Securing Network Time Protocol

Crackers discover how to use NTP as a weapon for abuse

Harlan Stenn

In the late 1970s David L. Mills began working on the problem of synchronizing time on networked computers, and NTP (Network Time Protocol) version 1 made its debut in 1980. This was at a time when the net was a much friendlier place—the ARPANET days. NTP version 2 appeared approximately a year later, about the same time as CSNET (Computer Science Network). NSFNET (National Science Foundation Network) launched in 1986. NTP version 3 showed up in 1993.

Depending on where you draw the line, the Internet became useful in 1991-1992 and fully arrived in 1995. NTP version 4 appeared in 1997. Now, 17 years later, IETF (Internet Engineering Task Force) is almost done finalizing the NTP version 4 standard, and some of us are starting to think about NTP version 5.

All of this is being done by volunteers—with no budget, just by the good graces of companies and individuals who care. This is not a sustainable situation. NTF (Network Time Foundation) is the vehicle that can address this problem, with the support of other organizations and individuals. For example, the Linux Foundation's Core Infrastructure Initiative recently started partially funding two NTP developers: Poul-Henning Kamp for 60 percent of his available time to work on NTP, and me for 30-50 percent of my NTP development work. (Please visit <http://nwtime.org/> to see who is supporting Network Time Foundation.)

On the public Internet, NTP tends to be visible from three types of machines. One is in embedded systems. When shipped misconfigured by the vendor, these systems have been the direct cause of abuse (http://en.wikipedia.org/wiki/NTP_server_misuse_and_abuse). These systems do not generally support external monitoring, so they are not generally abusable in the context of this article. The second set of machines would be routers, and the majority of the routers that run NTP are from Cisco and Juniper. The third set of machines tend to be Windows machines that run win32time (which does not allow monitoring, and is therefore neither monitorable, nor abusable in this context), and Unix boxes that run NTP, acting as local time servers and distributing time to other machines on the LAN that run NTP to keep the local clock synchronized.

For the first 20 years of NTP's history, these local time servers were often old, spare machines that ran a dazzling array of operating systems. Some of these machines kept much better time than others, and people would eventually run them as their master time servers. This is one of the main reasons the NTP codebase stuck with K&R C (named for its authors Brian Kernighan and Dennis Ritchie) for so many years, as that was the only easily available compiler on some of these older machines.

It wasn't until December 2006 that NTP upgraded its codebase from K&R C to ANSI C. For a good while, only C89 was required. This was a full six years beyond Y2K, when a lot of these older operating systems were obsolete but still in production. By this time, however, the hardware on which NTP was "easy" to run had changed to x86 gear, and gcc (GNU Compiler Collection) was the easy compiler choice.

The NTP codebase does its job very well, is very reliable, and has had an enviable record as far as security problems go. Companies and people often run ancient versions of this software on some embedded systems that effectively never get upgraded and run well enough for a very long time.

People just expect accurate time, and they rarely see the consequences of inaccurate time. If the time is wrong, it's often more important to fix it fast and then—maybe—see if the original problem can be identified. The odds of identifying the problem increase if it happens with any frequency. Last year, NTP and our software had an estimated 1 *trillion* hours plus of operation. We've received some bug reports over this interval, and we have some open bug reports we would love to resolve, but in spite of this, NTP generally runs very, very well.

Having said all of this, I should re-emphasize that NTP made its debut in a much friendlier environment, and that if there was a problem with the time on a machine, it was important to fix the problem as quickly as possible. Over the years, this translated into making it easy for people to query an NTP instance to see what it has been doing. There are two primary reasons for this: one is so it's easy to see if the remote server you want to sync with is configured and behaving adequately; the other is so it's easy to get help from others if there is a problem.

While we've been taking steps over the years to make NTP more secure and immune to abuse, the public Internet had more than 7 million abusable NTP servers in the fall of last year. As a result of people upgrading software, fixing configuration files, or because, sadly, some ISPs and IXPs have decided to block NTP traffic, the number of abusable servers has dropped by almost 99 percent in just a few months. This is a remarkably large and fast decline, until you realize that around 85,000 abusable servers still exist, and a DDoS (distributed denial-of-service) attack in the range of 50 to 400 Gbps can be launched using 5,000 servers. There is still a lot of cleanup to be done.

One of the best and easiest ways of reducing and even eliminating DDoS attacks is to make sure that computers on *your* networks send packets that come only from *your* IP space. To this end, you should visit <http://www.bcp38.info/> and take steps to implement this practice for your networks, if you haven't already done so.

As I mentioned above, NTP runs on the public Internet in three major places: embedded devices; Unix and some Windows computers; and Cisco and Juniper routers. Before we take a look at how to configure the latter two groups of these so they can't be abused, let's look at the NTP release history.

