Misusing TCP Timestamps

Veit Hailperin
Offense Department, scip AG
veha@scip.ch
https://www.scip.ch

Marc Ruef (Editor)
Research Department, scip AG
maru@scip.ch
https://www.scip.ch

Keywords: Block, Detect, Exploit, Fingerprinting, Firewall, GitHub, Google, IIS, Linux, Master

1. Preface

This paper was written in 2015 as part of a research project at scip AG, Switzerland. It was initially published online at *https://www.scip.ch/en/?labs.20150305* and is available in English and German. Providing our clients with innovative research for the information technology of the future is an essential part of our company culture.

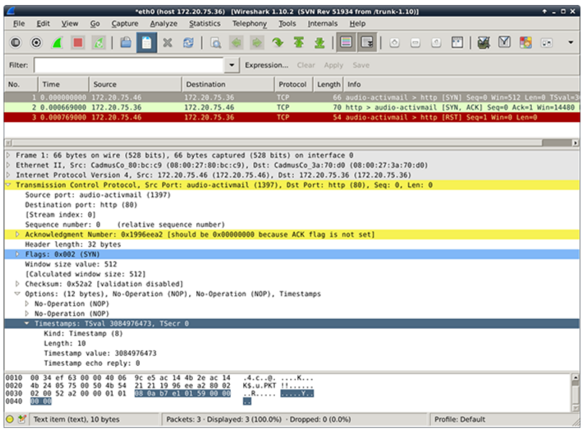
2. Introduction

Before any attack is launched an attacker tries to get as much information about the target as possible. It will be done by passive information gathering like *Google dorking* [1] as well as active information gathering, like *port scanning* [2]. Information gathering using TCP timestamps is considered active information gathering, since it interacts directly with the systems. First we will take a closer look at what TCP timestamps are and how they work. Then we will discuss ways how TCP timestamps have been used, to then move on to how networks have changed and how that has introduced new challenges to information gathering. Last we will introduce a not previously published way to utilize TCP timestamps and also how to mitigate these attacks.

3. TCP timestamps

If at any point in the article it states just "timestamp" then it refers to TCP timestamps. TCP timestamps as defined in *RFC 1323* [3] are an extension of the original TCP Stack. They were introduced to protect against wrapped sequence numbers and to improve Round Trip-Time Measurement. The timestamp echo reply is sent in any ACK or data segment.

> *Quote RFC 1323:*
>
> *The Timestamps option carries two four-byte timestamp fields. The Timestamp Value field (TSval) contains the current value of the timestamp clock of the TCP sending the option.*
>
> *The Timestamp Echo Reply field (TSecr) is only valid if the ACK bit is set in the TCP header; if it is valid, it echos a timestamp value that was sent by the remote TCP in the TSval field of a Timestamps option.*


*eth0 (host 172.20.75.36) [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save

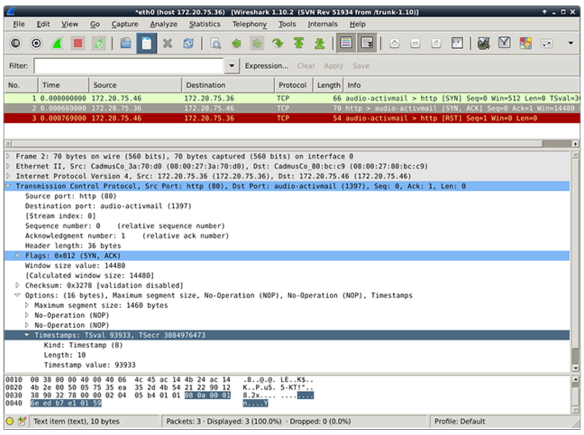
```
No.  Time         Source        Destination   Protocol Length Info
   1 0.000000000  172.20.75.46  172.20.75.36  TCP      66 audio-activmail > http [SYN] Seq=0 Win=512 Len=0 TSval=3
   2 0.000669000  172.20.75.36  172.20.75.46  TCP      70 http > audio-activmail [SYN, ACK] Seq=0 Ack=1 Win=14480
   3 0.000769000  172.20.75.46  172.20.75.36  TCP      54 audio-activmail > http [RST] Seq=1 Win=0 Len=0
```

```
Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: CadmusCo_80:bc:c9 (08:00:27:80:bc:c9), Dst: CadmusCo_3a:70:d0 (08:00:27:3a:70:d0)
Internet Protocol Version 4, Src: 172.20.75.46 (172.20.75.46), Dst: 172.20.75.36 (172.20.75.36)
Transmission Control Protocol, Src Port: audio-activmail (1397), Dst Port: http (80), Seq: 0, Len: 0
  Source port: audio-activmail (1397)
  Destination port: http (80)
  [Stream index: 0]
  Sequence number: 0    (relative sequence number)
  Acknowledgment Number: 0x1996eea2 [should be 0x00000000 because ACK flag is not set]
  Header length: 32 bytes
  Flags: 0x002 (SYN)
  Window size value: 512
  [Calculated window size: 512]
  Checksum: 0x52a2 [validation disabled]
  Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    No-Operation (NOP)
    No-Operation (NOP)
    Timestamps: TSval 3084976473, TSecr 0
      Kind: Timestamp (8)
      Length: 10
      Timestamp value: 3084976473
      Timestamp echo reply: 0
```

```
0010  00 34 ef 63 00 00 40 06  9c e5 ac 14 4b 2e ac 14   .4.c..@. ....K...
0020  4b 24 05 75 00 50 4b 54  21 21 19 96 ee a2 80 02   K$.u.PKT !!......
0030  02 00 52 a2 00 00 01 01  08 0a b7 e1 01 59 00 00   ..R..... .....Y..
0040  00 00                                              ..
```

```
Text item (text), 10 bytes   Packets: 3 · Displayed: 3 (100.0%) · Dropped: 0 (0.0%)   Profile: Default
```

