



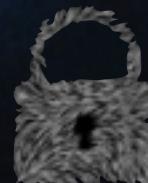
# Bypassing HTTP Strict Transport Security

Jose Selvi

Twitter: @JoseSelvi

# \$ whois jselvi

- Jose Selvi
- +10 years working in security
- Principal Penetration Tester
- SANS Institute Community Instructor
- GIAC Security Expert (GSE)
- Twitter: @JoseSelvi
- Blog: <http://www.pentester.es>



# Not a Silver Bullet



# Let's Go!

- History of Bypassing SSL
- HTTP Strict Transport Security
- HSTS Weakness
- \*\*\*\*\*
- \*\*\*\*\*
- \*\*\*\*\*

# False SSL Certificate

Client



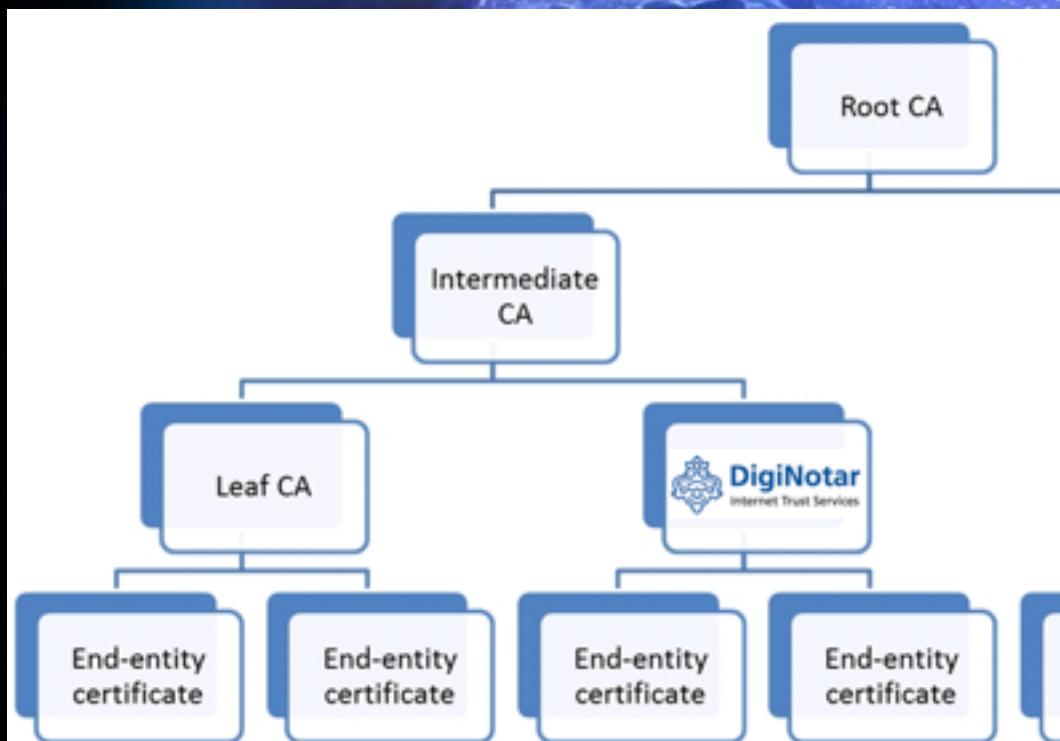
Attacker



Server



# PKI Compromise



**black hat**  
EUROPE 2014



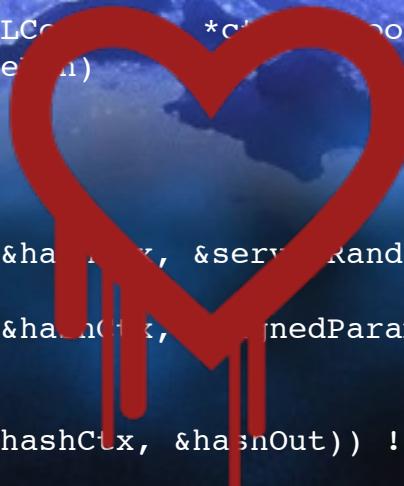
# Design weaknesses

- BEAST / CRIME
  - By Juliano Rizzo & Thai Duong
- BREACH
  - By Angel Prado, Neal Harris & Yoel Gluck
- Based on compression characteristics before encryption
- Chosen plaintext attack
- It can decrypt secrets (cookie, csrf-token, etc)

