# Bypassing HTTP Strict Transport Security

Jose Selvi
jselvi@pentester.es

***Abstract*—For the last few years, some different attacks against SSL/TLS have been released. Some of them based on cryptography or protocol weaknesses such as BEAST, CRIME, BREACH, etc, and some others, such as SSLStrip, based on rewriting HTTPS links into HTTP ones and keep user communications always in HTTP. In order to protect users against SSLStrip attacks, a new protection called HTTP Strict Transport Security (HSTS) has been developed and it's currently supported by most widely used browsers.**

**However, under certain circumstances, an attacker could exploit an inter-operation vulnerability in order to bypass HTTP Strict Transport Security protection and use other well-known attack techniques such as SSLStrip. In this paper, we review the HSTS strengths and weaknesses, and we go in-depth on this inter-operation vulnerability and how it could be exploited.**

## I. Brief History of Bypassing SSL

SSL/TLS is certainly one of the most important protocols in the security field since on it relies our communications privacy and security. Because of that, it's a target for both attackers and security professionals.

In the past few years some different techniques have been presented. Some of them focused on design weaknesses such as BEAST [1], CRIME [2] or BREACH [2], some other based on exploting some implementations weaknesses such as HeartBleed or the famous Apple goto fail vulnerability.

Moxie Marlinspike [4] presented one of the most used techniques, based on stripping a previous HTTP connection by rewriting all the HTTPS links into HTTP ones, changing properties of cookies and other similar changes. Since users usually type only website name in his browser and not the full URL, their fists connection would be an HTTP one, so it could be intercepted and stripped.

## II. HTTP Strict Transport Security

HTTP Strict Transport Security [5] (also known as HSTS or STS) is the industry response for the Moxie's stripping attacks and his tool SSLStrip.

HSTS protocol defines a new HTTP header called 'Strict-Transport-Security' that can be sent by a webserver to his clients in order to specify a new policy regarding how the browser in going to handle the future connections.

There are two main parameters in an HSTS policy. One of them is 'max-age' that represents the amount of seconds that the browser should connect in HTTPS-only mode. As a result, a browser that receive an HSTS policy with 'max-age:1000' from 'mywebsite.com' would stay using HTTPS if the user clicks on HTTP links or even if the user type an HTTP link. That policy would be active for the following 1000 seconds from the last HTTPS connection. After that, the policy outdates and the browser returns to his usual behavior.

An optional parameter in an HSTS is 'IncludeSubdomains'. If this parameter is set, then the HSTS policy applies to the visited domain and all the subdomains as well. If not it only applies to the exact domain that the user has visited.

In addition, an HSTS policy prevents an user from accepting self-signed or abnormally signed certificates, since remember the certification authority (CA) that signed the previous seen certificate.

Unfortunately, HSTS is not a security feature that is currently widely deployed in the Internet, since just a few websites use it. However, some reference companies such as Twitter, Paypal or Google use this security feature.

Most desktop browsers support this security feature as well. Some of them share a 'Preloaded HSTS' list [6][7] that contains a domain list of hosts that should be configured automatically even before the first HTTPS connection, so users remain protected after a fresh install or after wiping out their local state.

## III. Network Time Protocol (NTP)

The Operating Systems use the Internet for a big amount of internal tasks or features such as software updates, the OS activation itself and so on. One of those features is the Time Synchronization. By default, almost all the desktop operating systems automatically synchronize its time with Internet Servers usually owned by the operating system provider (for example 'time.windows.com' for Microsoft operating systems).

All of them use different versions (v3 or v4) of the Network Time Protocol (NTP) [8][9][10] that is widely used to provide time synchronization between computers.

NTP messages are sent via UDP packets (123/UDP). The message format is the same for both requests and responses, but each peer use a different set of fields and ignore the rest of them.

Most important fields are:

- Leap (LI): Leap is a warning indicator that should be usually set to zero. Clients often set this value to 3 (clock unsynchronized) when request time synchronization.
- Version (VN): NTPv3 (3) or NTPv4 (4).
- Mode: Usually client (3) or server (4) depending if it is a request or a response. Other values are also possible but they are not used in by default NTP configuration.

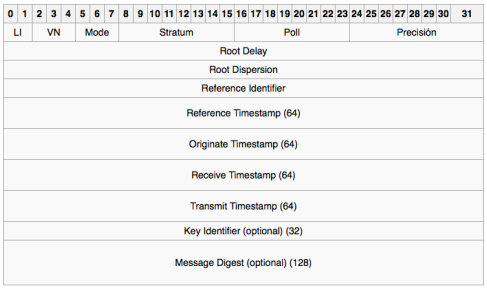


Fig. 1. NTP Packet

- Stratum: Usually from 2 to 15. Values 0 and 1 are used by reference clocks and primary servers and shouldn't be used by NTP servers.
- Precision: Usually -18 or -20 (microseconds). Precision of the system clock. Value in log2 seconds.
- Root delay & dispersion: Total round-trip and dispersion from de reference clock. Value in NTP short format.