Figure: TSOPTS Request


*eth0 (host 172.20.75.36) [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save

```
No.  Time         Source        Destination   Protocol Length Info
   1 0.000000000  172.20.75.46  172.20.75.36  TCP      66 audio-activmail > http [SYN] Seq=0 Win=512 Len=0 TSval=3
   2 0.000669000  172.20.75.36  172.20.75.46  TCP      70 http > audio-activmail [SYN, ACK] Seq=0 Ack=1 Win=14480
   3 0.000769000  172.20.75.46  172.20.75.36  TCP      54 audio-activmail > http [RST] Seq=1 Win=0 Len=0
```

```
Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
Ethernet II, Src: CadmusCo_3a:70:d0 (08:00:27:3a:70:d0), Dst: CadmusCo_80:bc:c9 (08:00:27:80:bc:c9)
Internet Protocol Version 4, Src: 172.20.75.36 (172.20.75.36), Dst: 172.20.75.46 (172.20.75.46)
Transmission Control Protocol, Src Port: http (80), Dst Port: audio-activmail (1397), Seq: 0, Ack: 1, Len: 0
  Source port: http (80)
  Destination port: audio-activmail (1397)
  [Stream index: 0]
  Sequence number: 0    (relative sequence number)
  Acknowledgment number: 1    (relative ack number)
  Header length: 36 bytes
  Flags: 0x012 (SYN, ACK)
  Window size value: 14480
  [Calculated window size: 14480]
  Checksum: 0x3278 [validation disabled]
  Options: (16 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), Timestamps
    Maximum segment size: 1460 bytes
    No-Operation (NOP)
    No-Operation (NOP)
    Timestamps: TSval 93933, TSecr 3084976473
      Kind: Timestamp (8)
      Length: 10
      Timestamp value: 93933
```

```
0010  00 38 00 00 40 00 40 06  4c 45 ac 14 4b 24 ac 14   .8..@.@. LE..K$..
0020  4b 2e 00 50 05 75 35 ea  35 2d 4b 54 21 22 90 12   K..P.u5. 5-KT!"..
0030  38 90 32 78 00 00 02 04  05 b4 01 01 08 0a 00 01   8.2x.... ........
0040  6e ed b7 e1 01 59                                  n....Y
```

```
Text item (text), 10 bytes   Packets: 3 · Displayed: 3 (100.0%) · Dropped: 0 (0.0%)   Profile: Default
```

Figure: TSOPTS Response

4. Information Gathering

There are two main ways TCP timestamps have been used until today to help during attacks on systems:

1. Uptime calculation
2. Host identification using *clock skew* [4]

Most systems start with timestamp `0` when they are booted. From multiple timestamps it is possible to infer the frequency. Frequency and the actual timestamp then can be used to calculate uptime. The tool *hping3* does all of this for you.

```
# hping3 -c 4 -S -p <open port> --tcp-
timestamp <ip>
```

The output will give you the actual timestamp, the frequency and the uptime:

```
len=56 ip=172.20.75.36 ttl=64 DF id=0
sport=80 flags=SA seq=0 win=14480 rtt=1.3 ms
  TCP timestamp: tcpts=500722

len=56   ip=172.20.75.36   ttl=64   DF   id=0
sport=80 flags=SA seq=1 win=14480 rtt=1.1 ms
  TCP timestamp: tcpts=500972
  HZ seems hz=100
   System uptime seems: 0 days, 1 hours, 23
minutes, 29 seconds

len=56   ip=172.20.75.36   ttl=64   DF   id=0
sport=80 flags=SA seq=2 win=14480 rtt=1.0 ms
  TCP timestamp: tcpts=501222
  HZ seems hz=100
   System uptime seems: 0 days, 1 hours, 23
minutes, 32 seconds

len=56   ip=172.20.75.36   ttl=64   DF   id=0
sport=80 flags=SA seq=3 win=14480 rtt=1.0 ms
  TCP timestamp: tcpts=501472
  HZ seems hz=100
   System uptime seems: 0 days, 1 hours, 23
minutes, 34 seconds
```

Knowing the uptime can be utilized in various ways.