NTP RELEASE HISTORY

David L. Mills, now a professor emeritus and adjunct professor at the University of Delaware, gave us NTP version 1 in 1980. It was good, and then it got better. A new "experimental" version, xntp2, installed the main binary as xntpd, because, well, that was the easy way to keep the previous version and new version on a box at the same time. Then version 2 became stable and a recommended standard (RFC 1119), so work began on xntp3. But the main program was still installed as xntpd, even though the program wasn't really "experimental." Note that RFC-1305 defines NTPv3, but that standard was never finalized as a recommended standard—it remained a draft/elective standard. The RFC for NTPv4 is still in development but is expected to be a full, recommended standard.

As for the software release numbering, three of the releases from Mills are xntp3.3wx, xntp3.3wy, and xntp3.5f. These date from just after the time I started using NTP heavily, and I was also sending in portability patches. Back then, you unpacked the tarball, manually edited a config.local file, and did interesting things with the makefile to get the code to build. While Perl's metaconfig was

available then and was great for poking around a system, it did not support subdirectory builds and thus could not use a single set of source code for multiple targets.

GNU autoconf was still pretty new at that time, and while it did not do nearly as good a job at poking around, it did support subdirectory builds. xntp3.5f was released just as I volunteered to convert the NTP code base to GNU autoconf. As part of that conversion, Mills and I discussed the version numbers, and he was OK with my releasing the first cut of the GNU autoconf code as xntp3-5.80. These were considered alpha releases, as .90 and above were reserved for beta releases. The first production release for this code would be xntp3-6.0, the sixth major release of NTPv3, except that shortly after xntp3-5.93e was released in late November 1993, Mills decided that the NTPv3 code was good enough and that it was time to start on NTPv4.

At that point, I noticed that many people had problems with the version-numbering scheme, as the use of both the dash (-) and dot (.) characters really confused people. So ntp-4.0.94a was the first beta release of the NTPv4 code in July 1997. The release numbers went from ntpPROTO-Maj.Min to ntp-PROTO.Maj.Min.

While this change had the desired effect of removing confusion about how to type the version number, it meant that most people didn't realize that going from ntp-4.1.x to 4.2.x was a *major* release upgrade. People also didn't seem to understand just how many items were being fixed or improved in minor releases. For more information about this, see table 1.

At one point I tried going back to a version-numbering scheme that was closer to the previous method, but I got a lot of pushback so I didn't go through with it. In hindsight, I should have stood my ground. Having seen how people don't appreciate the significance of the releases—major or minor—we will go back to a numbering scheme much closer to the original after 4.2.8 is released. The major release after ntp-4.2.8 will be ntp4-5.0.0, or ntpPROTO-Maj.Min.Point. (Our source

TABLE 1: NTP Release History

Version	Release Date	EOL Date	# Bugs fixes and Improvements
ntp-4.2.8	Dec 2014		Over 1100 so far
ntp-4.2.6	Dec 2009	Dec 2014	630-1000
ntp-4.2.4	Dec 2006	Dec 2009	Over 450
ntp-4.2.2	Jun 2006	Dec 2006	Over 560
ntp-4.2.0	Oct 2003	Jun 2006	?
ntp-4.1.2	Jul 2003	Oct 2003	?
ntp-4.1.1	Feb 2002	Jul 2003	?
ntp-4.1.0	Aug 2001	Feb 2002	?
ntp-4.0.99	Jan 2000	Aug 2001	?
ntp-4.0.90	Nov 1998	Jan 2000	?
ntp-4.0.73	Jun 1998	Nov 1998	?
ntp-4.0.72	Feb 1998	Jun 1998	?
ntp-4.0	Sep 1997	Feb 1998	?
xntp3-5.86.5	Oct 1996	Sep 1997	?
xntp3.5f	Apr 1996	Oct 1996	?
xntp3.3wy	Jun 1994	Apr 1996	?
xntp3	Jun 1993	Jun 1994	?
xntp2	Nov 1989	Jun 1993	?

archives reveal how the release numbering choices have evolved over the years, and how badly some of them collated.)

SECURING NTP

Before we delve into how to secure NTP, I recommend you listen to Dan Geer's keynote speech for Blackhat 2014, if you have not already done so (<https://www.youtube.com/watch?v=nT-TGvYOBpI>). It will be an excellent use of an hour of your time. If you watch it and disagree with what he says, then I wonder why you're reading this document to look for a solution to NTP abuse vector problems.

Now, to secure NTP, first implement BCP38 (<http://www.bcp38.info>). It's not that hard.