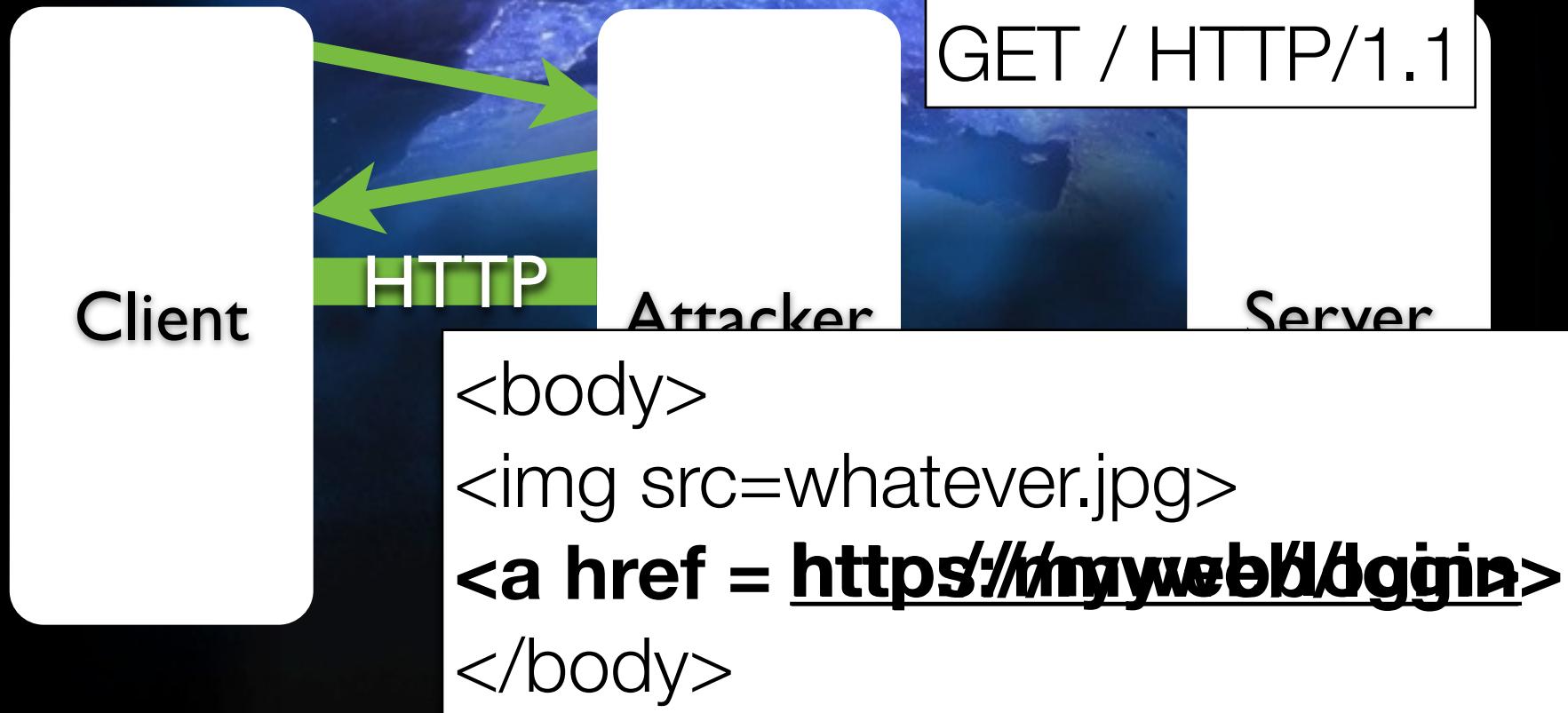
```
## Attack payload sequence initiated..  
$ ./BREACH -sniffSSL=true  
> Secret Extracted: !bb63e4b
```

# Implementation weaknesses

```
static OSStatus  
SSLVerifySignedServerKeyExchange(SSLConnectionRef *connection, const void *cert, Boolean isRsa, SSLBuffer signedParams,  
uint8_t *signature, UInt16 signatureLen)  
{  
    OSStatus err;  
    ...  
  
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)  
        goto fail;  
    if ((err = SSLHashSHA1.update(&hashCtx, signedParams)) != 0)  
        goto fail;  
    goto fail;  
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)  
        goto fail;  
    ...  
  
fail:  
    SSLFreeBuffer(&signedHashes);  
    SSLFreeBuffer(&hashCtx);  
    return err;  
}
```



# Stripping SSL Links



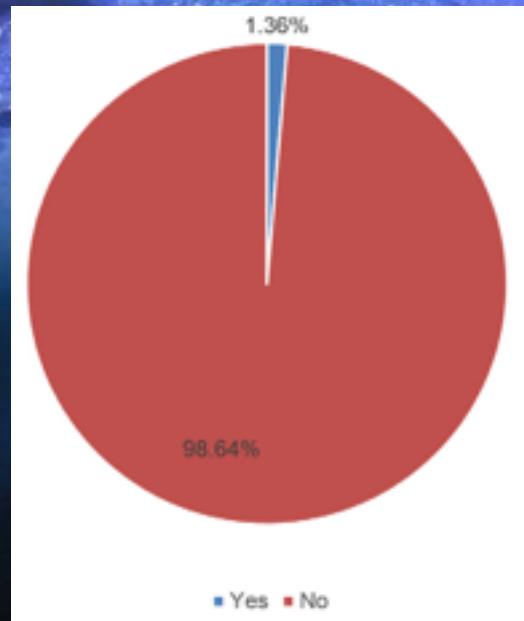
# Let's Go!