- It can confirm if a DoS-Attack has been successful, by comparing uptime before and after.
- It can also help you determine the patch level of the system.

If you have fingerprinted the system properly you have a good chance of knowing if this system has to be rebooted for applying patches. Keep in mind that it is not a requirement of TCP timestamps to start at `0` at boot time. It is theoretically possible to set a random start value.

5. Network Layout Information Gathering

Systems used to often connect directly to the internet. Nowadays systems usually connect to the internet through firewalls. Firewalls often use *Network Address Translation* [5] , short NAT. NAT translates an IP address from one network to an IP address of another network. For example: the internal server DMZ to the Internet. This IP is then used throughout the internet. But firewalls have more features, one of which is that different servers can have one outgoing IP address, by using different ports.

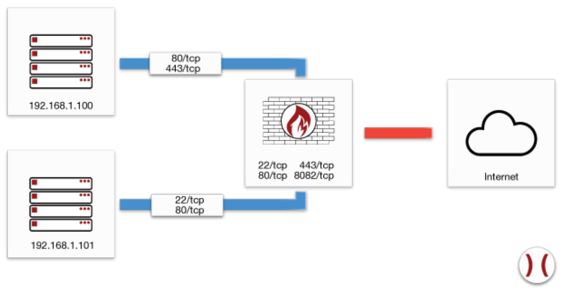


Figure: Example Network

[6]

Network mapping has been a topic of research that was limited by the scenario above. My research has overcome this problem using the attacks described below.

First, let's gather some information about the network layout despite the firewall. We will send valid TSOpts to the server and there are three possible cases, if we've received TSval in the `<SYN,ACK>` packet on two different ports on the same IP:

1. Timestamps are different
2. Timestamps are the same
3. At least one port doesn't answer with set TSopts

Since timestamps are continuously increasing (strong monotony is required from the clock) and timestamps are incremented somewhere in the range of milliseconds, it is highly unlikely that two systems are having a timestamp difference of less than a second, so we will call it "same". Knowing that we can determine which ports of an externally facing IP belong to the same system behind the firewall. Let us look at each case:

1. It is very likely that timestamps are coming from different systems
2. It is likely that timestamps are coming from the same system
3. If only one port responds without set TCP timestamp options, it is safe to assume that two different systems are responding. If TSopts are not included in the answer or TSval = 0 on both ports, then no knowledge can be gained, because it could be the same system having timestamps disabled or timestamps got disabled on all systems.

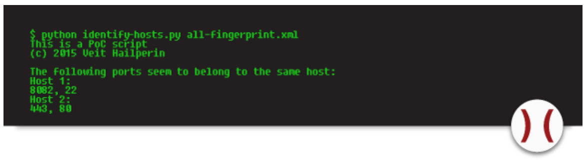
This information gives us an idea of how the target structures their DMZ.

Second, one bit of information that is helpful if not necessary before attacking is to know the running operating system of a server. For that a technique called *fingerprinting* [7] is used. Just like a human fingerprint, a computer has a fingerprint based on the running services, TTL, the way a server responds to a closed port during port scanning, etc.

For example: A server is running Microsoft IIS. It has an initial TTL guess of 80. Therefore it is likely a Microsoft Windows.

Fingerprinting functionalities, as used in port scanners like *nmap* [8], often assume that the IP being scanned is one system only. The problem, depicted in scenario in figure 1 ,is that multiple systems behind a firewall are NAT-ed to one IP address and thus create an unusable fingerprint. As shown, we can determine if one or multiple systems are NAT-ed through one IP. Using only ports that belong to one system, will likely eliminate closed ports as sources, but will remove the noise from having other systems in the fingerprint, und thus result in a clearer fingerprint.