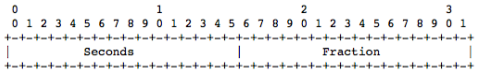


Fig. 2. NTP Short Format

- Reference identifier: Server identifier, usually his IP Address.
- Timestamps: Different values that are used by the client in order to calculate the current date and time. Values in NTP format

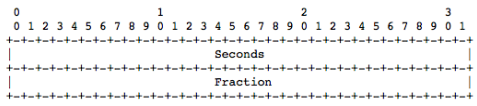


Fig. 3. NTP Format

NTPv4 supports authentication based on asymmetric cryptography. The server signs NTP messages using his own private key. As a result, clients can verify messages integrity, so Man-in-the-Middle techniques shouldn't be possible.

However, none operating system use authentication, so all of them would be vulnerable to Main-in-the-Middle attacks.

## IV. Delorean: An NTP MitM tool

In order to perform NTP Man-in-the-Middle attacks, a new tool called 'Delorean' has been developed and it is available for download at Github. His name, as you probably know, is a reference to the well-known 80's film 'Back to the future' and its time machine.

Delorean is a python script based on the kimifly's tool 'ntpserver' [11] but adding some additional options for on-the-fly manipulation.

```
$ ./delorean.py -h

Usage: delorean.py [options]

Options:
  -h, --help            show this help message and exit
  -i INTERFACE, --interface=INTERFACE
                        Listening interface
  -p PORT, --port=PORT  Listening port
  -n, --nobanner        Not show Delorean banner
  -s STEP, --force-step=STEP
                        Force the time step: 3m (minutes), 4d (days), 1M
                        (month)
  -d DATE, --force-date=DATE
                        Force the date: YYYY-MM-DD hh:mm[:ss]
  -k SKIM, --skim-step=SKIM
                        Skimming step: 3m (minutes), 4d (days), 1M (month)
  -t THRESHOLD, --skim-threshold=THRESHOLD
                        Skimming Threshold: 3m (minutes), 4d (days), 1M
                        (month)
  -r, --random-date     Use random date each time
```

Delorean can be used in five different modes:

- Automatic: If not other mode is selected, Delorean works in an automatic mode. In this mode, Delorean tries to find a date at least 1000 days in the future with the same month day and weekday than the current one. It makes harder for the user to detect that something happened on his computer clock.
- Step mode (-s): Using this mode you can choose how many seconds, hours, days, etc you want to jump to the future. The base date and time are the local date and time in the host that runs Delorean.
- Date mode (-d): Using this mode you can choose the exact date and time when you want to jump to the future.
- Random mode (-r): This mode makes Delorean to answer with different date and time on each response. Useful for testing integer overflows and other similar issues in NTP implementations.
- Skimming Attack (-k & -t): This mode makes Delorean work in a different way. It is compatible with all the previous modes but it jumps to the future in several steps (-k) instead of a single one. The flag 't' makes Delorean to jump that amount of time before the '-k' time.

Delorean can't intercept communications itself, so it should be used with other tools such as arpspoof + iptables, metasploit's fakedns, etc.

## V. Time Synchronization in major OS

Even though all major operating systems use NTP as a Time Synchronization Protocol over the Internet, they use it in a different ways. Some of them synchronized each few minutes, some others only under certain circumstances or using more complex algorithms.

### A. Ubuntu Linux

Ubuntu Linux perhaps is the most widely used desktop linux distribution.

It doesn't run a NTP daemon itself but it is configured by default to synchronized via 'ntpdate' command each time a network interface goes up. It uses unauthenticated NTPv4 messages, so it is be vulnerable to MitM attacks.

```
$ ls /etc/network/if-up.d/
000resolvconf   avahi-daemon    ntpdate     wpasupplicant
avahi-autoipd   ethtool         upstart
```

In those environments where an attacker could control the physical medium (fake AP, switch controlling, deauthentication) he could force an interface down and up. When going up, the time would be synchronized, so it could be intercepted and manipulated by Delorean.

### B. Fedora Linux

Fedora Linux perhaps is another widely used desktop linux distribution.

Unlike what happens with Ubuntu, Fedora runs a NTP daemon called 'chronyd' that synchronizes each minute. It uses unauthenticated NTPv3 messages, so it is be vulnerable to MitM attacks. The default chrony configuration use the parameter 'rtcsync' witch means that the system time is copied to the real time clock each 11 minutes.

```
# netstat -anp | grep 123
udp    0   0   0.0.0.0:123   0.0.0.0:*   540/chronyd
udp6   0   0   :::123        :::*        540/chronyd
```

Waiting up to one minute, an attacker could intercept and manipulate that communication with Delorean and control the desktop's system time. After up to 11 minutes of intercepting those messages, the new time would be applied to the host.

