

NTP Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

A. Malhotra
Boston University
K. Teichel
PTB
M. Hoffmann
W. Toorop
NLnet Labs
July 8, 2019

On Implementing Time
draft-aanchal-time-implementation-guidance-02

Abstract

This document describes the properties of different types of clocks available on digital systems. It provides implementors of applications with guidance on choices they have to make when working with time to provide basic functionality and security guarantees.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Scope of the document	3
3. Expressing Time	3
3.1. Absolute Time	4
3.2. Relative Time	4
4. Keeping Time: Different Clocks	4
4.1. Native Clock	4
4.2. World Clock	5
5. Implementation Approaches	6
6. Accessing the Native Clock on Selected Operating Systems . .	7
6.1. POSIX	7
6.2. Microsoft Window	7
7. IANA Considerations	7
8. Security Considerations	7
9. References	8
9.1. Normative References	8
9.2. Informative References	8
Appendix A. Acknowledgements	8
Authors' Addresses	8

1. Introduction

It is hard to overstate the importance of time in modern digital systems. The functionality and security of applications (distributed or local to one system) and that of network protocols generally hinge on some notion of time. For implementation, these applications and protocols have to choose one of the types of clocks available on their system, each of which has its own specific properties. However, currently many of these applications seem to be oblivious to the implications of choosing one or the other clock for implementation. This behavior can be attributed to:

- a. the lack of clear understanding of the distinct properties of these clocks,
- b. trade-offs of using one or the other for an application, and
- c. availability and compatibility of these clocks on different systems.

This document discusses a) and b).

More specifically, in this document we first define different methods used by protocols and applications to express time. We then define properties of clocks maintained by modern digital systems. Next we describe how systems obtain these values from these clocks and the security considerations of using these values to implement protocols and applications that use time. Finally we discuss trade-offs between security and precision of choosing a clock. The document aims to provide guidance to the implementors make an informed choice with an example of POSIX system.

2. Scope of the document

This document aims to provide software developers implementing protocols and applications that have to deal with time with the knowledge and understanding to make informed decisions regarding the available clocks and their respective trade-offs.

It does not describe functionality that is specific to the architecture of a PC, or other devices such as phones, IoT devices, switches, routers, base stations, or synchrophasors. Nor is the document applicable to a specific operating system. Throughout the document we assume that one or the other clock is available on most devices. How these clocks are available on different PCs or other devices is out of scope of this document.

We do not exactly recommend which clock should be used. We discuss the available options and trade-offs. The final decision would vary depending on the availability of clocks and the security requirements of the specific application under implementation.

Note: Since there is a lack of standards on terminology related to time, we define some terms in the following section. Also, throughout the document, we define the terms as they become relevant. Different systems, depending on their OS, may use different terms for the same types of clocks. A survey on this is not in the scope of this document. We provide a discussion on how to access these values on POSIX and Windows systems. On other systems, implementors will have to determine themselves which of these values are available.

3. Expressing Time

Protocols and applications can express time in several forms, depending on whether they need to express a point in time or a time interval.

3.1. Absolute Time

Absolute time expresses a universally agreed upon reference to a specific point in time. Such a reference can be expressed in different ways. For instance, Unix Time refers to the number of seconds since midnight UTC, January 1st, 1970, while in everyday life, we referenced such a point through year, month, day, and so on.

Because absolute time expresses a shared view of time, a system needs to synchronize its clock with a common reference clock, for instance one base on UTC.

Absolute time is often used to express the start or end of the validity of objects with a limited lifetime that are shared over the network.

3.2. Relative Time

Relative time measures the time interval that has elapsed from some well-defined reference point (e.g., 20 minutes from the time of your query).

Relative time is commonly used in network protocols, for instance to determine when a packet should be considered dropped or to express Time To Live (TTL) values that govern the length of time for which an object is valid or usable.

Since relative time does not express a point in time, it does not rely on synchronized clocks between systems but only on a shared clock rate.

4. Keeping Time: Different Clocks

In this section, we will have a look at the different clocks a system uses and how it maintains these clocks

4.1. Native Clock

Each system has its own perception of time. It gains access it via its native clock. Typcially, this clock counts cycles of an oscillator but some systems use process CPU times or thread CPU timers (via timers provided by the CPU). The quality of the native clock therefore depends on either the stability of the oscillator or the CPU timer.

The timescale of the native clock is a purely subjective -- no general meaning can be attached to any specific clock value. One can only obtain relative time by comparing two values. Because the value

of the native clock always grows at a steady pace, never decreases, never make unexpected jumps, and never skips, the difference between two clock values provides the time intervall between the two measurements.