There is a *Proof-of-Concept (PoC)-Script* [9] that reads the tcp-timestamp and based on that returns if there are services running on different servers.



```
$ python identify-hosts.py all-fingerprint.xml
This is a PoC script
(c) 2015 Veit Hailperin

The following ports seem to belong to the same host:
Host 1:
8082, 22
Host 2:
443, 80
```

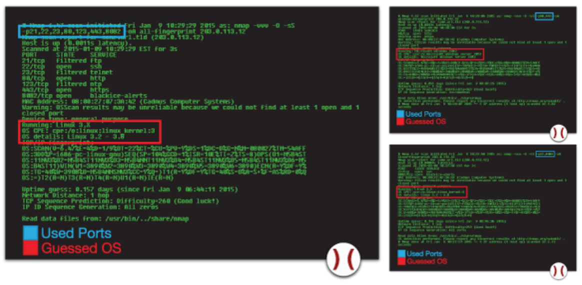
Figure: PoC-Script in Action

Known problems are that some firewalls end the tcp connection and then build a new one – the timestamp is then of the firewall or reverse-proxy and thus it is impossible to find out more about the network behind the firewall using timestamps. Another known problem: If multiple servers are run virtually on the same machine, they can have almost same timestamps.

5.1. Proof of concept:

The setup is a *monowall* [10] with one external facing IP address, having port 80 and 443 binding to a Windows 2012 R2 Server with IIS 8.5 and port 22 and 8082 binding to a Debian running SSH and Apache.

The first screenshot shows a garbled, useless fingerprint. This occurs when all ports are used for fingerprinting. The other two show fingerprints, one for each system that we were able to identify using TCP timestamps.



```
# Nmap 6.47 scan initiated Fri Jan  9 10:29:29 2015 as: nmap -vvv -O -sS
-p21,22,23,80,123,443,8082 -oA all-fingerprint 203.0.113.12
Nmap scan report for [illegible]1.tld (203.0.113.12)
Host is up (0.0011s latency).
Scanned at 2015-01-09 10:29:29 EST for 3s
PORT     STATE    SERVICE
21/tcp   filtered ftp
22/tcp   open     ssh
23/tcp   filtered telnet
80/tcp   open     http
123/tcp  filtered ntp
443/tcp  open     https
8082/tcp open     blackice-alerts
MAC Address: 08:00:27:07:3A:42 (Cadmus Computer Systems)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1
closed port
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.2 - 3.8
TCP/IP fingerprint:
OS:SCAN(V=6.47%E=4%D=1/9%OT=22%CT=%CU=%PV=Y%DS=1%DC=D%G=N%M=080027%TM=54AFF
OS:3DD%P=i686-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=10E%TI=Z%TS=8)OPS(O1=M5B4ST
OS:11NW3%O2=M5B4ST11NW3%O3=M5B4NNT11NW3%O4=M5B4ST11NW3%O5=M5B4ST11NW3%O6=M5
OS:B4ST11)WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6=3890)ECN(R=Y%DF=Y%
OS:T=40%W=3908%O=M5B4NNSNW3%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q
OS:=)T2(R=N)T3(R=N)T4(R=N)U1(R=N)IE(R=N)

Uptime guess: 0.157 days (since Fri Jan  9 06:44:11 2015)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=260 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/bin/../share/nmap
```

Figure: Several Fingerprints

[11]

As side note: Don't let nmap fool you – a closed port is a good idea when fingerprinting a host directly. This is not the case anymore when the device is behind a firewall.

A special way of network layout information gathering is *load-balance checking* for a service. While there are several ways of figuring it out, TCP timestamps offer us another way to access that information. As seen in multiple systems on multiple ports, if there are multiple timestamps on one port, the service is likely to be active-active load-balanced.

6. Mitigation

Currently the best way to defend against all possible information gathering done using TCP timestamps is disabling them, which comes with the cost of lost Protection Against Wrapped Sequence numbers (PAWS) and less good Round-Trip Time Measurement (RTTM).

Disabling timestamps can be done under Linux using:

```
    # echo 0 > /proc/sys/net/ipv4/tcp_timestamps
```

On Windows servers they can be disabled by setting:

```
    Tcp1323Opts = 0
```

A proper solution to the above described problems does not currently exist.

- **Update May 8th, 2015**: Update chapter *Network Layout Information Gathering* to highlight that this attack is new.

7. External Links

[1] http://www.exploit-db.com/google-dorks/
[2] http://nmap.org/man/man-port-scanning-techniques.html
[3] http://www.ietf.org/rfc/rfc1323.txt
[4] http://www.caida.org/publications/papers/2005/fingerprinting/KohnoBroidoClaffy05-devicefingerprinting.pdf
[5] http://en.wikipedia.org/wiki/Network_address_translation
[6] https://www.scip.ch/labs/images/tcp_timestamps_beispiel_netzwerk_h_1200.png
[7] http://nmap.org/book/os-detect.html
[8] http://www.nmap.org
[9] https://github.com/luh2/timestamps/blob/master/identify-hosts.py
[10] http://www.m0n0.ch
[11] labs/images/tcp_timestamps_fingerprints_1800_en.png