Did the Shark Eat the Watchdog in the NTP Pool? Deceiving the NTP Pool’s Monitoring System

Jonghoon Kwon\textsuperscript{+}, Jeonggyu Song\textsuperscript{*}, Junbeom Hur\textsuperscript{*}, and Adrian Perrig\textsuperscript{+}

\textsuperscript{+}ETH Zürich, \textsuperscript{*}Korea University
Public servers abuse

2891 views

David L. Mills

Jan 21, 2003, 10:47:35 PM

Folks,

At the request of a national time standards laboratory I have removed their NTP servers from the public lists. The timekeepers cited gross violations of their access policy and the expense of the network service, especially for unintended international users. As you know from my previous grouse to this list, this is a growing problem and may well lead to the loss of public time service altogether.

You may not have noticed it, but provisions added to recent NTP versions includes symmetric and public key cryptography, which is my recommended method for source authentication. It is a trivial matter to require this for access control as well and I am preparing to do exactly this for our public time servers and recommending it for the national laboratories.

It is to work like this. With NTPv4 you will need OpenSSL and an encrypted identity key, as well as public/private keys you generate

Michael Wouters

Jan 23, 2003, 7:50:11 AM

The problem we are facing is simply paying for the traffic.

A year ago, life was simple. We got about 10 packets/server/s and this was growing linearly, or at least close to linear over a time scale of two years. Then, something changed. Traffic started to grow exponentially and is now at 200 packets/s. Projecting current growth we will have another factor of 10 in about 3 months.

200 packets/s is about 1.5 GB per day or roughly $40 per day or $15000 per year. Not so frightening now, but in 3 months it will be 10 times more.
NTP Pool Project: the Largest NTP Ecosystem

- Response to the increasing resource consumption at popular NTP servers

4.6 k public timeservers (Aug. 2023)

Hundreds of millions of Clients
- Linux distributions (e.g., Debian)
- Networked appliances (e.g., Netgear)
- Android smartphones and IoT devices
NTP Pool Architecture at a Glance

NTP Server Pool

DNS

ch.pool.ntp.org?

Client

Non-authoritative answer:
Name: ch.pool.ntp.org
Address: 217.147.223.78
Name: ch.pool.ntp.org
Address: 82.197.188.130
Name: ch.pool.ntp.org
Address: 192.33.214.47
Name: ch.pool.ntp.org
Address: 195.141.190.190
NTP Pool Architecture at a Glance

DNS

NTP Pool

Client

ch.pool.ntp.org?

Non-authoritative answer:
Name: ch.pool.ntp.org
Address: 217.147.223.78
Name: ch.pool.ntp.org
Address: 82.197.188.130
Name: ch.pool.ntp.org
Address: 192.33.214.47
Name: ch.pool.ntp.org
Address: 195.141.190.190
How is the time accuracy of these public servers managed?

Non-authoritative answer:
Name: ch.pool.ntp.org
Address: 217.147.223.78
Name: ch.pool.ntp.org
Address: 82.197.188.130
Name: ch.pool.ntp.org
Address: 192.33.214.47
Name: ch.pool.ntp.org
Address: 195.141.190.190
NTP Pool Architecture at a Glance

- **Client**
  - Queries `ch.pool.ntp.org` for NTP servers

- **NTP Pool**
  - Contains multiple NTP servers

- **NTP Server**
  - Responds to client queries

- **Monitor**
  - Oversees the NTP pool and its servers

Non-authoritative answer:
- **Name:** ch.pool.ntp.org
- **Address:** 217.147.223.78
- **Name:** ch.pool.ntp.org
- **Address:** 82.197.188.130
- **Name:** ch.pool.ntp.org
- **Address:** 192.33.214.47
- **Name:** ch.pool.ntp.org
- **Address:** 195.141.190.190
NTP Pool Monitoring System

• Scoring algorithm

  \[ \text{score}_{new} = \min(\text{max\_score}, (\text{score}_{old} \times 0.95) + \text{step}) \]

  – Step formula:

  ```
  Algorithm 1: Step Formula
  (https://github.com/ntppool/monitor/client/localok/local-check.go, commit 6005ff4)

  1 if no_response or stratum == 0 then
  2     step = -5
  3 else
  4     if |offset| > 3 or stratum >= 8 then  // 3s
  5         step = -4
  6         if |offset| > 3 then
  7             max_score = -20
  8         end
  9     else if |offset| > 0.75 then          // 750 ms
 10         step = -2
 11     else if |offset| > 0.075 then        // 75 ms
 12         step = -4 * |offset| + 1
 13     else
 14         step = +1
 15     end
 16 end
  ```
NTP Pool Monitoring System

- Monitoring server inspects timeservers approx. every 13 min
  - Each timeserver is scored between 20 to -100

**BAD_SERVER_THRESHOLD = -15**
NTP Pool Monitoring System

• Monitoring results are publicly available
NTP Pool Architecture at a Glance

DNS update

Monitor

NTP

DNS

ch.pool.ntp.org?