If you want to make sure that NTP on your Cisco or Juniper routers is protected, then consult their documentation on how to do so. You will find lots of good discussions and articles on the Web with additional updated information, and I recommend <http://www.team-cymru.org/ReadingRoom/Templates/secure-ntp-template.html> for information about securing NTP on Cisco and Juniper routers.

The NTP support site provides information on how to secure NTP through the `ntp.conf` file. Find some discussion and a link to that site at <http://nwtime.org/ntp-winter-2013-network-drdoS-attacks/>. NTF is also garnering the resources to produce an online `ntp.conf` generator that will implement BCP for this file and make it easy to update that file as our codebase and knowledge evolve.

That said, the most significant NTP abuse vectors are disabled by default starting with `ntp-4.2.7p27`, and these and other improvements will be in `ntp-4.2.8`, which was released at the end of 2014. For versions 4.2.6 through 4.2.7p27, this abuse vector can be prevented by adding the following to your `ntp.conf` file:

```
restrict default ... noquery ...
```

Note well that if you have additional `restrict` lines for IPs or networks that do `_not_` include the `noquery` restriction, ask yourself if it's possible for those IPs to be spoofed...

For version 4.2.4, which was released in December 2006 and EOled (brought to the end-of-life) in December 2009, consider the following:

- You didn't pay attention to what Dan Geer said.
- Did you notice that we fixed 630-1,000 issues going from 4.2.4 to 4.2.6?
- Are you still interested in running 4.2.4? Do you really have a good reason for this?

If so, add to your `ntp.conf` file:

```
restrict default ... noquery ...
```

For version 4.2.2, which was released in June 2006 and EOled in December 2006:

- You didn't pay attention to what Dan Geer said.
- Did you notice that we fixed about 450 issues going from 4.2.2 to 4.2.4, and 630-1,000 issues going from 4.2.4 to 4.2.6? That's between 1,000 and 1,500 issues. Seriously.
- Are you still interested in running 4.2.2? Do you really have a good reason for this?

If so, add to your `ntp.conf` file:

```
restrict default ... noquery ...
```

For version 4.2.0, which was released in 2003 and EOled in 2006:

- You didn't pay attention to what Dan Geer said.
- Did you notice that we fixed about 560 issues going from 4.2.0 to 4.2.2, 450 issues going from 4.2.2 to 4.2.4, and 630-1,000 issues going from 4.2.4 to 4.2.6? That's between 1,500 and 2,000 issues. Seriously.
- Are you still interested in running 4.2.2? Do you really have a good reason for this?
If so, add to your ntp.conf file:

```
restrict default ... noquery ...
```

For versions 4.0 through 4.1.1, which were released and EOled somewhere around 2001 to 2003, no numbers exist for how many issues were fixed in these releases:

- You didn't pay attention to what Dan Geer said.
- There are probably in excess of 2,000-2,500 issues fixed since then.
- Are you still interested in running 4.0 or 4.1 code? Do you really have a good reason for this?
If so, add to your ntp.conf file:

```
restrict default ... noquery ...
```

Now let's talk about xntp3, which was EOled in September 1997. Do the math on how old that is, take a guess at how many issues have been fixed since then, and ask yourself and anybody else who has a voice in the matter: why are you running software that was EOled 17 years ago, when thousands of issues have been fixed and an improved protocol has been implemented since then?

If your answer is: "Because NTPv3 was a standard and NTPv4 is not yet a standard," then I have bad news for you. NTPv3 was *not* a recommended standard; it was only a draft/elective standard. If you really want to run only officially standard software, you can drop back to NTPv2—and I don't know anybody who would want to do that.

If your answer is: "We're not sure how stable NTPv4 is," then I'll point out that NTPv4 has an estimated 5-10 *trillion* operational hours at this point. How much more do you want?

But if you insist, the way to secure xntp2 and xntp3 against the described abuse vector is to add to your ntp.conf file:

```
restrict default ... noquery ...
```

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

HARLAN STENN learned programming on a PDP-8/S (mostly FOCAL and assembler), which had 4 KB of real core memory, at about the same time he learned to drive a car. While helping other companies write and maintain portable Unix software, he discovered the need for synchronized time (to build software stored on NFS file systems), which led him to "timed" and "ntp." He started submitting NTP patches to David Mills, which led to his converting the NTP codebase to use GNU Autoconf for its build

system. Then he became NTP's release engineer and bugfix integrator. From there, he became the NTP project manager. Watching the inability of the volunteer crew to keep up with the increasing workload, he founded Network Time Foundation in 2011, a public benefit company with a mission to improve the state of computer network timekeeping, working to support NTP (Network Time Protocol), PTP (Precision Time Protocol), and other time-related projects.

© 2014 ACM 1542-7730/14/1200 \$10.00