Rough times? TUF shines

A Framework for Secure Software Updates

Trishank Karthik Kuppusamy, Vladimir Diaz, Sebastien Awwad
Lukas Pühringer, Justin Cappos
Software updates

Experts agree that software updates are the most important thing to stay safe [USENIX SOUPS 2015]

Updates fix security vulnerabilities

However, an important problem in software updates is 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.
A compromise can have enormous impacts

- Nation state actors
- Microsoft Windows Update (2012): Flame malware targeted Iran nuclear efforts
- NotPetya (2017): infected multinational corporations
- Compromise millions of devices
- Worst case: human lives
Just sign it, ... right?
SSL / TLS (online key)

→ Protects users from man-in-the-middle attacks
The problem with SSL / TLS

➔ Doesn’t say anything about the security of the server
➔ Single point of failure: easy to compromise
GPG (offline key)

➔ Why not sign updates using offline GPG?
➔ Assuming usability and key distribution problem solved...
➔ Mission accomplished, right?
What do these organizations have in common?
Vulnerabilities in software updates
Only question is when not if a compromise happens
A Look in the Mirror: Attacks on Package Managers

➔ Survey of package managers [CCS 2008]
➔ Many package managers had bad security
➔ APT did better than most
➔ But still had problems!
Endless Data Attack

Serve update until storage is full
Freeze Attack

Trick updater into believing that there are no updates available
Replay Attack

Serve obsolete packages that might have vulnerabilities
So why TUF?
The Update Framework

→ Not every software updater needs an in-house solution
→ Many years of experience in secure software updates
→ Shields against a variety of attacks
→ Minimizes impact of key compromise
Responsibility Separation

- Root of trust
- Content
- Consistency
- Timeliness
Minimize individual Key and Role Risk

\[ \text{DAMAGE} \approx \text{PROBABILITY} \times \text{IMPACT} \]

- High-impact role? → Highly secure keys
- Online keys? → Low-impact role
Multi-signature Trust (Thresholds)

```json
{
"_type": "root",
"compression_algorithms": [ ... ],
"consistent_snapshot": true,
"version": VERSION,
"expires": EXPIRES,
"keys": {
  KEYID : KEY
  ...
},
"roles": {
  ROLE : {
    "keyids": [ KEYID, ... ],
    "threshold": THRESHOLD
  }
}
}```
Explicit and implicit Revocation

Revocation

Expiration
TUF Roles Overview

Root (root of trust)  Timestamp (timeliness)  Snapshot (consistency)  Targets (integrity)
Deployment?
Server (repository)

➔ Use TUF repository tools to manage keys and metadata

➔ Generate keys for each role

➔ Keep them offline

➔ Upload signed metadata + packages to Debian server
Client (package manager)

➔ Modify update client to use TUF client updater (just ship out with root metadata)

➔ Automatically & transparently download & verify packages

➔ Users won’t see difference

➔ Except when attacks occur
Conclusions

- Works with existing software updater
- Prevents from a variety of attacks
  (arbitrary software, endless data, extraneous dependencies, fast-forward, freeze, mix-and-match, rollback, slow retrieval, wrong software)
- Key compromise-resilient
- No out-of-band PKI or web of trust required
- Spin-offs and adoptions already exist
Deployments & Integrations
Thank You! Questions?

https://theupdateframework.github.io/
jcappos@nyu.edu