Comments

Comment on "Remote Physical Device Fingerprinting"

Libor Polčák, Jakub Jirásek, and Petr Matoušek

Abstract—In this paper we revisited a method to identify computers by their clocks skew computed from TCP timestamps. We introduced our own tool to compute clock skew of computers in a network. We validated that the original method is suitable for the computer identification but we also discovered that Linux hosts running NTP had become immune to the identification.

Index Terms—Network-level security and protection, privacy

1 Introduction

Kohno et al. [1] have shown that TCP timestamps can be used by a remote observer (fingerprinter) to compute clock skew which is quite unique for every computer (fingerprintee). Thus, a fingerprinter may learn that observed traffic was generated by the original host even though the host has changed its IP address. Moreover, even though the fingerprintee changes the interface used for a communication, e.g., it switches from Ethernet to Wi-Fi, the fingerprinter still can see the traffic with the same clock skew.

Clock skew is defined as the difference between the measured and exact time expressed in parts per million (ppm), i.e., the number of microseconds that the clocks differ every second. Kohno et al. [1] proposed to compute the clock skew of a tracked computer from the first derivative of the offset between the time observed in TCP timestamps and local time on the fingerprinter with respect to time. Clock skew of one computer running the same OS differs by 1-2 ppm over time whereas the difference between different computers can be in tens or even hundreds of ppm.

This comment validates the results achieved by Kohno et al. [1] and provides up-to-date information about the behavior of Linux hosts that synchronize their time to improve local time precision. This note is organized as follows. Section 2 introduces a tool to compute clock skew and it summarizes which of the results originally published by Kohno et al. [1] are still valid and what has changed. Section 3 covers a change in Linux kernel that changed TCP timestamps generation process. This change requires an update to the results originally published by Kohno et al. [1, Section 8.1]. Section 4 concludes the contribution of the paper.

2 Measurements

We have developed a tool called PC Fingerprinter (PCF) [2] that computes clock skew of the computers whose traffic can be seen on the selected network interfaces and that have TCP timestamps enabled.

PCF stores offsets between time from TCP timestamps and local time on the fingerprinter. The clock skew is computed according to the algorithm and formulae presented by Kohno et al. [1, Sections 3 and 4.3]. As pointed in the original paper the clock skew is represented by the slope of the upper bound of all offset points (i.e., the first derivative of the function defined by the upper bound of the observed offsets between the time carried in TCP timestamps and local time). If the function of upper bound of the observed offsets forms a line, then the first derivative is constant at every point of the function and, therefore, the clock skew is constant. However, if the function of upper bound of the observed offsets forms a curve, then the first derivative is not constant at every point of the upper bound function and the clock skew changes over time.

2.1 Validation of Originally Published Results

The goal of PCF is twofold. The first goal is validation of the results published by Kohno et al. [1] that we need for our research [3] on determining computer identity in IPv6 networks. The second goal of PCF is to provide a tool for other researchers to investigate the topic further.

During experiments, we validated [4] that:

- Computers can by identified by their clock skew. However, it is important to understand that two computers may have very similar clock skew. For example, some computers with same hardware and software configuration located in our laboratory were not uniquely distinguishable by their clock skew. See Table 1 for the results of the measurements.
- Clock skew remains the same even if the computer is restarted. However, booting up a different OS or sometimes even usage of a different Linux kernel resulted in a very different clock skew, see Table 1 for an example. Note that even though two computers have different clock skew in one OS, they may have very similar clock skew in another OS.
- Clock skew is the same for different physical network interfaces. A laptop with both Ethernet and Wi-Fi card was fingerprinted in several locations. The laptop had a similar clock skew in all measurements regardless of the communicating interface and the distance between the fingerprinter and the fingerprintee.
- NTP daemon running on a computer with FreeBSD or Windows does not influence the clock skew of the host computer.

These experiments confirm the results published by Kohno et al. [1].

2.2 Discovered Distinctions

Investigation of the influence of NTP on Linux hosts unveiled that the behavior of this OS had changed. Fig. 1 shows a graph of a measurement of a computer running Linux kernel 3.4.5 (released in 2012). The upper bound of the observed offset between the time observed in TCP timestamps and local time on the fingerprinter does not form a straight line but it forms a curve. Therefore, the first derivative changes over time and the clock skew of the computer is not constant.

Examination of the TCP timestamps generated by various Linux distributions including Ubuntu, CentOS, Debian, Gentoo, and Arch Linux confirmed that NTP influence on TCP timestamps is not distribution-specific as all Linux distributions behaved the same. Unlike the results published by Kohno et al. [1, Section 8.1] our observations suggest that the clock skew of a Linux computer is indeed influenced by the time synchronization process.

Further investigation exposed that even though an NTP daemon is not active but rather a utility called *ntpdate* quickly adjusts the time to the one used by NTP servers, TCP timestamps are still influenced as can be seen in Fig. 2.

