Mercury

Bandwidth-effective Prevention of Rollback Attacks Against Community Repositories

Trishank Karthik Kuppusamy, Vladimir Diaz, Justin Cappos

NYU Tandon School of Engineering
Software repositories
Software updates

- Experts agree: software updates the most important thing (USENIX SOUPS 2015)
- Updates fix security vulnerabilities
- However, important problem in software updates often neglected...

“...no one can hack my mind”: Comparing Expert and Non-Expert Security Practices

---

Figure 1: Security measures mentioned by at least 5% of each group. While most experts said they keep their system updated and use two-factor authentication to stay safe online, non-experts emphasized using antivirus software and using strong passwords.
Repository compromise: impact

- Nation state actors:
  - Microsoft Windows Update (2012): Flame malware targeted Iran nuclear efforts
  - South Korea cyberattack (2013): >$750M USD in economic damage
  - NotPetya (2017): infected multinational corporations

- Compromise millions of devices
- Worst case: human lives
SSL / TLS

- Use online key to sign all updates (e.g., SSL / TLS, CUP)
- Protects users from man-in-the-middle attacks
The problem with SSL / TLS

- Doesn’t say anything about the security of the server: just the connection
- Single point of failure: easy to compromise
- If repository is compromised, attacker can install malware and control devices
GPG / RSA

- Why not sign updates using GPG / RSA keys kept off repository?
GPG / RSA

- Why not sign updates using GPG / RSA keys kept off repository?
- Assumes key distribution problem solved, but OK...
- Mission accomplished, right?


Why Johnny Can’t Encrypt: A Usability Evaluation of PGP 5.0

Alma Whitten
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
alma@cs.cmu.edu

J. D. Tygar
EECS and SIMS
University of California
Berkeley, CA 94720
tygar@cs.berkeley.edu
What do these organizations have in common?
Vulnerabilities in software updates
Goal: compromise-resilience

- Only a question of when, not if
- Cannot prevent a compromise
- But must severely limit its impact
One way GPG / RSA is insecure
Project metadata & packages

- A repository has many projects
- A project has many packages
- A project signs a metadata file listing packages
Rollback attacks

- Choose obsolete updates with known security vulnerabilities
Rollback attacks

- No need to tamper with signed updates
- Just replace new signed updates with old signed updates!
Why rollback attacks are bad

- Compromise users w/o tampering with updates! [CCS 2008]
- Obsolete = vulnerable = just as bad as malware

A Look In the Mirror: Attacks on Package Managers

Justin Cappos  Justin Samuel  Scott Baker  John H. Hartman
Department of Computer Science, University of Arizona
Tucson, AZ 85721, U.S.A.
{justin, jsamuel, bakers, jhh}@cs.arizona.edu
Prevents rollback attacks on installed projects

- Verify project metadata to verify packages
- Download project metadata for only package to be installed
- Compare previous & current version numbers of project metadata

\[\text{prev} > \text{curr}!\]
What about projects *yet to be installed*?

- **BAD!** Does not prevent rollback attacks on projects *yet to be installed*
- What is the previous version number?

---

<table>
<thead>
<tr>
<th>previous</th>
<th>current</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3</td>
<td>v6</td>
</tr>
<tr>
<td>Django</td>
<td>Django-1.7.tar.gz</td>
</tr>
<tr>
<td>Bcrypt</td>
<td>Bcrypt-0.1.tar.gz</td>
</tr>
</tbody>
</table>

(prev ≤ curr ?) vs (prev > curr !)
Compromise-resilience with Diplomat
The Update Framework (TUF)

- **Design principles**
  - Separation of duties
  - Threshold signatures
  - Explicit & implicit revocation of keys
  - Minimizing risk using offline keys
  - Selective delegation of trust
- **CCS 2010**
Diplomat

- Provides compromise-resilience & immediate project registration
- USENIX NSDI 2016
Snapshot metadata

- Repositories distribute snapshot metadata, or manifest of all projects
Download snapshot metadata

- To prevent rollback attacks, first download snapshot metadata
Download project metadata

