieved more professionally by using the DNS system to inform the web client that the website uses HSTS.

GLOSSARY

1. **HTTP** – Hyper-text Transfer Protocol
2. **HTTPS** – Hyper-text Transfer Protocol Secured
3. **Web Server** – A software server that serves web pages
4. **Web Client** – A software client that requests and reads web pages.

REFERENCES

[1] - Hodges, Jeff; Jackson, Collin; Barth, Adam (November 2012). RFC 6797. IETF. URL: https://tools.ietf.org/html/rfc6797#section-5.2