- *The authors are with the Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 61266 Brno, Czech Republic. E-mail: {ipolcak, matousp}@fit.vutbr.cz, xjiras02@stud.fit.vutbr.cz.*

Manuscript received 31 Aug. 2012; accepted 11 June 2013. Date of publication 24 June 2013; date of current version 17 Sept. 2014.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2013.26

TABLE 1
Clock Skew of Five Computers in Our Laboratory Tested Running Different OSes

PC	Windows XP SP3	FreeBSD 8.2	Debian (Linux 2.6.32)
1	-18,408	-277,313	59,450
2	-37,285	-277,308	31,938
3	-22,766	-270,529	-67,035
4	-22,256	-277,789	-90,805
5	-26,640	-277,846	-184,733

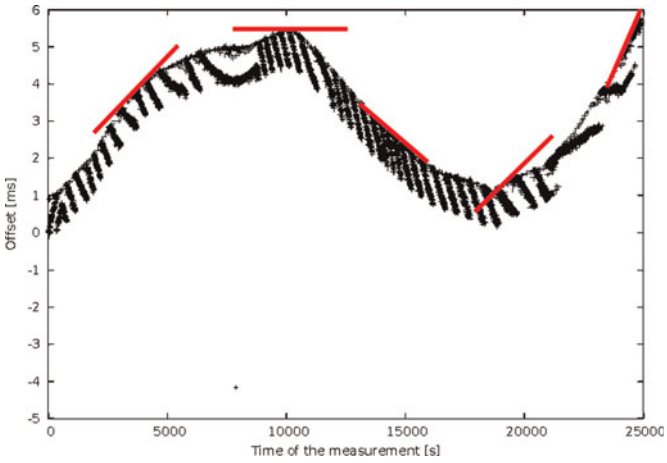

Fig. 1. As the offset between TCP timestamps and local clocks changes irregularly, tangent line has different slope at different points (indicated by several line segments in the figure). Thus, clock skew of this computer with Linux kernel 3.4.5 and NTP enabled varies through time.

3 NTP Influence on TCP Timestamps

NTP [5] is a protocol that distributes information about time among computers. These computers form a hierarchy. Top level consists of computers that get exact time by other means such as GPS signal or atomic clock. On the lower levels, computers synchronize their time with computers on higher levels. A computer that synchronizes its time via NTP runs an NTP tool usually as a daemon. The NTP daemon running on such computer establishes a relationship with few NTP servers and measures the network delay time, jitter, etc. After a while the NTP daemon starts to adjust the period duration of an increment of the local computer internal clocks. If the internal clocks are late, then the clock tick period decreases. Conversely, if the internal clocks are early, the clock tick period is prolonged. The final goal of the NTP daemon is to find such a correction for the internal clocks that the built-in error of the clocks diminishes.

Linux is a free software OS so it is possible to download its source code [6] and locate the part responsible for TCP timestamp assignment. Current version of the code uses kernel function called *ktime_get_real()*, which returns current time influenced by NTP. This confirms our measurements. It is also possible to study the history of changes in the Linux code. The TCP code of the Linux kernel was rewritten around 2007. The remainder of this section focuses on TCP timestamps generated by different versions of Linux kernel. Linux kernel of the time when Kohno et al. [1] wrote their work generated TCP timestamps that were not influenced by NTP. However, Linux kernel version 2.6.24 and newer changed their behavior and TCP timestamps are influenced by NTP.

The behavior of different versions of Linux kernel was studied on old releases of Ubuntu distribution [7] in order to get suitable application environment for the kernels that are already unsupported by current versions of some system tools.

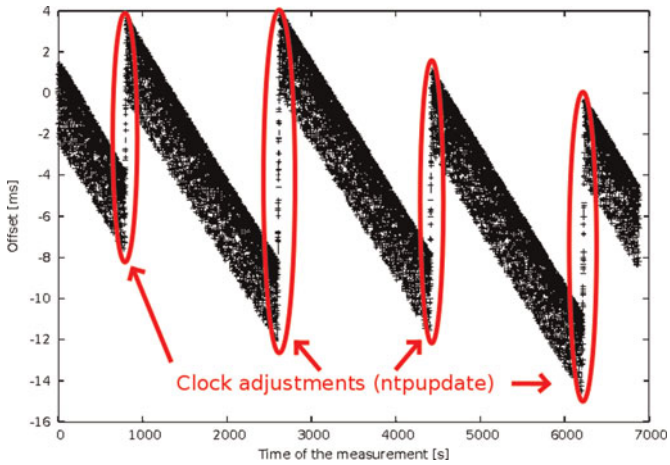

Fig. 2. Observed offset between TCP timestamps and local time changes dramatically during periods of quick clock adjustments. Clock skew of the computer is constant in between the adjustment periods.