- ~~History of Bypassing SSL~~
- HTTP Strict Transport Security
- HSTS Weakness
- \*\*\*\*\*
- \*\*\*\*\*
- \*\*\*\*\*

# HTTP Strict Transport Security

- RFC-6797: Published in November 2012.
- Also known as HSTS or STS.
- Prevent HTTP connections.
- Prevent accepting self-signed and rogue certificates.
- Use a new “Strict-Transport-Security” header.

# Who uses HSTS?



<http://paul.vanbrouwershaven.com/2014/05/everyone-needs-http-strict-transport.html>

# Who uses HSTS?



# Browsers support



<http://caniuse.com/#feat=stricttransportsecurity>

# HTTPS Strict Transport Security



Strict-Transport-Security: max-age=3153600

# HTTP Strict Transport Security

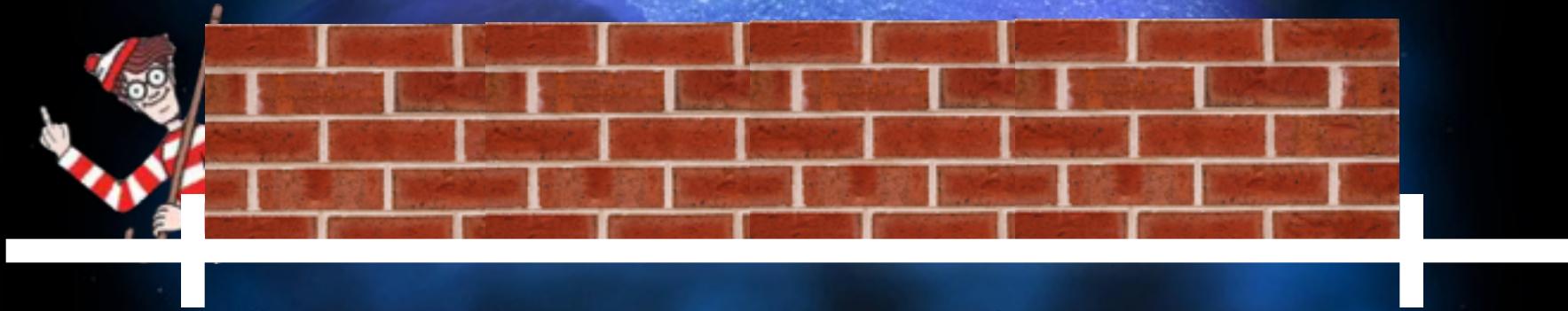
- **max-age**: number of seconds that the policy is enabled.

max-age=0 -> Delete policy

- **includeSubdomains**: If present, the policy applies all subdomains, not just the visited one.

```
$ ./hsts_catcher.py -U https://accounts.google.com
max-age=10893354; includeSubDomains
$
$ ./hsts_catcher.py -U https://paypal.com
max-age=14400
$
$ ./hsts_catcher.py -U https://github.com
max-age=31536000; includeSubdomains; preload
```

# HSTS Timeline



HTTPS  
connection

3153600  
secs later

# Preloaded HSTS

- Harcoded list of well known website names that should use always HTTPS.
- Prevent the security gap before the first HTTPS connection.
- Google, Twitter, Paypal, ...



# Let's Go!

- ~~History of Bypassing SSL~~
- ~~HTTP Strict Transport Security~~
- HSTS Weakness
- \*\*\*\*\*
- \*\*\*\*\*
- \*\*\*\*\*

# Too short max-age

🔒 Response from https://www.paypal.com:443/es/cgi-bin/webscr?cmd=\_home&country\_lang.x=true [23.39.94.246]

Forward Drop Intercept is on Action Comment this item  ?

Raw Headers Hex

HTTP/1.1 301 Moved Permanently

Server: Apache

Cache-Control: private

Pragma: no-cache

Expires: Thu, 05 Jan 1995 22:00:00 GMT

X-Frame-Options: SAMEORIGIN

**Strict-Transport-Security: max-age=14400**

Location: https://www.paypal.com/es/webapps/mpp/home

Strict-Transport-Security: max-age=14400

Content-Type: text/html; charset=UTF-8

DC: slc-a-origin-www-2.paypal.com

Date: Sun, 14 Sep 2014 15:26:01 GMT

Connection: keep-alive

Vary: Accept-Encoding

Connection: Transfer-Encoding



4 hours

?

<

+

>

Type a search term

0 matches

# Looking for weaknesses

# HSTS Pipeline



# Preloaded HSTS - Google

There is still a window where a user who has a fresh install, or who wipes out their local state, is vulnerable. Because of that, Chrome and Firefox share a "Preloaded HSTS" list. These domains will be configured for HSTS out of the box.