- Then, compare previous & current version number of project metadata
Download *all* project metadata

- Do this for every single project metadata file listed in snapshot metadata
Integrations & deployments

- Docker
- Core OS
- Flynn
- R
- Python
- OCaml
- Ruby
Problem

- Diplomat too expensive on some repositories like PyPI
- A large number of frequently updated projects
Bandwidth cost for new users

- Requires *new* users to download *all* project metadata
- 20MB (31x!)
Bandwidth cost for returning users

- Requires *returning* users to download all *new or updated* project metadata
- 2.1MB (3.2x!)
Mercury: a new security system
Diplomat: repository cannot be trusted

- No trusted party (e.g., humans) to always correctly indicate new project metadata
- Projects are updated too rapidly
Diplomat: repository cannot be trusted

- Repositories use automation to indicate only *which* projects have been updated
Diplomat: repository cannot be trusted

- But attackers who compromise repository can launch rollback attacks
- Just point to obsolete project metadata!
Diplomat: only developers can be trusted

- Only developers trusted to provide version numbers
- Price: prohibitive b/w costs
Mercury: shift trust from developers to repository

- Safely shift source of trust from developers to repository
- Snapshot metadata indicates version numbers of project metadata
Mercury: low bandwidth cost

- Uses low bandwidth costs
- To prevent rollback attacks, first download snapshot metadata
Mercury: low bandwidth cost

- Download project metadata for only package to be installed
- Use delta compression for more savings
Security analysis

- But is it secure?
Security analysis: rollback attacks

- Mercury always prevents rollback attacks
Security analysis: rollback attacks

- Always compare previous & current version numbers in snapshot metadata
Security analysis: rollback attacks

- Do not delete projects from snapshot metadata
- Otherwise, attackers can rollback these projects
Security analysis: fast-forward attacks

- Unlike Diplomat, susceptible to fast-forward attacks
Security analysis: fast-forward attacks

- Arbitrarily increase version numbers in snapshot metadata
- Can deny packages to users
Security analysis: fast-forward attacks

- Waste b/w by setting arbitrarily large version numbers

![Version numbers of software projects graph]

999999999... 99999999
Security analysis: fast-forward attacks

- Increase version numbers to MAXINT
- Makes recovery impossible
Recovering from fast-forward attacks

- Revoke and replace keys used to sign snapshot metadata
- Discard and replace snapshot metadata
Recovering from fast-forward attacks

<table>
<thead>
<tr>
<th>System / Cost</th>
<th>Common case</th>
<th>Rare case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diplomat</td>
<td>More expensive</td>
<td>Less complicated</td>
</tr>
<tr>
<td>Mercury</td>
<td>Less expensive</td>
<td>More complicated</td>
</tr>
</tbody>
</table>

Explicit & implicit revocation of keys

Design principles:
1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
Persistent Mirror +
Developer Compromise
Protection against malicious mirrors

- Malicious mirrors in powerful nation-states
- Cannot sign new snapshot metadata, but can sign some new project metadata
- Can switch project metadata w/o getting caught
Protection against malicious mirrors

- Mercury-hash: hash + version number in snapshot metadata
- Malicious mirrors cannot switch project metadata w/o getting caught
- Higher b/w cost
Evaluation of bandwidth costs
Experimental setup

● Security systems
  ○ GPG / RSA — insecure!
  ○ Mercury
  ○ Mercury-hash
  ○ Diplomat-version: projects sign detached version numbers
  ○ Diplomat

● An anonymized log of a month of package downloads from PyPI
## Bandwidth overhead by security system