The independence of the native clock from any external time sources renders it resistant to any manipulation but in return there is no guarantee that its clock rate is similar to that of any other system. This difference in rate, especially when compared to a reference clock, is called clock drift.

Clock drift depends on the quality of the clock itself but also on factors such as system load or ambient temperatur which makes it hard to predict.

4.2. World Clock

The native clock only provides means to measure relative time. In order to be able to also process absolute time, it needs to be synchronized with a global reference clock. Since this clock strives to be the same on all systems, we call it the world clock.

There are a number of ways to maintain the world clock based on the system's native clock.

- o The first is to manually maintain an offset between values of the native clock and the reference world clock. Because of the clock drift of the native clock, this offset needs to be updated from time to time if a minimal divergence from the reference clock is to be maintained.
- o Secondly, a hardware clock provided by the system and set to be equivalent to the reference time can be used, allowing the system to retain the offset across reboots.
- o Finally, the reference clock can be obtained from an external time source. Typically, the Internet is used through a variety of timing protocols including the Network Time Protocol² (NTP), Chrony, SNTP, OpenNTP and others.

Each of these approaches has own problems attached to it.

- o Manual configurations can be subject to errors and misconfiguration.
- o Accessing the hardware clock requires an I/O operation which is resource intensive, therefore many systems use the hardware clock

only upon reboot, to initialize the clock offset; subsequent updates are made either manually or through timing protocols.

Further, on many systems the quality of the hardware clock isn't very high, leading to a large clock drift if solely relying on it. Worse, systems like microcontrollers that operate within embedded systems (e.g., Raspberry Pi, Arduino, etc.) often lack hardware clocks altogether. These systems rely on external time sources upon reboot and have no means to process absolute time until synchronization with these sources has completed.

- o Relying on Internet timing protocols opens up the system time to attack. Recent papers show vulnerabilities in NTP [ANTP][ANABM][SECNTP] and SNTP [BPHSTS] that allow attackers to maliciously alter system's world clock -- pushing it into the past or even into the future. Moreover, many of these time-shifting attacks can be performed by off-path attackers, who do not occupy a privileged position on the network between the victim system and its time sources on the Internet. Researchers have also demonstrated off-path denial of service attacks on timing protocols that prevent systems from synchronizing their clocks.

In other words, the process of obtaining the offset necessary to provide a world clock creates dependencies that can be exploited.

5. Implementation Approaches

Because absolute time relies on a shared interpretation of a value expressing time, the world clock is necessary when processing such values.

For relative time, however, where only the rate of passage of time needs to be close enough to that of the other systems involved, there is no need to rely on the world clock when determining whether an interval has passed.

Instead, by obtaining a value from the native clock when the interval has started only the native clock is necessary to determine when this interval ends. As the native clock does not rely on any external time sources, the implementation becomes resistant to the difficulties of coordinating with these sources.

However, using the native clock in this way comes with a caveat. Since the native clock is not subject to any adjustments by timing protocols, it is not adjusted for the error introduced by clock drift. While this is likely of little consequence for short intervals, it may become significant for intervals that span long periods of time.

The choice of clock to be used is situation-specific. If a certain amount of clock drift can be tolerated or if time intervals are short, implementors may prefer to use the native clock. However, if precise timing over long periods is required, then the implementors have no choice but to fall back to world clock

6. Accessing the Native Clock on Selected Operating Systems

In most operating systems, the standard functions to access time use the world clock since that is normally what users would expect. This section provides an overview how the native clock can be accessed on some common operating systems.

6.1. POSIX

POSIX defines a system C API function which may provide native time: "clock_gettime()", when used with a "clock_id" of "CLOCK_MONOTONIC".

Note that on some systems "CLOCK_MONOTONIC" is still influenced by an external time source (for synchronizing the clock rate) and the non-standard "CLOCK_MONOTONIC_RAW" needs to be used for clock values not influenced by an external time source and not susceptible for time-shifting attacks.

6.2. Microsoft Windows

In the Microsoft Windows operating system, native time is called 'Windows Time' and can be accessed through the "GetTickCount" and "GetTickCount64" API functions. The returned value is nominally the number of milliseconds since system start. "GetTickCount" will return a 32 bit value while "GetTickCount64" returns a value 64 bits wide that will wrap around less

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

Time is a fundamental component for the security guarantees claimed by various applications. A system that uses a time distribution protocol may be affected by the security aspects of the time protocol. The security considerations of time protocols in general are discussed in [RFC7384]. This document discusses the security considerations with respect to implementing time values in applications in various sections.

9. References

9.1. Normative References

- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<https://www.rfc-editor.org/info/rfc7384>>.