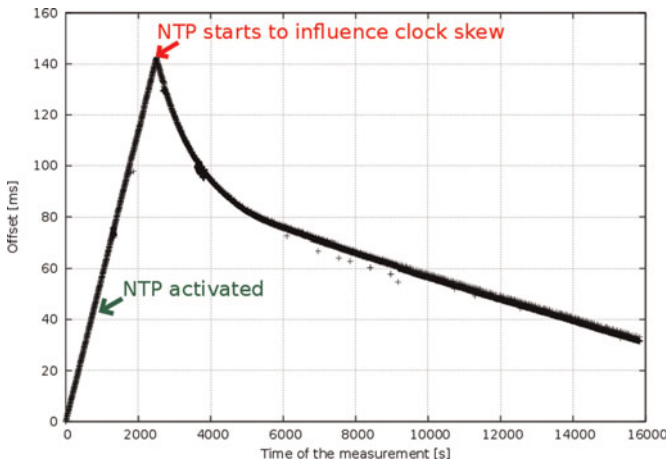

Fig. 3. NTP enabled during the measurement of Ubuntu 7.10 with 2.6.22 Linux kernel after about 15 minutes causes that clock skew starts to vary over time.

Each release was tested in the following way:

1) it was booted up with the disabled NTP and its TCP traffic was monitored,
2) after 15 minutes, an NTP daemon was enabled while the TCP traffic monitoring continued.

The testing started with the recent Ubuntu release and continued with several older releases in reversed order of their release date. Tests advanced until a release with TCP timestamps not influenced by NTP was found.

The experiments revealed that Ubuntu 7.10 with 2.6.22 kernel and later produce NTP-aware TCP timestamps (see Fig. 3). However, TCP timestamps generated by Ubuntu 7.04 with 2.6.20 kernel are not influenced by the NTP (see Fig. 4). Both tested kernels were originally released in 2007. It supports the original theory that the change in the Linux kernel behavior was caused by some of the patches included to the Linux tree in 2007.

Table 2 summarizes the current behavior of different Linux kernels and other OSes.

4 Conclusion

This comment updates knowledge about the clock-skew-based computer identification method originally published by Kohno et al. [1]. We introduced a tool for computer identification that uses this method and is freely available to be used by other researchers for their own experiments.

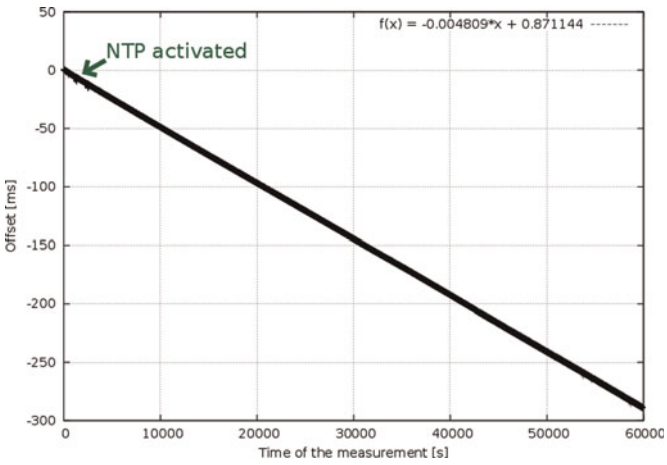

Fig. 4. Even though the NTP daemon was enabled in Ubuntu 7.04 with 2.6.20 Linux kernel during the measurement, clock skew of the computer is constant over time.

TABLE 2
Current Behavior of Different OSes with Respect to NTP Influence on TCP Timestamp Generation

Operating system	TCP Timestamps influenced by NTP
Windows XP, Vista, 7	No
Linux $\leq$ 2.6.20	No
Linux $\geq$ 2.6.22	Yes
FreeBSD	No

We manifested that it is feasible to identify computers with their clock skew. We confirmed that NTP does not influence TCP timestamps in FreeBSD and Windows. In contrast, we demonstrated that Linux behavior changed in 2007 and Linux computers running NTP are now immune to the clock skew detection method. With the recent advent of smart phones, this may hinder identification of Android devices as they use Linux kernel.

Acknowledgments

This work was supported by the ESF project CZ.1.07/2.3.00/09.0067 "TeamIT—Building Competitive Research Teams in IT". This work is a part of the project VG20102015022 supported by Ministry of the Interior of the Czech Republic and was partially supported by the research plan MSM0021630528.

References

[1] T. Kohno, A. Broido, and K. Claffy, "Remote Physical Device Fingerprinting," *IEEE Trans. Dependable Secure Computing*, vol. 2, no. 2, pp. 93-108, Apr.-June 2005.
[2] J. Jirásek and L. Polčák, "PC Fingerprinter," https://github.com/polcak/pcf, Aug. 2012.
[3] M. Grégr, P. Matoušek, T. Podermański, and M. Švéda, "Practical IPv6 Monitoring—Challenges and Techniques," *Proc. 12th IFIP/IEEE Int'l Symp. Integrated Network Management*, IEEE Computer Society, pp. 660-663, 2011.
[4] J. Jirásek, "Computer Identification Using Time Information," master's thesis, Faculty of Information Technology, Brno Univ. of Technology, 2012.
[5] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," 2010.
[6] "The Linux Kernel Archives," http://www.kernel.org/, Aug. 2012.
[7] "Old Ubuntu Releases," http://old-releases.ubuntu.com/releases/, Aug. 2012.