### C. Mac OS X Lion

Mac OS X Lion (probably all pre-Mavericks versions) runs a NTP daemon called 'ntpd' [12] that synchronizes each 9 minutes. It uses unauthenticated NTPv4 messages, so it is be vulnerable to MitM attacks.

```
09:02:18.166708 IP 192.168.1.100.123 > 17.72.148.53.123: NTPv4, Client, length 48
09:02:18.224746 IP 17.72.148.53.123 > 192.168.1.100.123: NTPv4, Server, length 48
09:11:20.059792 IP 192.168.1.100.123 > 17.72.148.53.123: NTPv4, Client, length 48
09:11:20.116683 IP 17.72.148.53.123 > 192.168.1.100.123: NTPv4, Server, length 48
09:20:17.951361 IP 192.168.1.100.123 > 17.72.148.53.123: NTPv4, Client, length 48
09:20:18.013108 IP 17.72.148.53.123 > 192.168.1.100.123: NTPv4, Server, length 48
```

Waiting up to 9 minutes, an attacker could intercept and manipulate that communication with Delorean and control the desktop's system time.

### D. Mac OS X Mavericks

Mac OS X Mavericks changed its time synchronization model. The 'ntpd' daemon [12] still exists but it sends NTP messages in a less predictable way. However, even not being predictable, you should be able to intercept at least one NTP message waiting for some minutes.

```
20:57:59.038956 IP 192.168.1.100.123 > 17.151.16.21.123: NTPv4, Client, length 48
20:57:59.247494 IP 17.151.16.21.123 > 192.168.1.100.123: NTPv4, Server, length 48
21:06:53.259078 IP 192.168.1.100.123 > 17.151.16.21.123: NTPv4, Client, length 48
21:06:53.462394 IP 17.151.16.21.123 > 192.168.1.100.123: NTPv4, Server, length 48
21:15:54.423944 IP 192.168.1.100.123 > 17.151.16.21.123: NTPv4, Client, length 48
21:15:54.629670 IP 17.151.16.21.123 > 192.168.1.100.123: NTPv4, Server, length 48
21:32:24.624282 IP 192.168.1.100.123 > 17.151.16.21.123: NTPv4, Client, length 48
21:32:24.833084 IP 17.151.16.21.123 > 192.168.1.100.123: NTPv4, Server, length 48
21:57:18.017906 IP 192.168.1.100.123 > 17.151.16.21.123: NTPv4, Client, length 48
21:57:18.211821 IP 17.151.16.21.123 > 192.168.1.100.123: NTPv4, Server, length 48
22:30:32.740008 IP 192.168.1.100.123 > 17.151.16.21.123: NTPv4, Client, length 48
22:30:32.930711 IP 17.151.16.21.123 > 192.168.1.100.123: NTPv4, Server, length 48
```

Mac OS X Mavericks synchronizes each 9 minutes, as other Mac OS Xs, but the synchronization interval increase when the computer is not being intensive using. For example, in an unattended MacBook it increases up to around 30 minutes or even more.

```
$ ps -ef | grep ntpd
0  44  1  0  8:50PM  ??  0:00.04  /usr/sbin/ntpd -c /private/etc/ntp-restrict.conf
                                   -n -g -p /var/run/ntpd.pid -f /var/db/ntp.drift
```

In Mavericks the ntpd daemon doesn't change the system time by its own. It writes the detected drift in the file '/var/db/ntp.drift'. There is also another important difference. The daemon in launched with the 'panicgate' option. It makes the ntpd daemon to accept big time adjustments (more than 1000 seconds by default) but only once.

```
$ ps -ef | grep pacemaker
0  60  1  0  8:50PM  ??  0:00.10  /usr/libexec/pacemaker -b -e 0.0001 -a 10
```

The second one daemon is called 'pacemaker' [13] and it is checking the drift file for changes each 10 seconds. It slew the clock or completely change it with the new date and time depending on the adjustment needed.

There is an exception in this process. When the user opens the 'Date & Time Preferences' the system clock automatically updates and any security restriction is consider.

### E. Microsoft Windows

Microsoft Windows is the securest NTP implementation from the major operating systems. It doesn't use authentication (in a standalone configuration) but it implements some additional security features that make more difficult a reliable exploitation.

One of them is the synchronization period. Windows by default only synchronizes once a week: Sunday at 02:00. If the computer is not running then, the synchronization is made in the next boot (if it's in the next three days).

The second security feature is the 'MaxPosPhaseCorrection' and 'MinPosPhaseCorrection' parameters that are set in the windows registry (HKEY_LOCAL_MACHINE SYSTEM\CurrentControlSet\Services\W32Time\Config). These parameters specify the maximum and minimum amount of seconds that the clock can be adjusted by the time synchronization. Any time update greater is automatically ignored. In windows desktop systems such as Windows 7 or 8, these parameters are set to 15 hours, while in servers such as Windows Server 2012 they are set to 48 hours [14].