9.2. Informative References

- [ANABM] Malhotra, A. and S. Goldberg, "Attacking NTP's Authenticated Broadcast Mode", 2016, <<https://eprint.iacr.org/2016/055>>.
- [ANTP] Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", 2015, <<https://eprint.iacr.org/2015/1020>>.
- [BPHSTS] Jose, J., "Bypassing HTTP Strict Transport Security", 2014, <<https://www.blackhat.com/docs/eu-14/materials/eu-14-Selvi-Bypassing-HTTP-Strict-Transport-Security-wp.pdf>>.
- [SECNTP] Malhotra, A., Gundy, M., Varia, M., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", 2016, <<http://eprint.iacr.org/2016/1006>>.

Appendix A. Acknowledgements

We are thankful to Sharon Goldberg and Benno Overreinder for useful discussions. Thanks to Dieter Sibold, Joachim Fabini and Denis Reilly, for value input and suggestions.

Authors' Addresses

Aanchal Malhotra
Boston University
111 Cummington Mall
Boston 02215
USA

Email: aanchal4@bu.edu

Kristof Teichel
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Email: kristof.teichel@ptb.de

Martin Hoffmann
NLnet Labs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: martin@nlnetlabs.nl

Willem Toorop
NLnet Labs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: willem@nlnetlabs.nl

Network Time Protocol (ntp) Working Group
Internet-Draft
Updates: rfc5905 (if approved)
Intended status: Standards Track
Expires: February 7, 2020

F. Gont
G. Gont
SI6 Networks
August 6, 2019

Port Randomization in the Network Time Protocol Version 4
draft-gont-ntp-port-randomization-04

Abstract

The Network Time Protocol can operate in several modes. Some of these modes are based on the receipt of unsolicited packets, and therefore require the use of a service/well-known port as the local port number. However, in the case of NTP modes where the use of a service/well-known port is not required, employing such well-known/service port unnecessarily increases the ability of attackers to perform blind/off-path attacks. This document formally updates RFC5905, recommending the use of port randomization for those modes where use of the NTP service port is not required.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Considerations About Port Randomization in NTP	3
3.1. Mitigation Against Off-path Attacks	3
3.2. Effects on Path Selection	4
3.3. Filtering of NTP traffic	4
3.4. Effect on NAT devices	5
3.5. Relation to Other Mitigations for Off-Path Attacks	5
4. Update to RFC5905	5
5. Possible Future Work	6
6. Implementation Status	6
7. IANA Considerations	7
8. Security Considerations	7
9. Acknowledgments	8
10. References	8
10.1. Normative References	8
10.2. Informative References	9
Authors' Addresses	10

1. Introduction

The Network Time Protocol (NTP) is one of the oldest Internet protocols, and currently specified in [RFC5905]. Since its original implementation, standardization, and deployment, a number of vulnerabilities have been found both in the NTP specification and in some of its implementations [NTP-VULN]. Some of these vulnerabilities allow for off-path/blind attacks, where an attacker can send forged packets to one or both NTP peers for achieving Denial of Service (DoS), time-shifts, and other undesirable outcomes. Many of these attacks require the attacker to guess or know at least a target NTP association, typically identified by the tuple {srcaddr, srcport, dstaddr, dstport, keyid}. Some of these parameters may be easily known or guessed.

NTP can operate in several modes. Some of these modes rely on the ability of nodes to receive unsolicited packets, and therefore require the use of a service/well-known port number. However, for

modes where the use of a service/well-known port is not required, employing such well-known/service port improves the ability of an attacker to perform blind/off-path attacks (since knowledge of such port number is typically required for such attacks). A recent study [NIST-NTP] that analyzes the port numbers employed by NTP clients suggests that a considerable number of NTP clients employ the NTP service/well-known port as their local port, or select predictable ephemeral port numbers, thus improving the ability of attackers to perform blind/off-path attacks against NTP.

BCP 156 [RFC6056] already recommends the randomization of transport-protocol ephemeral ports. This document aligns NTP with the recommendation in BCP 156 [RFC6056], by formally updating [RFC5905] such that port randomization is employed for those NTP modes for which the use of the NTP service port is not required.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Considerations About Port Randomization in NTP

The following subsections analyze a number of considerations about transport-protocol port randomization when applied to NTP.

3.1. Mitigation Against Off-path Attacks

There has been a fair share of work in the area of off-path/blind attacks against transport protocols and upper-layer protocols, such as [RFC5927] and [RFC4953]. Whether the target of the attack is a transport protocol instance (e.g., TCP connection) or an upper-layer protocol instance (e.g., an application protocol instance), the attacker is required to know or guess the five-tuple {Protocol, IP Source Address, IP Destination Address, Source Port, Destination Port} that identifies the target transport protocol instance or the transport protocol instance employed by the target upper-layer protocol instance. Therefore, increasing the difficulty of guessing this five-tuple helps mitigate blind/off-path attacks.

