Complex Paths and Derelict Sentinels
software engineering underpinnings of recent NTP vulnerabilities

Matthew Van Gundy <mvangund@cisco.com>
Technical Leader, Cisco Advanced Security Initiatives Group (ASIG)
LangSec Workshop 2016
All opinions expressed are my own, not those of Cisco.
How NTP Works

- Every host is both a client and server
- Your laptop will answer queries from the public Internet

(ntp.conf
server a
server b
server c)
Preventing Off-Path Impersonation Attacks

NTP Packet

<table>
<thead>
<tr>
<th>LI</th>
<th>Ver</th>
<th>Mode</th>
<th>Stratum (8)</th>
<th>Poll (8)</th>
<th>Precision (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root delay (32)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root dispersion (32)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Clock Id (32)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Clock Timestamp (64)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1: Origin Timestamp (64)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2: Receive Timestamp (64)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3: Transmit Timestamp (64)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyid (32, optional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digest (128+, optional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- No source port randomization
- TEST2: Drop packet unless T3 in query == T1 in response
- Transmit timestamp has ≈ 32-bits entropy
- Similar to TCP sequence number randomization
def receive( pkt ):
    # ...
    if pkt.T3 == 0:
        flash |= test3  # fail test3
    elif pkt.T3 == org:
        flash |= test1  # fail test1
    return
    elif broadcast == True:
        pass  # skip further tests
    elif interleave == False:
        if pkt.T1 == 0:
            xmt = 0
        elif (xmt == 0 or pkt.T1 != xmt):
            flash |= test2  # fail test2
        if (rec != 0 and pkt.T1 == rec):
            interleave = True
            return
        else:
            xmt = 0  # pass test2, clear xmt
    elif (pkt.T1 == 0 or pkt.T2 == 0):
        flash |= test3  # fail test3
    elif (rec != 0 and rec != pkt.T1):
        flash |= test2
        return  # fail interleave test2
    if auth in { ERROR, CRYPTO } \ 
        or (need_auth and auth != OK):
        return
    if interleave == False:
        rec = pkt.receive_time()
    org = pkt.T3
    if flash == True:
        return
    else:
        process( pkt )
    if interleave == True:
        rec = pk.receive_time()
Crypto-NAK Packets

• “Ephemeral associations are mobilized upon the arrival of a packet and are **demobilized upon error** or timeout.”

• Authentication errors elicit a crypto-NAK response

• Handled “late”, during other packet consistency checks

• Authentication states:
  
  { NONE, OK, ERROR, CRYPTO }

<table>
<thead>
<tr>
<th>NTP Crypto-NAK Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LI</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>Root delay (32)</td>
</tr>
<tr>
<td>Root dispersion (32)</td>
</tr>
<tr>
<td>Reference Clock Id (32)</td>
</tr>
<tr>
<td>Reference Clock Timestamp (64)</td>
</tr>
<tr>
<td>T1: Origin Timestamp (64)</td>
</tr>
<tr>
<td>T2: Receive Timestamp (64)</td>
</tr>
<tr>
<td>T3: Transmit Timestamp (64)</td>
</tr>
</tbody>
</table>

Keyid (32, optional) == 0x00000000

Digest (128+, optional)
NAK to the Future Vulnerability (CVE-2015-7871)

- Most ephemeral associations
  - auth in {ERROR, CRYPTO}: reject
  - auth == NONE: reject if auth required
  - else: mobilize

- Symmetric active mode packets
  - auth in {NONE, ERROR}: Special handling for certain broken clients
  - else: mobilize
  - (auth == CRYPTO): crypto-NAK packets mobilize new symmetric associations

- keyid == 0: Unauthenticated association

Unauthenticated Symmetric NTP

crypto-NAK

Peers:
- server
- attacker
Refclock spoofing

127.127.1.1

127.127.1.1

10.1.1.2

© 2016 Cisco and/or its affiliates. All rights reserved.