NTP Pool

Non-authoritative answer:
Name: ch.pool.ntp.org
Address: 217.147.223.78
Name: ch.pool.ntp.org
Address: 82.197.188.130
Name: ch.pool.ntp.org
Address: 192.33.214.47
Name: ch.pool.ntp.org
Address: 195.141.190.190

Client
NTP Pool Architecture at a Glance
What if an attacker can manipulate the monitoring system?
Exploiting NTP Pool Monitoring System

• Attacker needs to influence time at **many** of the servers assigned to the client

- Inject or compromise 10s or even 100s of timeservers: Ananke[NDSS’21]

- Or… remove legitimate timeservers from the pool **by leveraging the monitoring system**
Attack Modeling

- Exploit the NTP pool monitoring system
- **Exclude legitimate timeservers** from the NTP pool operation
- Silent attack: the target timeservers just turn into **inactive state**
Injecting Asymmetric Delays to Monitoring Packets

1. Hijack monitoring server’s IP prefix
2. Send NTP requests to TSes
3. Reroute NTP replies through the attacker’s network
Injecting Asymmetric Delays to Monitoring Packets

① Hijack monitoring server’s IP prefix
② Send NTP requests to TSes
③ Reroute NTP replies through the attacker’s network
④ Calculate and inject additional delays

\[
\text{Offset}_{TS_i} = \alpha_i / 2
\]

where, \( \alpha_i = \text{Rep}_i - \text{Req}_i \)

\[ \beta_i = \text{target\_delay} - \alpha_i \]
\[ \text{target\_delay} = 500 \text{ ms} \]
Impact of Adding 500 ms of Asymmetric Delay

Achieved target delay (red line)

Logged offsets (red dots) and corresponding score drops (blue line)
Did the Shark Eat the Watchdog in the NTP Pool?  
Deceiving the NTP Pool’s Monitoring System

Jonghoon Kwon  
ETH Zürich

Jeonggyu Song  
Korea University

Junbeom Hur  
Korea University

Adrian Perrig  
ETH Zürich

Abstract
The NTP pool has become a critical infrastructure for modern Internet services and applications. With voluntarily joined thousands of timeservers, it supplies millions of distributed (heterogeneous) systems with time. While numerous efforts have been made to enhance NTP’s accuracy, reliability, and security, unfortunately, the NTP pool attracts relatively little attention. In this paper, we provide a comprehensive analysis of NTP pool security, in particular the NTP pool monitoring system, which oversees the correctness and responsiveness of the participating servers. We first investigate strategic attacks that deceive the pool’s health-check system to remove legitimate timeservers from the pool. Then, through empirical analysis using monitoring servers and timeservers injected into the pool, we demonstrate the feasibility of our approaches, show their effectiveness, and debate the implications. Finally, we discuss designing a new pool monitoring system to mitigate these attacks.

1 Introduction
Time synchronization across distributed systems is essential in modern Internet services and applications, for instance in the validation of certificates [14, 16]. Accurate time is vital also for network infrastructure and its control and data plane operations, e.g., updating routing tables with a precise clock would enable dynamic congestion control and avoid routing loops [2, 36]. The Network Time Protocol (NTP) is packet manipulation and hijacking. Nonetheless, delay attacks and compromised timeservers remain effective. Thus, redesigning NTP received attention to achieve Byzantine robustness even in the presence of adversarial timeservers [8,32].

While numerous efforts to secure NTP communication have been made, only a limited number of studies have focused on the NTP ecosystem [24, 35].

The NTP Pool Project [30], the biggest NTP ecosystem, bundles thousands of public timeservers into regional or vendor-specific domains, and provides NTP clients across the globe centralized access via the domain name service (DNS). With the NTP pool, NTP clients enjoy reliable and available time sources. Indeed, millions of network devices, including routers, IoT devices, and Android mobile devices rely on the NTP pool. Given this critical infrastructure, interesting research questions arise: “What if a determined attacker takes control over the pool?” “What if an attacker is able to remove the majority of legitimate timeservers while keeping malicious timeservers as only sources?” To answer these questions, we conduct an analysis of the current NTP pool architecture and vulnerabilities in its centralized management.

We explore strategic attack approaches against the pool’s health-check system [29]. The NTP pool is a monitoring system that inspects the status of timeservers. It frequently sends NTP changefreq messages, which timeservers check their clock accuracy within a certain tolerance. If the clock is not accurate, timeservers that are removed from the pool.

More in the Paper

NTP Pool architecture
Scoring mechanism
Impact of network delay
New monitoring system

Case Study

Attack Analysis

Mitigation

Robust reference clock
RTT-offset correlation
New scoring algorithm
Summary

• The paper provides a comprehensive analysis of the NTP pool and discloses vulnerabilities in its watchdog system

• We introduce strategic attacks exploiting the vulnerabilities and demonstrate their feasibility

• We present possible mitigations and discussion on securing the NTP pool monitoring system
Hi @newshok,

I think these are all accurate. The solution, I think, will be for the central system to filter the test results to “weed out” intentionally bad results. The opposite could happen too, a monitor trying to keep a poorly functioning server in the system.

More specific answers:

1. If I remember right the score has to be lower before an email is sent. The intention is to minimize emails sent for temporary glitches.
2. Bypassing sanity checking: the monitor operator can also just compile a binary that returns completely made up monitoring results. There isn’t a way around this unless we deploy and manage our own hardware. The solution will be our usual community trust plus some extra sanity checking and consensus verification by the central system.
3. Multiple monitors operated by one party: I expect we’ll allow many monitors (about 15-25?) but then limit who can add more to (for example) users who have had servers in the system for a relatively long time. Maybe by default each user / account will only be allowed to add for example two monitors, and we can limit the overlap between servers they in turn monitor. We have a similar potential issue with the NTP servers (which will have different mitigations).

Thank you for thinking about this!
Q&A

Jonghoon Kwon
Network Security Group
jong.kwon@inf.ethz.ch

CAB F83
Universitätstrasse 6
8092 Zürich, Switzerland

https://netsec.ethz.ch