<table>
<thead>
<tr>
<th></th>
<th>Initial cost</th>
<th>Recurring cost</th>
<th>Recovery cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPG/RSA</td>
<td>0.6KB (0.1%)</td>
<td>0.02KB (0.003%)</td>
<td>N/A</td>
</tr>
<tr>
<td>Mercury</td>
<td>319KB (48%)</td>
<td>23KB (3.5%)</td>
<td>320KB (48%)</td>
</tr>
<tr>
<td>Mercury-hash</td>
<td>2.4MB (360%)</td>
<td>156KB (24%)</td>
<td>2.4MB (361%)</td>
</tr>
<tr>
<td>Diplomat-version</td>
<td>7.6MB (1,152%)</td>
<td>1.1MB (171%)</td>
<td>2.3MB (350%)</td>
</tr>
<tr>
<td>Diplomat</td>
<td>20MB (3,092%)</td>
<td>2.1MB (320%)</td>
<td>2.3MB (350%)</td>
</tr>
</tbody>
</table>
Bandwidth overhead by security system

<table>
<thead>
<tr>
<th></th>
<th>Initial cost</th>
<th>Recurring cost</th>
<th>Recovery cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPG/RSA</td>
<td>0.6KB (0.1%)</td>
<td>0.02KB (0.003%)</td>
<td>N/A</td>
</tr>
<tr>
<td>Mercury</td>
<td>319KB (48%)</td>
<td>23KB (3.5%)</td>
<td>320KB (48%)</td>
</tr>
<tr>
<td>Mercury-hash</td>
<td>2.4MB (360%)</td>
<td>156KB (24%)</td>
<td>2.4MB (361%)</td>
</tr>
<tr>
<td>Diplomat-version</td>
<td>7.6MB (1,152%)</td>
<td>1.1MB (171%)</td>
<td>2.3MB (350%)</td>
</tr>
<tr>
<td>Diplomat</td>
<td>20MB (3,092%)</td>
<td>2.1MB (320%)</td>
<td>2.3MB (350%)</td>
</tr>
</tbody>
</table>
Bandwidth overhead by security system

<table>
<thead>
<tr>
<th></th>
<th>Initial cost</th>
<th>Recurring cost</th>
<th>Recovery cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPG/RSA</td>
<td>0.6KB (0.1%)</td>
<td>0.02KB (0.003%)</td>
<td>N/A</td>
</tr>
<tr>
<td>Mercury</td>
<td>319KB (48%)</td>
<td>23KB (3.5%)</td>
<td>320KB (48%)</td>
</tr>
<tr>
<td>Mercury-hash</td>
<td>2.4MB (360%)</td>
<td>156KB (24%)</td>
<td>2.4MB (361%)</td>
</tr>
<tr>
<td>Diplomat-version</td>
<td>7.6MB (1,152%)</td>
<td>1.1MB (171%)</td>
<td>2.3MB (350%)</td>
</tr>
<tr>
<td>Diplomat</td>
<td>20MB (3,092%)</td>
<td>2.1MB (320%)</td>
<td>2.3MB (350%)</td>
</tr>
</tbody>
</table>
### Bandwidth overhead by security system

<table>
<thead>
<tr>
<th>Security Feature</th>
<th>Total initial costs of new users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packages</td>
<td>2.2TB</td>
</tr>
<tr>
<td>GPG/RSA</td>
<td>0.005TB (0.2%)</td>
</tr>
<tr>
<td>Mercury</td>
<td>0.4TB (17%)</td>
</tr>
<tr>
<td>Mercury-hash</td>
<td>2.8TB (125%)</td>
</tr>
<tr>
<td>Diplomat-version</td>
<td>8.9TB (396%)</td>
</tr>
<tr>
<td>Diplomat</td>
<td>23.9TB (1,067%)</td>
</tr>
</tbody>
</table>
Bandwidth vs. number of projects
Bandwidth vs. rate of project updates

The average number of projects updated per minute

Recurring cost as rate of project updates is varied

- Diplomat
- Diplomat-version
- Mercury-hash
- Mercury
- GPG/RSA

Average downloaded package size
Actual rate of updates over the month
Conclusions
Takeaways

- Safely shift trust from developers to repository
- Common case less expensive, but rare case slightly more complicated
- Practical use uncovers problems
Integrations & deployments
Q & A

Thanks! Questions?

https://theupdateframework.github.io/

https://uptane.github.io/

trishank@nyu.edu