As a result of this considerations, BCP 156 [RFC6056] recommends the randomization of transport-protocol ephemeral ports. And as such, this document aims to bring the NTP specification [RFC5905] in line with the aforementioned recommendation.

We note that the use of port randomization is a transport-layer mitigation against off-path/blind attacks, and does not preclude (nor

is it precluded by), other possible mitigations for off-path attacks that might be implemented by an application protocol (e.g. [I-D.ietf-ntp-data-minimization]). For instance, some of the aforementioned mitigations may be ineffective against some off-path attacks [NTP-FRAG] or may benefit from the additional entropy provided by port randomization [NTP-security].

3.2. Effects on Path Selection

Intermediate systems implementing the Equal-Cost Multi-Path (ECMP) algorithm may select the outgoing link by computing a hash over a number of values, that include the transport-protocol source port. Thus, as discussed in [NTP-CHLNG], the selected client port may have an influence on the measured delay and jitter values.

This might mean, for example, that two systems in the same network that synchronize their clocks with the same NTP server might end up with a significant offset between their clocks as a result of their NTP samples taking paths with very different characteristics.

If port randomization is applied for every NTP request, requests/responses would be distributed over the different available paths, including those with the smallest delay. The clock filter algorithm could readily select one of such samples with lowest delays, in the same way that the clock selection and clock cluster algorithms might also end up selecting other time sources with smaller resulting dispersion. On the other hand, if port-randomization is applied on a per-association basis, in scenarios where the aforementioned ECMP algorithm is employed, request/responses to the same association would likely follow the same path, since the IP addresses and transport port numbers employed for an association would not change.

Section 4 recommends NTP implementations to randomize the ephemeral port number of non-symmetrical associations on a per-association basis (as opposed to "per-transaction"), since this more conservative approach avoids the possible negative implications of port randomization on time synchronization.

3.3. Filtering of NTP traffic

In a number of scenarios (such as when mitigating DDoS attacks), a network operator may want to differentiate between NTP requests sent by clients, and NTP responses sent by NTP servers. If an implementation employs the NTP service port for the client port number, requests/responses cannot be readily differentiated by inspecting the source and destination port numbers. Implementation of port randomization for non-symmetrical modes allows for simple differentiation of NTP requests and responses, and for the

enforcement of security policies that may be valuable for the mitigation of DDoS attacks.

3.4. Effect on NAT devices

Some NAT devices will not translate the source port of a packet when a privileged port number is employed. In networks where such NAT devices are employed, use of the NTP service port for the client port will essentially limit the number of hosts that may successfully employ NTP client implementations.

In the case of NAT devices that will translate the source port even when a privileged port is employed, packets reaching the external realm of the NAT will not employ the NTP service port as the local port, since the local port will normally be translated by the NAT device possibly, but not necessarily, with a random port.

3.5. Relation to Other Mitigations for Off-Path Attacks

Ephemeral Port Randomization is a best current practice (BCP 156) that helps mitigate off-path attacks at the transport-layer. It is orthogonal to other possible mitigations for off-path attacks that may be implemented at other layers (such as the use of timestamps in NTP) which may or may not be effective against some off-path attacks (see e.g. [NTP-FRAG]). This document aligns NTP with the existing best current practice on ephemeral port selection, irrespective of other techniques that may (and should) be implemented for mitigating off-path attacks.

4. Update to RFC5905

The following text from Section 9.1 ("Peer Process Variables") of [RFC5905]:

dstport: UDP port number of the client, ordinarily the NTP port number PORT (123) assigned by the IANA. This becomes the source port number in packets sent from this association.

is replaced with:

dstport: UDP port number of the client. In the case of broadcast server mode (5) and symmetric modes (1 and 2), it must contain the NTP port number PORT (123) assigned by the IANA. In other cases, it SHOULD contain a randomized port number, as specified in [RFC6056]. The value in this variable becomes the source port number of packets sent from this association.

NOTES:

When port randomization is employed, the port number must be randomized on a per-association basis. That is, a random port number is selected when an association is first mobilized, and the selected port number is expected to remain constant during the life of an association.

On most current operating systems (that implement ephemeral port randomization [RFC6056]), an NTP client may normally rely on the operating system for performing port randomization. For example, NTP implementations employing the Sockets API may achieve port randomization by *not* specifying the local port for the corresponding socket, or `bind()`ing the local socket to the "special" port 0 (which for the Sockets API has the special meaning of "any port"). `connect()`ing the docket will make the port inaccessible by other systems (that is, only packets from the specified remote socket will be received by the application).

