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

Iulia Ion
Google
iuliaion@google.com

Rob Reeder
Google
rreeder@google.com

Sunny Consolvo
Google
sconsolvo@google.com

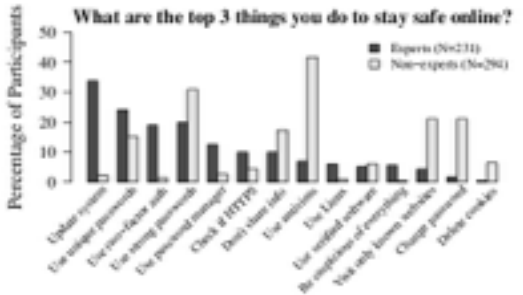


Figure 1: Security measures mentioned by at least 5% of each group. While most experts said they keep their system updated and use two-factor authentication to stay safe online, non-experts emphasized using antivirus software and using strong passwords.

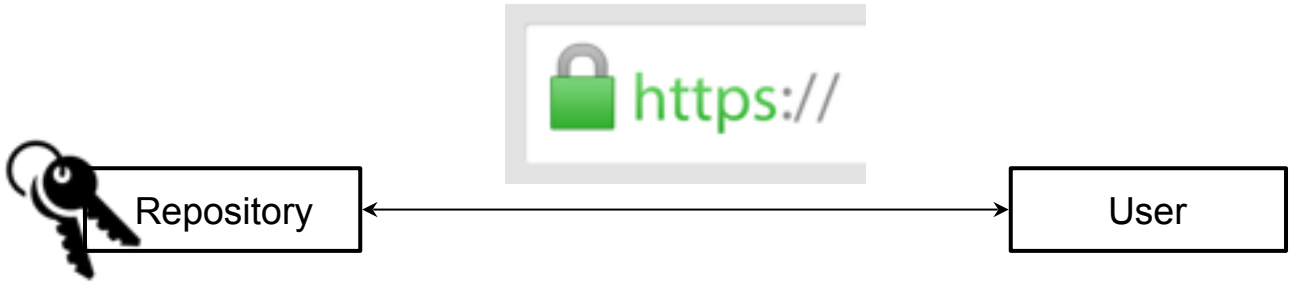
A compromise can have enormous impacts

- ➔ 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?

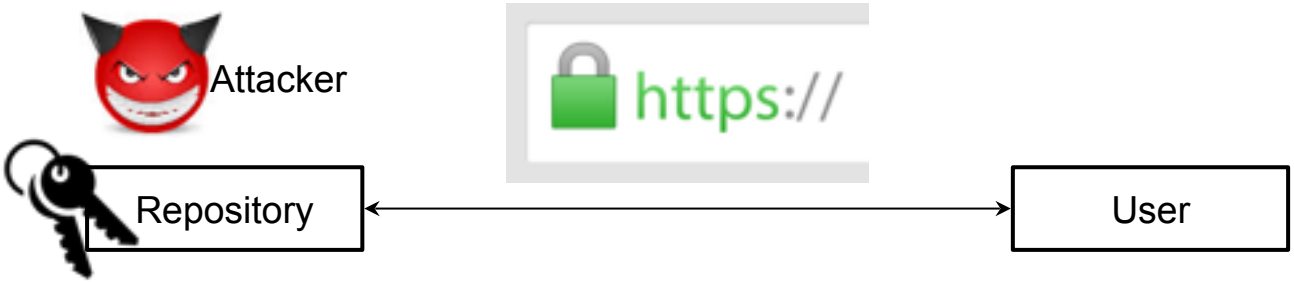
SSL / TLS (online key)

➔ Protects users from man-in-the-middle attacks



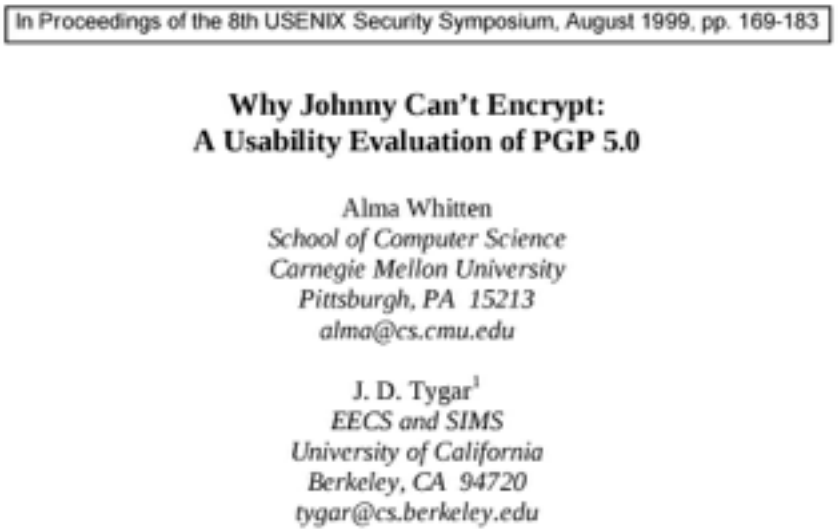
The problem with SSL / TLS

- ➔ 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?