If you own a site that you would like to see included in the preloaded HSTS list you can submit it at <https://hstspreload.appspot.com>.

A selected subset of the members of the preloaded HSTS list:

- Google
- Paypal
- Twitter
- Simple
- Linode
- Stripe
- Lastpass

Check the source for the [full list](#).

<http://www.chromium.org/sts>

# Preloaded HSTS - Firefox

However, when connecting to an HSTS host for the first time, the browser won't know whether or not to use a secure connection, because it has never received an HSTS header from that host. Consequently, an active network attacker could prevent the browser from ever connecting securely (and even worse, the user may never realize something is amiss). To mitigate this attack, we have added to Firefox a list of hosts that want HSTS enforced by default. When a user connects to one of these hosts for the first time, the browser will know that it must use a secure connection. If a network attacker prevents secure connections to the server, the browser will not attempt to connect over an insecure protocol, thus maintaining the user's security.

<https://blog.mozilla.org/security/2012/11/01/preloading-hsts/>

# Chromium Source Code

net\_internals\_ui.cc

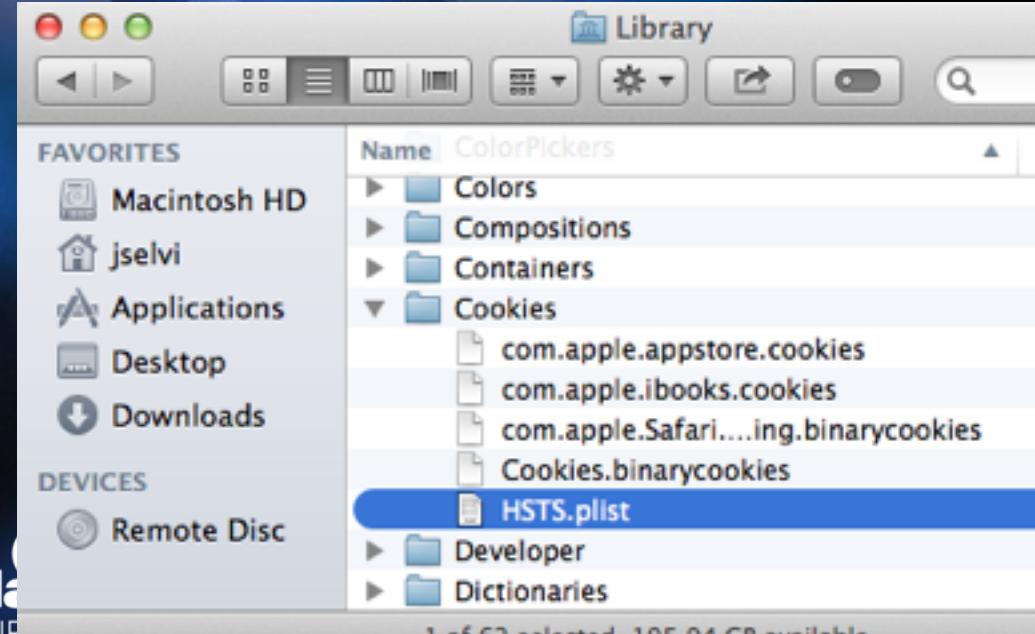
Layers ▾ Find ▾

```
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::unique_ptr<TransportSecurityState> transport_security_state =
1244         GetMainContext()->transport_security_state();
1245     if (!transport_security_state)
1246         return;
1247     // ...
1248     base::Time expiry = base::Time::Now() + base::TimeDelta::FromDays(1000);
1249 }
1250 bool CheckIfHSTSIncludedSubdomains(base::StringPiece domain,
1251                                   const base::ListValue* list) {
1252     std::unique_ptr<TransportSecurityState> transport_security_state =
1253         GetMainContext()->transport_security_state();
1254     if (!transport_security_state->AddHSTS(domain, expiry, sts_include_subdomains));
1255 }
```

# Safari PList

```
$ plutil -p HSTS.plist
```

```
{  
"com.apple.CFNetwork.defaultStorageSession" => {  
    "ssl.google-analytics.com" => -inf  
    "webmail.mayfirst.org" => -inf  
    "braintreegateway.com" => -inf  
    "code.google.com" => -inf  
    "dm.mylookout.com" => inf  
    "therapynotes.com" => inf  
    "chrome.google.com" => -inf  
    "sol.io" => -inf  
    "www.sandbox.mydigipass.com" => inf  
    [...]}
```





# DEMO

# HSTS Weakness

- Its security relies on time.
- It completely trust the OS's current time.
- Is it trustable?
- Is it possible to change the system time from the network?



# Let's Go!

- ~~History of Bypassing SSL~~
- ~~HTTP Strict Transport Security~~
- ~~HSTS Weakness~~
- Network Time Protocol (NTP)
- Get in a Delorean
- OS Time Synchronisation & Browsers

# Network Time Protocol (NTP)

- Time Synchronisation Services.
- RFC-1305 (v3) / RFC-5905 (v4) / RFC-4330 (SNTPv4).
- Set up by default on most (or all) Operating Systems.
- Security features (v4) NOT used by default.
- Vulnerable to Man-in-the-Middle techniques.

