Attacking the Network Time Protocol (NTP)

Aanchal Malhotra
Isaac E. Cohen, Erik Brakke
Sharon Goldberg

NDSS, 2016
Outline of the talk

• **Background**
  - How does NTP work?
  - How does NTP client take time?

• **Our attacks**
  - Denial of Service by Spoofed Kiss-of-Death (off-path)
  - Denial of Service by Priming the Pump (off-path)
  - Timeshifting by IPv4 Packet Fragmentation (off-path)
Background: How does NTP work?

- Sends queries at randomized and adaptively selected intervals
- Requires certain number of self-consistent responses to update its clock

- Every host can act as both a client and a server
- My laptop will answer queries from the public Internet
We assume NTP messages are not cryptographically authenticated.
(Ask me why after.)

We attack the NTPv4 spec (RFC5905)
and its reference implementation
(ntpd v4.2.8p2 & ntpd v4.2.6p5)
Non-Crypto Authentication with Origin Timestamp ($T_1$)

**TEST2:** Match $T_3$ in Query to $T_1$ in Response.

How much entropy is in Origin Timestamp ($T_1$)?

≈ 32 bits!

*ntpd does not randomize UDP source port!

Analogous to
- UDP source port randomization
- TCP sequence no randomization

<table>
<thead>
<tr>
<th>v4</th>
<th>IHL=20</th>
<th>TOS</th>
<th>Total length = 76</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source Port = 123

Destination Port = 123

LI | v4 | Response | Stratum | Poll | Precision |
---|----|----------|---------|------|-----------|
    |    |          |         |      | Root Delay |
    |    |          |         |      | Root Dispersion |
    |    |          |         |      | Reference ID |
    |    |          |         |      | Reference Timestamp |

$T_1 = Origin Timestamp$

$T_2 = Receive Timestamp$

$T_3 = Transmit Timestamp$
Outline of the talk

• Background
  ➢ How does NTP work?
  ➢ How does NTP client take time?

• Our attacks
  ➢ Denial of Service by Spoofed Kiss-of-Death (off-path)
  ➢ Denial of Service by Priming the Pump (off-path)
  ➢ Timeshifting by IPv4 packet fragmentation (off-path)
# Denial of Service via Spoofed Kiss-o-Death

**Kiss-o’-Death (KoD)**

"Keep quiet for 2^pol^ sec!"

(36 hours!)

<table>
<thead>
<tr>
<th>Version</th>
<th>IHL=20</th>
<th>TOS</th>
<th>Total length = 76</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>Protocol = 17</td>
<td>IP Header Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Port = 123</td>
<td>Destination Port = 123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length = 76</td>
<td>UDP Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LI</td>
<td>Response</td>
<td>Stratum</td>
<td>Poll</td>
</tr>
<tr>
<td>Poll</td>
<td>Poll</td>
<td>Poll</td>
<td></td>
</tr>
<tr>
<td>Root Delay</td>
<td>Root Dispersion</td>
<td>Reference ID = RATE</td>
<td></td>
</tr>
<tr>
<td>Reference Timestamp = Jan 1, 1970 0:00:00 UTC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TEST2?**

One packet prevents client from querying its servers for days or years!

1. **T1** = Origin Timestamp = July 29, 2015 01:23:45
2. **T2** = Receive Timestamp = July 29, 2015 01:23:45
3. **T3** = Transmit Timestamp = July 29, 2015 01:23:45

"Keep quiet for 2^pol^ sec!"
How to learn the server’s IP for the spoofed KoD?

An attacker can deactivate NTP for the whole Internet within hours / days with one machine!
Denial of Service by Priming-the-Pump

1. Denial of Service by Spoofed Kiss-of-Death (off-path)

2. Denial of Service by Priming the Pump (off-path)

Patched!

ntpd 4.2.8p4
Outline of the talk

• Background
  ➢ How does NTP work?
  ➢ How does NTP client take time?

• Our attacks
  ➢ Denial of Service by Spoofed Kiss-of-Death (off-path)
  ➢ Denial of Service by Priming the Pump (off-path)
  ➢ Timeshifting by IPv4 packet fragmentation (off-path)
Background: IPv4 Packet Fragmentation
How Our Attacker Uses IPv4 Packet Fragmentation?

Query

IPv4 packet fragmentation needed to 68 bytes

Client

Server

Off-path attacker

Client buffer

Origin Timestamp

IPID=1

52 bytes

8 bytes

16 bytes

IPID=1

LF1

68 bytes

IPID=1

LF2

8 bytes

IPID=1

SF1

52 bytes

IPID=1

SF2

16 bytes

IPv4 packet fragmentation needed to 68 bytes
Reassembled Packet

Key Challenge: Pass TEST2!
Craft a stream of packets where $T_2 - T_1$ is consistent within 1 sec!

$T_2 - T_1 = -10\text{ years} + 52\text{ sec}$
Conditions for the Attack

- Server must fragment NTP packets to 68 bytes
  - Scanned 13M servers
  - About 24K servers were willing to fragment to 68-byte

- Client reassembles overlapping fragments according to First policy
  - The client prefers fragments that arrive earliest
    (We can not safely measure because of teardrop [CA-1997-28])

- Server uses incrementing IPID
  - attacker can infer IPID using techniques explained in
    [Gilad, Herzberg’2013] and [Knockell, Crandall’2014]
Summary, Recommendations & Impact

- **Attack: DoS by spoofed KoD:**
  - Rec: Implement TEST2 *(patched in v4.2.8p4 & NTPSec & Cisco & RedHat Linux etc.)*

- **Attack: DoS by priming the pump:**
  - Rec: Authentication in both directions *(IETF Network Time Security draft updated)*
    - client ➔ server & server ➔ client
  - Rate limit like Response Rate Limiting (RRL) in DNS *(under discussion)*

- **Attack: Time shifting by IPv4 Packet Fragmentation:**
  - Rec: Server should not fragment to 68 bytes *(Test your server on our site)*
  - Clients should drop overlapping fragments

- **Other recommendations:**
  - Stop my laptop from answering timing queries
  - More work on cryptography for NTP
Thank You!

Questions ?