As a result, there is a narrow attack surface in a default configured windows computer. However, there are lots of non official articles in the Internet that recommend to synchronize the time more often, maybe each hour or even each 5 minutes. If the user set up his computer in order to synchronize more often than his own MaxPosPhaseCorretion time then his computer would become vulnerable to Time Skimming attacks.

There is an exception in this process. When the user manually requests a time synchronization, when any security restriction is consider.

## VI. Time Skimming Attack

A Time Skimming Attack works in a similar way than a 'Stone Skimming' effect. Perhaps the attacker can't jump to the proper date in the future, but if he can jump a few seconds before the next time synchronization then he could reach the proper date by doing multiple jumps to the future.

```
# ./delorean.py -k 12h -t 30s
```

```
Sent to 192.168.10.31:123 - Going to the future! 2014-09-27 01:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-27 13:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-28 01:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-28 13:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-29 01:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-29 13:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-30 01:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-30 13:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-01 01:32
Sent to 192.168.10.31:123 - Going to the future! 2014-09-01 13:32
[...]
```

Some Windows or Mac OS X configurations could be vulnerable to this kind of attacks.

## VII. Browsers & Preloaded HSTS

Using the Delorean tool, under certain circumstances and configurations, could allow to manipulate the system time and force the HSTS policies to expire. However, there is an extra security feature in browsers: the preloaded HSTS. When reading its documentations it seems that preloaded hosts are enforced by default so they wouldn't be vulnerables to time manipulation attacks but the real truth is that those hosts are 'preloaded' but not 'static' on most browsers, so they would be vulnerable as well.

For example in Chrome, the enforced hosts in the 'Preloaded HSTS' list are configured with a 1000 days policy. These policies can be overwritten when the browser visits de host for the first time.

net_internals_ui.cc

```
1239 void NetInternalsMessageHandler::IOThreadImpl::OnHSTSAdd(
1240     const base::ListValue* list) {
1241   // |list| should be: [<domain to query>, <STS include subdomains>, <PKP
1242   // include subdomains>, <key pins>].
1243   std::string domain;
1244   CHECK(list->GetString(0, &domain));
1245   if (!IsStringASCII(domain)) {
1246     // Silently fail. The user will get a helpful error if they query for the
1247     // name.
1248     return;
1249   }
1250   bool sts_include_subdomains;
1251   CHECK(list->GetBoolean(1, &sts_include_subdomains));
1252   bool pkp_include_subdomains;
1253   CHECK(list->GetBoolean(2, &pkp_include_subdomains));
1254   std::string hashes_str;
1255   CHECK(list->GetString(3, &hashes_str));
1256
1257   net::TransportSecurityState* transport_security_state =
1258       GetMainContext()->transport_security_state();
1259   if (!transport_security_state)
1260     return;
1261
1262   base::Time expiry = base::Time::Now() + base::TimeDelta::FromDays(1000);
1263   net::HashValueVector hashes;
1264   if (!hashes_str.empty()) {
1265     if (!Base64StringToHashes(hashes_str, &hashes))
1266       return;
1267   }
1268
1269   transport_security_state->AddHSTS(domain, expiry, sts_include_subdomains);
1270   transport_security_state->AddHPKP(domain, expiry, pkp_include_subdomains,
```

Fig. 4. Chrome Source Code

Only one tested browser, Safari, seems to configure those preloaded hosts as a static values ('inf'/'-inf'), so hosts preloaded by Safari couldn't be attacked using these techniques.

## VIII. Conclusion

We have reviewed how the major desktop operating systems work regarding its time synchronization and we have found that, on certain systems and under certain circumstances, an NTP MitM attack is possible and it could be used in order to force HSTS policies to expire. We have developed a NTP MitM tool, called Delorean, that could be used to perform the proposed attacks.

## References


[1] J.Rizzo and T.Duong, *BEAST*, Ekoparty 2011.

[2] J.Rizzo and T.Duong, *CRIME*, Ekoparty 2012.

[3] A.Prado, N.Harris and Y.Gluck, *BREACH*, Black Hat USA 2013.

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

[5] https://tools.ietf.org/html/rfc6797

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

[7] https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_securit

[8] https://tools.ietf.org/html/rfc1308

[9] https://tools.ietf.org/html/rfc5905

[10] https://tools.ietf.org/html/rfc4330

[11] https://github.com/limifly/ntpserver

[12] https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man1/ntpd.1.html

[13] https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man8/pacemaker.8.html

[14] http://technet.microsoft.com/es-es/library/dd723684(v=ws.10).aspx

[15] http://support2.microsoft.com/kb/884776/es