# Network Time Protocol (NTP)



# NTP Packet (I)

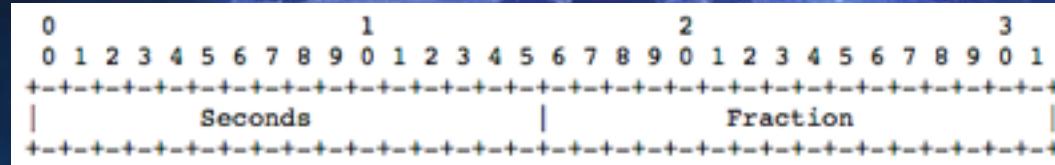
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LI	VN	Mode	Stratum					Poll					Precisión																		
Root Delay					Root Dispersion					Reference Identifier					Reference Timestamp (64)					Originate Timestamp (64)					Receive Timestamp (64)						
Transmit Timestamp (64)					Key Identifier (optional) (32)					Message Digest (optional) (128)																					

# NTP Packet (II)

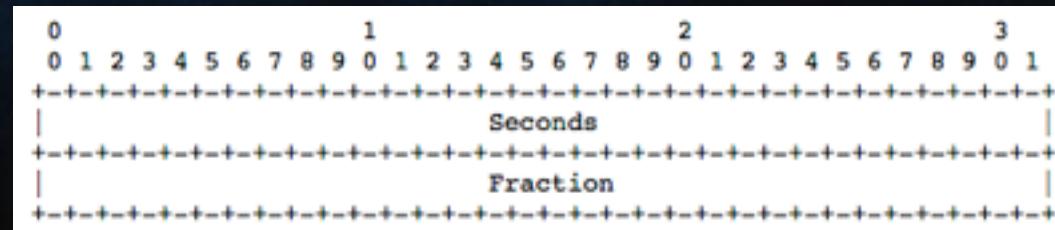
- Leap: 3 -> Clock not synchronised.
- Version: 3 /4
- Mode: Client (3) / Server (4) / etc.
- Stratum: Usually 2-15.
- Poll: NTP polling interval.
- Precision: Usually -20 ( $2^{-20}$ ) -> Microseconds.

# NTP Packet (III)

- Root delay & dispersion: NTP short format.



- Reference ID: 4 octets IPv4.
- Timestamps: NTP timestamp format



# Example: Ubuntu Linux

```
▽ Network Time Protocol (NTP Version 4, client)
  ▽ Flags: 0xe3
    11... .... = Leap Indicator: unknown (clock unsynchronized) (3)
    ..10 0... = Version number: NTP Version 4 (4)
    .... .011 = Mode: client
    Peer Clock Stratum: unspecified
    Peer Polling Interval: invalid
    Peer Clock Precision: 0.01
    Root Delay: 1.0000 sec
    Root Dispersion: 1.0000
    Reference ID: NULL
    Reference Timestamp: Jan
    Origin Timestamp: Jan 1,
    Receive Timestamp: Jan 1,
    Transmit Timestamp: Sep 3
  ▽ Network Time Protocol (NTP Version 4, server)
    ▽ Flags: 0x24
      00.. .... = Leap Indicator: no warning (0)
      ..10 0... = Version number: NTP Version 4 (4)
      .... .100 = Mode: server (4)
      Peer Clock Stratum: secondary reference (2)
      Peer Polling Interval: invalid (3)
      Peer Clock Precision: 0.000001 sec
      Root Delay: 0.0099 sec
      Root Dispersion: 0.0239 sec
      Reference ID: 192.93.2.20
      Reference Timestamp: Sep 3, 2014 08:36:01.601928000 UTC
      Origin Timestamp: Sep 3, 2014 08:40:04.634295000 UTC
      Receive Timestamp: Sep 3, 2014 08:40:04.653302000 UTC
      Transmit Timestamp: Sep 3, 2014 08:40:04.653354000 UTC
```

# NTP Man-in-the-Middle

Oct 21:2015 07:28

VICTIM

It's 11:00

Actually It's  
Oct 21 2015 07:28

FAKE  
NTP

NTP



# Let's Go!

- ~~History of Bypassing SSL~~
- ~~HTTP Strict Transport Security~~
- ~~HSTS Weakness~~
- ~~Network Time Protocol (NTP)~~
- Get in a Delorean
- OS Time Synchronisation & Browsers

# Delorean

- NTP MitM Tool. Free. Open Source. Python.
  - <http://github.com/PentesterES/Delorean>
- Inspired on a kimifly's work:
  - <http://github.com/lmifly/ntpserver>
- Implements some attacks.
- Pretend to become an NTP attack suite.

