Stalloris: RPKI Downgrade Attack

Tomas Hlavacek, Philipp Jeitner, Donika Mirdita, Haya Shulman and Michael Waidner

German National Research Center for Applied Cybersecurity ATHENE
Fraunhofer Institute for Secure Information Technology SIT
Technical University Darmstadt
Goethe-University Frankfurt
A Short Introduction to RPKI

- **Manifest** lists all signed objects in Publication Point repository
- **Route Origin Authentication (ROA)** signed pair of (IP Prefix Block, ASN)
Downgrade RPKI: Low-Rate Attack

1. Relying Party (RP) connects to the attacker's Publication Points (PP)
2. Attacker makes target PP unreachable via rate-limiting
   a. Spoof TCP SYN packets to overload PP
   b. Spoof DNS queries to nameserver
3. Queries from RP and its resolvers go unanswered, repeat periodically until
4. Objects in RP cache expire
   a. ROAs of target PP are no longer available
   b. BGP Router gets incomplete data => RPKI Downgrade
1. Victim RP sends request to Attacker PP
2. Attacker PP stalls the victim RP until timeout
3. Victim RP traverses the delegation tree of the attacker
   ▪ Stalling time is size_of_tree x timeout
4. Stalling persists until cached manifest times out
5. ROAs from expired manifests no longer available
   ▪ Route RPKI status in router switches: Valid -> Not Found
Vulnerabilities in the RPKI Environment

Rate-Limiting in DNS

- Rate-limiting in nameservers & PPs
- 47% of PPs vulnerable
- 20.4% of IPv4 space vulnerable

RP Predictability

- Regular refresh intervals (ex. Routinator)
- 70% of MFTs < 48h validity

Unlimited Delegation Chain

- Infinite (re-)delegation of same resources
Attack Feasibility

Refresh Intervals for RPs are deterministic

Packet volume for successful attack

Open Source Software to build attacker’s malicious Repo Tree

<table>
<thead>
<tr>
<th>(Scenario)</th>
<th>$r_{\text{attempts}}$</th>
<th>$l_{\text{attack}}$</th>
<th>$l_{\text{deep}}$</th>
<th>$\omega_{\text{start}}$</th>
<th>$\omega_{\text{start}}$</th>
<th>$\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>24</td>
<td>unbound (blocked)</td>
<td>900 s</td>
<td>unbound (blocked)</td>
<td>35</td>
<td>35.0</td>
</tr>
<tr>
<td>(2)</td>
<td>864</td>
<td>fresh manifest 1 day</td>
<td>routinator (normal)</td>
<td>600 s</td>
<td>1247</td>
<td>1247.0</td>
</tr>
<tr>
<td>(3)</td>
<td>23040</td>
<td>long-valid manifest 2 days</td>
<td>RIPE NCC validator</td>
<td>120 s</td>
<td>unbound normal 16</td>
<td>33240.0</td>
</tr>
<tr>
<td>(4)</td>
<td>55</td>
<td>fresh manifest 1 day</td>
<td>routinator (stalled)</td>
<td>2.6 hours</td>
<td>linux x64</td>
<td>80.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(Scenario)</th>
<th>$\omega$</th>
<th>$r_{\text{limit}}$</th>
<th>$r_{\text{attacker}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>3</td>
<td>60</td>
<td>105</td>
</tr>
<tr>
<td>35</td>
<td>1288</td>
<td>45,080</td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>3</td>
<td>60</td>
<td>3,741</td>
</tr>
<tr>
<td>1247</td>
<td>1288</td>
<td>74,820</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>3</td>
<td>60</td>
<td>99,720</td>
</tr>
<tr>
<td>33240</td>
<td>1288</td>
<td>1,994,400</td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td>3</td>
<td>60</td>
<td>240</td>
</tr>
<tr>
<td>80</td>
<td>1288</td>
<td>4,800</td>
<td></td>
</tr>
<tr>
<td>(S)</td>
<td>3</td>
<td>60</td>
<td>103,040</td>
</tr>
<tr>
<td>80</td>
<td>1288</td>
<td>4,800</td>
<td></td>
</tr>
</tbody>
</table>
Results

- Rate-Limiting affects almost half of all Publication Points
- An attacker has all the available open source tools necessary to do an optimized Stalloris Rate-Limiting attack
- Relying Party do not provide feedback when something abnormal is happening: Stalloris can only be detected via manual log checking
- Attack effectively bypasses RPKI protection of prefixes despite RPKI being correctly implemented by client and user alike
Thank you for your attention!

If you have any questions, contact at 

donika.mirdita@sit.fraunhofer.de