5. Possible Future Work

Port numbers could be randomized on a per-association basis, or on a per-request basis. When the port number is randomized on a per-association basis, a random port number is selected when an association is first mobilized, and the selected port remains constant during the life of the association. On the other hand, when the port number is randomized on a per-request basis, each client request will (statistically) employ a different ephemeral port for each request. As discussed in Section 3, varying the port number across requests may impact the time quality achieved with NTP. As a result, this document recommends the conservative approach of randomizing port numbers on a per-association basis (as opposed to a "per-transaction" basis). The possibility of randomizing port numbers on a per-transaction may be subject of future work, and is not recommended by this document.

6. Implementation Status

[RFC Editor: Please remove this section before publication of this document as an RFC.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort

has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

OpenNTPD:

[OpenNTPD] has never explicitly set the local port of NTP clients, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), OpenNTPD will employ port randomization for client ports.

chrony:

[chrony] has never explicitly set the local port of NTP clients, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), chrony will employ port randomization for client ports.

nwtime.org's sntp client:

sntp does not explicitly set the local port, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), it will employ port randomization for client ports.

7. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

8. Security Considerations

The security implications of predictable numeric identifiers [I-D.gont-predictable-numeric-ids] (and of predictable transport-protocol port numbers [RFC6056] in particular) have been known for a long time now. However, the NTP specification has traditionally followed a pattern of employing common settings and code even when not strictly necessary, which at times has resulted in negative security and privacy implications (see e.g. [I-D.ietf-ntp-data-minimization]). The use of the NTP service port (123) for the srcport and dstport variables is not required for all operating modes, and such unnecessary usage comes at the expense of reducing the amount of work required for an attacker to successfully

perform off-path/blind attacks against NTP. Therefore, this document formally updates [RFC5905], recommending the use of transport-protocol port randomization when use of the NTP service port is not required.

This issue has been tracked by US-CERT with VU#597821, and has been assigned CVE-2019-11331.

9. Acknowledgments

Watson Ladd raised the problem of DDoS mitigation when the NTP service port is employed as the client port (discussed in Section 3.3 of this document).

Miroslav Lichvar suggested randomization of the client port on a per-request basis, to intentionally cause each request/response to employ different paths in scenarios where ECMP is employed.

The authors would like to thank (in alphabetical order) Ivan Arce, Todd Glassey, Watson Ladd, Miroslav Lichvar, Aanchal Malhotra, Danny Mayer, Gary E. Miller, Dieter Sibold, Steven Sommars, and Ulrich Windl, for providing valuable comments on earlier versions of this document.

The authors would like to thank Harlan Stenn for answering questions about nwttime.org's NTP implementation.

Fernando would like to thank Nelida Garcia and Jorge Oscar Gont, for their love and support.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <<https://www.rfc-editor.org/info/rfc6056>>.

10.2. Informative References

- [chrony] "chrony", <<https://chrony.tuxfamily.org/>>.
- [I-D.gont-predictable-numeric-ids]
Gont, F. and I. Arce, "Security and Privacy Implications of Numeric Identifiers Employed in Network Protocols", draft-gont-predictable-numeric-ids-03 (work in progress), March 2019.
- [I-D.ietf-ntp-data-minimization]
Franke, D. and A. Malhotra, "NTP Client Data Minimization", draft-ietf-ntp-data-minimization-04 (work in progress), March 2019.
- [NIST-NTP]
Sherman, J. and J. Levine, "Usage Analysis of the NIST Internet Time Service", Journal of Research of the National Institute of Standards and Technology Volume 121, March 2016, <<https://tf.nist.gov/general/pdf/2818.pdf>>.
- [NTP-CHLNG]
Sommars, S., "Challenges in Time Transfer Using the Network Time Protocol (NTP)", Proceedings of the 48th Annual Precise Time and Time Interval Systems and Applications Meeting, Monterey, California pp. 271-290, January 2017, <http://leapsecond.com/ntp/NTP_Paper_Sommars_PTTI2017.pdf>.
- [NTP-FRAG]
Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", NDSS'17, San Diego, CA. Feb 2017, 2017, <<http://www.cs.bu.edu/~goldbe/papers/NTPattack.pdf>>.
- [NTP-security]
Malhotra, A., Van Gundy, M., Varia, V., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", Cryptology ePrint Archive Report 2016/1006, 2016, <<https://eprint.iacr.org/2016/1006>>.
- [NTP-VULN]
Network Time Foundation, "Security Notice", Network Time Foundation's NTP Support Wiki , <<https://support.ntp.org/bin/view/Main/SecurityNotice>>.
- [OpenNTPD]
"OpenNTPD Project", <<https://www.openntpd.org>>.