# Delorean

```
$ ./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]
- r, --random-date Use random date each time



# DEMO

DEMO

[w] : / \ : | : / \ : [w] :  
V \_\_\_\_\_| \ / | : | : / \ | : -"  
V \_\_\_\_\_| \ / | : | : / \ | : -"

# Let's Go!

- ~~History of Bypassing SSL~~
- ~~HTTP Strict Transport Security~~
- ~~HSTS Weakness~~
- ~~Network Time Protocol (NTP)~~
- ~~Get in a Delorean~~
- OS Time Synchronisation & Browsers

# Ubuntu Linux

- Really simple.
- NTPv4. No authentication.
- At each network reconnection (& boot time).



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

# Fedora Linux

- The simplest one.
- NTPv3.
- More than one NTP server.
- EACH minute!



```
$ tcpdump -i eth0 -nn src port 123
12:43:50.614191 IP 192.168.1.101.123 > 89.248.106.98.123: NTPv3, Client, length 48
12:44:55.696390 IP 192.168.1.101.123 > 213.194.159.3.123: NTPv3, Client, length 48
12:45:59.034059 IP 192.168.1.101.123 > 89.248.106.98.123: NTPv3, Client, length 48
```

# Mac OS X - Lion

- Pretty simple as well.
- NTPv4. No authentication.
- Each 9 minutes.



```
$ tcpdump -i eth0 -nn src port 123
09:02:18.166708 IP 192.168.1.100.123 > 17.72.148.53.123: NTPv4, Client, length 48