- [RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <<https://www.rfc-editor.org/info/rfc4953>>.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <<https://www.rfc-editor.org/info/rfc5927>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.

Authors' Addresses

Fernando Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <https://www.si6networks.com>

Guillermo Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: ggont@si6networks.com
URI: <https://www.si6networks.com>

Network Time Protocol (ntp) Working Group
Internet-Draft
Updates: 5905 (if approved)
Intended status: Standards Track
Expires: December 12, 2021

F. Gont
G. Gont
SI6 Networks
M. Lichvar
Red Hat
June 10, 2021

Port Randomization in the Network Time Protocol Version 4
draft-ietf-ntp-port-randomization-08

Abstract

The Network Time Protocol can operate in several modes. Some of these modes are based on the receipt of unsolicited packets, and therefore require the use of a well-known port as the local port number. However, in the case of NTP modes where the use of a well-known port is not required, employing such well-known port unnecessarily facilitates the ability of attackers to perform blind/off-path attacks. This document formally updates RFC5905, recommending the use of transport-protocol ephemeral port randomization for those modes where use of the NTP well-known port is not required.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 12, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Considerations About Port Randomization in NTP	3
3.1. Mitigation Against Off-path Attacks	3
3.2. Effects on Path Selection	4
3.3. Filtering of NTP traffic	4
3.4. Effect on NAPT devices	5
4. Update to RFC5905	5
5. Implementation Status	6
6. IANA Considerations	7
7. Security Considerations	7
8. Acknowledgments	8
9. References	8
9.1. Normative References	8
9.2. Informative References	9
Authors' Addresses	10

1. Introduction

The Network Time Protocol (NTP) is one of the oldest Internet protocols, and currently specified in [RFC5905]. Since its original implementation, standardization, and deployment, a number of vulnerabilities have been found both in the NTP specification and in some of its implementations [NTP-VULN]. Some of these vulnerabilities allow for off-path/blind attacks, where an attacker can send forged packets to one or both NTP peers for achieving Denial of Service (DoS), time-shifts, or other undesirable outcomes. Many of these attacks require the attacker to guess or know at least a target NTP association, typically identified by the tuple {srcaddr, srcport, dstaddr, dstport, keyid} (see section 9.1 of [RFC5905]). Some of these parameters may be easily known or guessed.

NTP can operate in several modes. Some of these modes rely on the ability of nodes to receive unsolicited packets, and therefore require the use of the NTP well-known port (123). However, for modes where the use of a well-known port is not required, employing the NTP well-known port unnecessarily facilitates the ability of an attacker to perform blind/off-path attacks (since knowledge of the port

numbers is typically required for such attacks). A recent study [NIST-NTP] that analyzes the port numbers employed by NTP clients suggests that a considerable number of NTP clients employ the NTP well-known port as their local port, or select predictable ephemeral port numbers, thus unnecessarily facilitating the ability of attackers to perform blind/off-path attacks against NTP.

BCP 156 [RFC6056] already recommends the randomization of transport-protocol ephemeral ports. This document aligns NTP with the recommendation in BCP 156 [RFC6056], by formally updating [RFC5905] such that port randomization is employed for those NTP modes for which the use of the NTP well-known port is not needed.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Considerations About Port Randomization in NTP

The following subsections analyze a number of considerations about transport-protocol ephemeral port randomization when applied to NTP.

3.1. Mitigation Against Off-path Attacks

There has been a fair share of work in the area of off-path/blind attacks against transport protocols and upper-layer protocols, such as [RFC5927] and [RFC4953]. Whether the target of the attack is a transport protocol instance (e.g., TCP connection) or an upper-layer protocol instance (e.g., an application protocol instance), the attacker is required to know or guess the five-tuple {Protocol, IP Source Address, IP Destination Address, Source Port, Destination Port} that identifies the target transport protocol instance or the transport protocol instance employed by the target upper-layer protocol instance. Therefore, increasing the difficulty of guessing this five-tuple helps mitigate blind/off-path attacks.

As a result of these considerations, transport-protocol ephemeral port randomization is a best current practice (BCP 156) that helps mitigate off-path attacks at the transport-layer. This document aligns the NTP specification [RFC5905] with the existing best current practice on ephemeral port selection, irrespective of other techniques that may (and should) be implemented for mitigating off-path attacks.

We note that transport-protocol ephemeral port randomization is a transport-layer mitigation against off-path/blind attacks, and does not preclude (nor is it precluded by) other possible mitigations for off-path attacks that might be implemented at other layers (e.g. [I-D.ietf-ntp-data-minimization]). For instance, some of the aforementioned mitigations may be ineffective against some off-path attacks [NTP-FRAG] or may benefit from the additional entropy provided by port randomization [NTP-security].