09:11:20.059792 IP 192.168.1.100.123 > 17.72.148.53.123: NTPv4, Client, length 48
09:20:17.951361 IP 192.168.1.100.123 > 17.72.148.53.123: NTPv4, Client, length 48
```

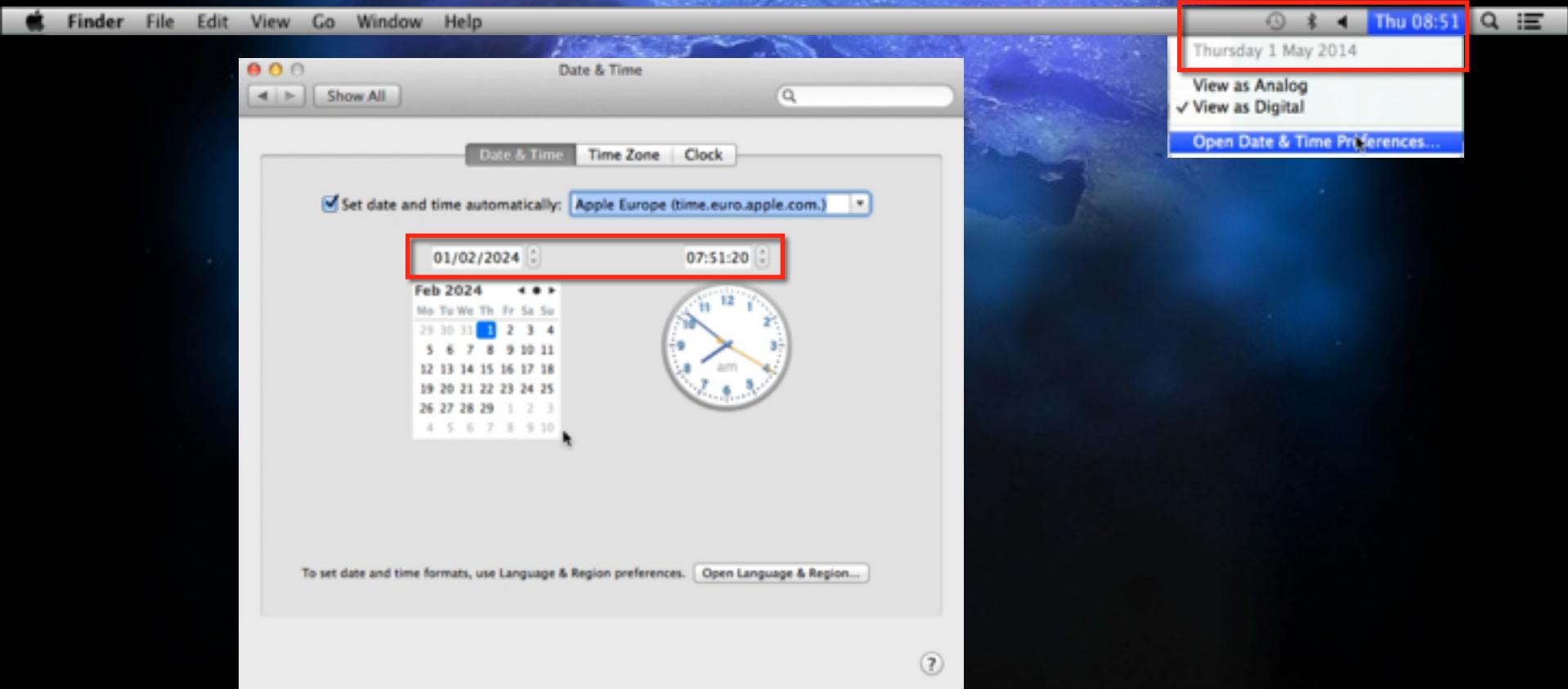
# Mac OS X - Mavericks

- New synchronisation service.
- NTP still exists but not synchronises.
  - Just write in /var/db/ntp.drift
- A new service called “**pacemaker**” check this file and synchronise the system clock.
- It seems it doesn’t work as expected...



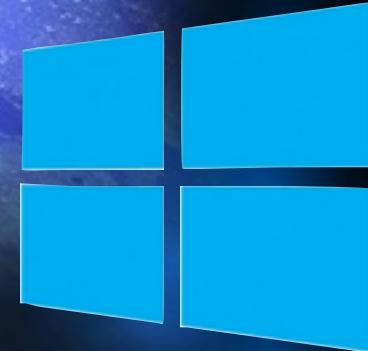
<http://www.atmythoughts.com/living-in-a-tech-family-blog/2014/2/28/what-time-is-it>

# Mac OS X - Mavericks



# Windows

- NTPv3 but...
- The securest one.
- Synchronization each 7 days.
- Doesn't accept more than 15 hours increment/decrement.
- Domain members have a different set up.



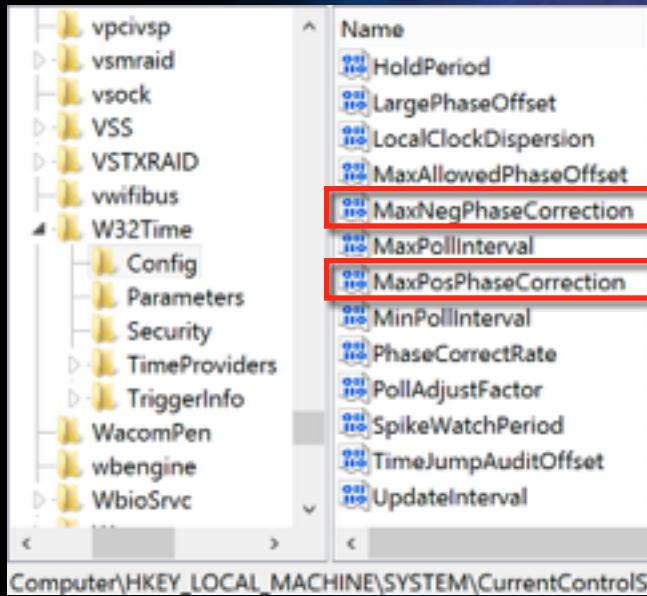
# W32time Service

The screenshot shows the Windows Task Scheduler interface. On the left, a tree view lists various system services as tasks. In the center, a detailed view of the 'ForceSynchronizeTime' task is displayed. The task is set to run at regular intervals (every 7 days) and uses a custom trigger. The XML configuration for the task is shown below, with several key elements highlighted by red boxes:

```
<Task version="1">
  <Trigger>
    <Repetition>
      <Period>P7D</Period>
    </Repetition>
    <Custom>
      <Time>19:00</Time>
    </Custom>
  </Trigger>
  <Actions Context="LocalService">
    <Exec>
      <Command>%windir%\system32\sc.exe</Command>
      <Arguments>start w32time task_started</Arguments>
    </Exec>
  </Actions>
</Task>
```

The highlighted sections include the Period of 7 days, the Arguments 'start w32time task\_started', and the Command '%windir%\system32\sc.exe'.

# Max[Pos | Neg]PhaseCorrection



W7 / W8  
15 hours

Name	Type	Data
(Default)	REG_SZ	(value not set)
AnnounceFlags	REG_DWORD	0x0000000a (10)
EventLogFlags	REG_DWORD	0x00000002 (2)
FrequencyCorrectRate	REG_DWORD	0x00000004 (4)
HoldPeriod	REG_DWORD	0x00000005 (5)
LargePhaseOffset	REG_DWORD	0x02faf080 (50000000)
LocalClockDispersion	REG_DWORD	0x0000000a (10)
MaxAllowedPhaseOffset	REG_DWORD	0x0000012c (300)
MaxNegPhaseCorrection	REG_DWORD	0x0002a300 (172800)
MaxPollInterval	REG_DWORD	0x0000000a (10)
MaxPosPhaseCorrection	REG_DWORD	0x0002a300 (172800)
MinPollInterval	REG_DWORD	0x00000006 (6)
PhaseCorrectRate	REG_DWORD	0x00000007 (7)
PollAdjustFactor	REG_DWORD	0x00000005 (5)
SpikeWatchPeriod	REG_DWORD	0x00000384 (900)
TimeJumpAuditOffset	REG_DWORD	0x00007080 (28800)
UpdateInterval	REG_DWORD	0x00000064 (100)

W2K12 48 hours

# Time Skimming Attack

H  
Time Syr



H  
153600  
ecs later

# Force Synchronisation

Date and Time

Date and Time Additional Clocks Internet Time

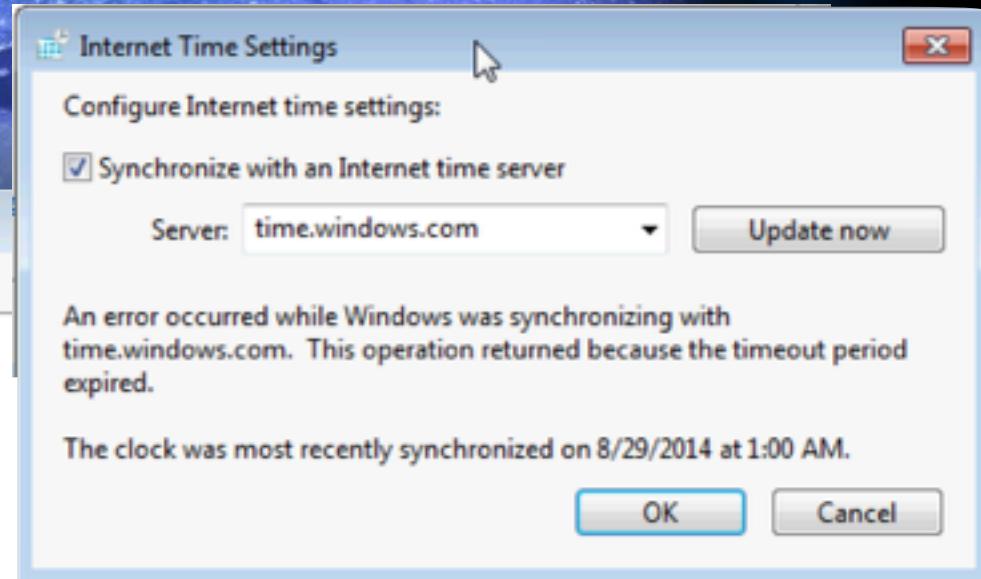
This computer is set to automatically synchronize with 'time.windows.com'.

Next synchronization: 8/31/2014 at 1:00 AM

An error occurred while Windows was synchronizing with time.windows.com. This operation returned because the timeout period expired.

The clock was most recently synchronized on 8/29/2014 at 1:00 AM.

 Change settings...





# DEMO

# Let's Go!

- ~~History of Bypassing SSL~~
- ~~HTTP Strict Transport Security~~
- ~~HSTS Weakness~~
- ~~Network Time Protocol (NTP)~~
- ~~Get in a Delorean~~
- ~~OS Time Synchronisation & Browsers~~

# Whose fault is?



# Answer from Google

#1 [REDACTED] @chromium.org

Yesterday (17 hours ago)

This is a pretty cool DoS.

[REDACTED] @ do you want to look at this?

I imagine we need some pre-established channel to establish the time. I am guessing this might be available already through Omaha or similar but I'm not sure.

Status: Available

Cc: [REDACTED] @chromium.org [REDACTED] @gmail.com

Labels: Security\_Impact-Stable Security\_Impact-Beta Security\_Severity-Low

#2 jse...@gmail.com

Today (16 hours ago) [Delete comment](#)

In fact, If you force hsts entries to expire, you can use ssl-stripping attacks and capture credentials and other information. DoS is possible as well, but I think the credential capture is the most critical attack surface.

#3 [REDACTED] @chromium.org

Today (19 minutes ago)

Thanks for the report, but this is a fairly well known, generic issue with relying on unauthenticated NTP and is not specific to HSTS. Consider that SSL/TLS, Kerberos, and MS ActiveDirectory all have the same underlying concerns and attack profile. The typical solutions for this are to use authenticated NTP, use multiple NTP servers with a client that detects excessive drift or "bad tickers", or to use some other mechanism to provide authenticated time.

Status: Won'tFix

# References

- [https://www.owasp.org/index.php/HTTP\\_Strict\\_Transport\\_Security](https://www.owasp.org/index.php/HTTP_Strict_Transport_Security)
- <https://tools.ietf.org/html/rfc6797>
- <http://dev.chromium.org/sts>
- [https://developer.mozilla.org/en-US/docs/Web/Security/HTTP\\_strict\\_transport\\_security](https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security)
- <http://www.ntp.org>
- <https://github.com/limifly/ntpserver>
- <http://www.thoughtcrime.org/software/sslstrip/>
- <https://github.com/LeonardoNve/dns2proxy>

# Thanks! Questions?

**Jose Selvi**

<http://twitter.com/JoseSelvi>

[jselvi@incide.es](mailto:jselvi@incide.es)

<http://www.incide.es>

[jselvi@pentester.es](mailto:jselvi@pentester.es)

<http://www.pentester.es>