3.2. Effects on Path Selection

Intermediate systems implementing the Equal-Cost Multi-Path (ECMP) algorithm may select the outgoing link by computing a hash over a number of values, that include the transport-protocol source port. Thus, as discussed in [NTP-CHLNG], the selected client port may have an influence on the measured offset and delay.

If the source port is changed with each request, packets in different exchanges will be more likely to take different paths, which could cause the measurements to be less stable and have a negative impact on the stability of the clock.

Network paths to/from a given server are less likely to change between requests if port randomization is applied on a per-association basis. This approach minimizes the impact on the stability of NTP measurements, but may cause different clients in the same network synchronized to the same NTP server to have a significant stable offset between their clocks due to their NTP exchanges consistently taking different paths with different asymmetry in the network delay.

Section 4 recommends NTP implementations to randomize the ephemeral port number of client/server associations. The choice of whether to randomize the port number on a per-association or a per-request basis is left to the implementation.

3.3. Filtering of NTP traffic

In a number of scenarios (such as when mitigating DDoS attacks), a network operator may want to differentiate between NTP requests sent by clients, and NTP responses sent by NTP servers. If an implementation employs the NTP well-known port for the client port number, requests/responses cannot be readily differentiated by inspecting the source and destination port numbers. Implementation of port randomization for non-symmetrical modes allows for simple differentiation of NTP requests and responses, and for the enforcement of security policies that may be valuable for the

mitigation of DDoS attacks, when all NTP clients in a given network employ port randomization.

3.4. Effect on NAT devices

Some NAT devices will reportedly not translate the source port of a packet when a system port number (i.e., a port number in the range 0-1023) [RFC6335] is employed. In networks where such NAT devices are employed, use of the NTP well-known port for the client port may limit the number of hosts that may successfully employ NTP client implementations at any given time.

NOTES:

NAT devices are defined in Section 4.1.2 of [RFC2663].

The reported behavior is similar to the special treatment of UDP port 500 that has been documented in Section 2.3 of [RFC3715].

In the case of NAT devices that will translate the source port even when a system port is employed, packets reaching the external realm of the NAT will not employ the NTP well-known port as the source port, as a result of the port translation function performed by the NAT device.

4. Update to RFC5905

The following text from Section 9.1 ("Peer Process Variables") of [RFC5905]:

dstport: UDP port number of the client, ordinarily the NTP port number PORT (123) assigned by the IANA. This becomes the source port number in packets sent from this association.

is replaced with:

dstport: UDP port number of the client. In the case of broadcast server mode (5) and symmetric modes (1 and 2), it SHOULD contain the NTP port number PORT (123) assigned by the IANA. In the client mode (3), it SHOULD contain a randomized port number, as specified in [RFC6056]. The value in this variable becomes the source port number of packets sent from this association. The randomized port number SHOULD NOT be shared with other associations, to avoid revealing the randomized port to other associations.

If a client implementation performs ephemeral port randomization on a per-request basis, it SHOULD close the corresponding socket/port after each request/response exchange. In order to prevent

duplicate or delayed server packets from eliciting ICMP port unreachable error messages at the client, the client MAY wait for more responses from the server for a specific period of time (e.g. 3 seconds) before closing the UDP socket/port.

NOTES:

Randomizing the ephemeral port number on a per-request basis will better mitigate off-path/blind attacks, particularly if the socket/port is closed after each request/response exchange, as recommended above. The choice of whether to randomize the ephemeral port number on a per-request or a per-association basis is left to the implementation, and should consider the possible effects on path selection along with its possible impact on time measurement.

On most current operating systems, which implement ephemeral port randomization [RFC6056], an NTP client may normally rely on the operating system to perform ephemeral port randomization. For example, NTP implementations using POSIX sockets may achieve ephemeral port randomization by **not** binding the socket with the `bind()` function, or binding it to port 0, which has a special meaning of "any port". `connect()`ing the socket will make the port inaccessible by other systems (that is, only packets from the specified remote socket will be received by the application).

5. Implementation Status

[RFC Editor: Please remove this section before publication of this document as an RFC.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

OpenNTPD:

[OpenNTPD] has never explicitly set the local port of NTP clients, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), OpenNTPD will employ port randomization for client ports.

chrony:

[chrony] by default does not set the local client port, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), chrony will employ port randomization for client ports.

nwtime.org's sntp client:

sntp does not explicitly set the local port, and thus employs the ephemeral port selection algorithm implemented by the operating system. Thus, on all operating systems that implement port randomization (such as current versions of OpenBSD, Linux, and FreeBSD), it will employ port randomization for client ports.

6. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

7. Security Considerations

The security implications of predictable numeric identifiers [I-D.irtf-pearg-numeric-ids-generation] (and of predictable transport-protocol port numbers [RFC6056] in particular) have been known for a long time now. However, the NTP specification has traditionally followed a pattern of employing common settings even when not strictly necessary, which at times has resulted in negative security and privacy implications (see e.g. [I-D.ietf-ntp-data-minimization]). The use of the NTP well-known port (123) for the srcport and dstport variables is not required for all operating modes. Such unnecessary usage comes at the expense of reducing the amount of work required for an attacker to successfully perform off-path/blind attacks against NTP. Therefore, this document formally updates [RFC5905], recommending the use of transport-protocol port randomization when use of the NTP well-known port is not required.

This issue has been assigned CVE-2019-11331 [VULN-REPORT] in the U.S. National Vulnerability Database (NVD).

8. Acknowledgments

The authors would like to thank (in alphabetical order) Ivan Arce, Roman Danyliw, Dhruv Dhody, Lars Eggert, Todd Glassey, Blake Hudson, Benjamin Kaduk, Erik Kline, Watson Ladd, Aanchal Malhotra, Danny Mayer, Gary E. Miller, Bjorn Mork, Hal Murray, Francesca Palombini, Tomoyuki Sahara, Zaheduzzaman Sarker, Dieter Sibold, Steven Sommars, Jean St-Laurent, Kristof Teichell, Brian Trammell, Eric Vyncke, Ulrich Windl, and Dan Wing, for providing valuable comments on earlier versions of this document.

Watson Ladd raised the problem of DDoS mitigation when the NTP well-known port is employed as the client port (discussed in Section 3.3 of this document).

The authors would like to thank Harlan Stenn for answering questions about nwttime.org's NTP implementation.

Fernando would like to thank Nelida Garcia and Jorge Oscar Gont, for their love and support.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <<https://www.rfc-editor.org/info/rfc6056>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [chrony] "chrony", <<https://chrony.tuxfamily.org/>>.
- [I-D.ietf-ntp-data-minimization]
Franke, D. F. and A. Malhotra, "NTP Client Data Minimization", draft-ietf-ntp-data-minimization-04 (work in progress), March 2019.
- [I-D.irtf-pearg-numeric-ids-generation]
Gont, F. and I. Arce, "On the Generation of Transient Numeric Identifiers", draft-irtf-pearg-numeric-ids-generation-07 (work in progress), February 2021.
- [NIST-NTP]
Sherman, J. and J. Levine, "Usage Analysis of the NIST Internet Time Service", Journal of Research of the National Institute of Standards and Technology Volume 121, March 2016, <<https://tf.nist.gov/general/pdf/2818.pdf>>.
- [NTP-CHLNG]
Sommars, S., "Challenges in Time Transfer Using the Network Time Protocol (NTP)", Proceedings of the 48th Annual Precise Time and Time Interval Systems and Applications Meeting, Monterey, California pp. 271-290, January 2017, <http://leapsecond.com/ntp/NTP_Paper_Sommars_PTTI2017.pdf>.
- [NTP-FRAG]
Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", NDSS'17, San Diego, CA. Feb 2017, 2017, <<https://www.cs.bu.edu/~goldbe/papers/NTPattack.pdf>>.
- [NTP-security]
Malhotra, A., Van Gundy, M., Varia, V., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", Cryptology ePrint Archive Report 2016/1006, 2016, <<https://eprint.iacr.org/2016/1006>>.
- [NTP-VULN]
Network Time Foundation, "Security Notice", Network Time Foundation's NTP Support Wiki , <<https://support.ntp.org/bin/view/Main/SecurityNotice>>.
- [OpenNTPD]
"OpenNTPD Project", <<https://www.openntpd.org>>.

- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <<https://www.rfc-editor.org/info/rfc2663>>.
- [RFC3715] Aboba, B. and W. Dixon, "IPsec-Network Address Translation (NAT) Compatibility Requirements", RFC 3715, DOI 10.17487/RFC3715, March 2004, <<https://www.rfc-editor.org/info/rfc3715>>.
- [RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <<https://www.rfc-editor.org/info/rfc4953>>.
- [RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, DOI 10.17487/RFC5927, July 2010, <<https://www.rfc-editor.org/info/rfc5927>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [VULN-REPORT] The MITRE Corporation, "CVE-2019-11331", April 2019, <<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-11331>>.

Authors' Addresses

Fernando Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <https://www.si6networks.com>

Guillermo Gont
SI6 Networks
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: ggont@si6networks.com
URI: <https://www.si6networks.com>

Miroslav Lichvar
Red Hat
Purkynova 115
Brno 612 00
Czech Republic

Email: mlichvar@redhat.com