In Proceedings of the 8th USENIX Security Symposium, August 1999, pp. 169-183

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[1]
EECS and SIMS
University of California
Berkeley, CA 94720
tygar@cs.berkeley.edu

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!

A Look In the Mirror: Attacks on Package Managers

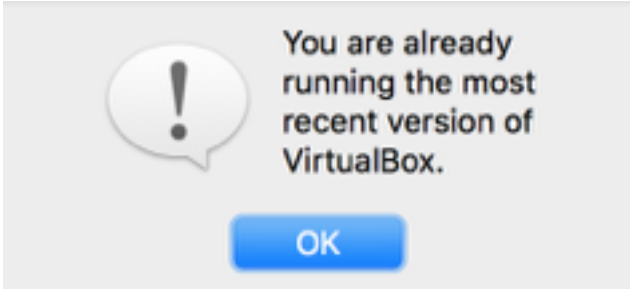
Justin Cappos Justin Samuel Scott Baker John H. Hartman
Department of Computer Science, University of Arizona
Tucson, AZ 85721, U.S.A.
{justin, jsamuel, bakers, jhh}@cs.arizona.edu

Endless Data Attack

Serve update until storage is full

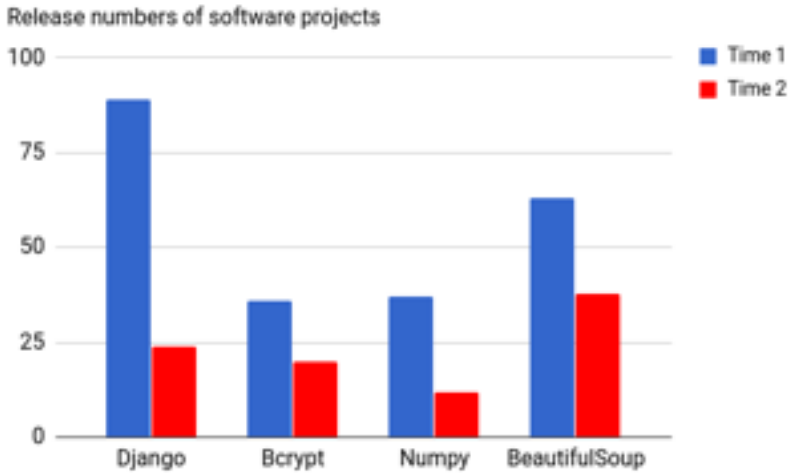
Freeze Attack

Trick updater into believing that there are no updates available



Replay Attack

Serve obsolete packages that might have vulnerabilities

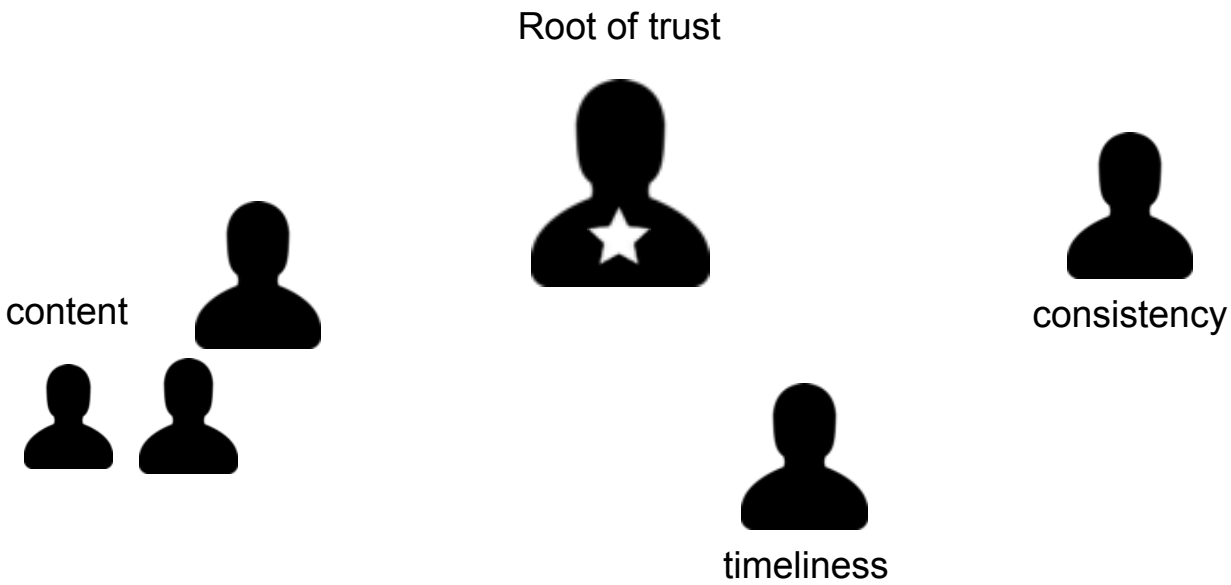


So why TUF?

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



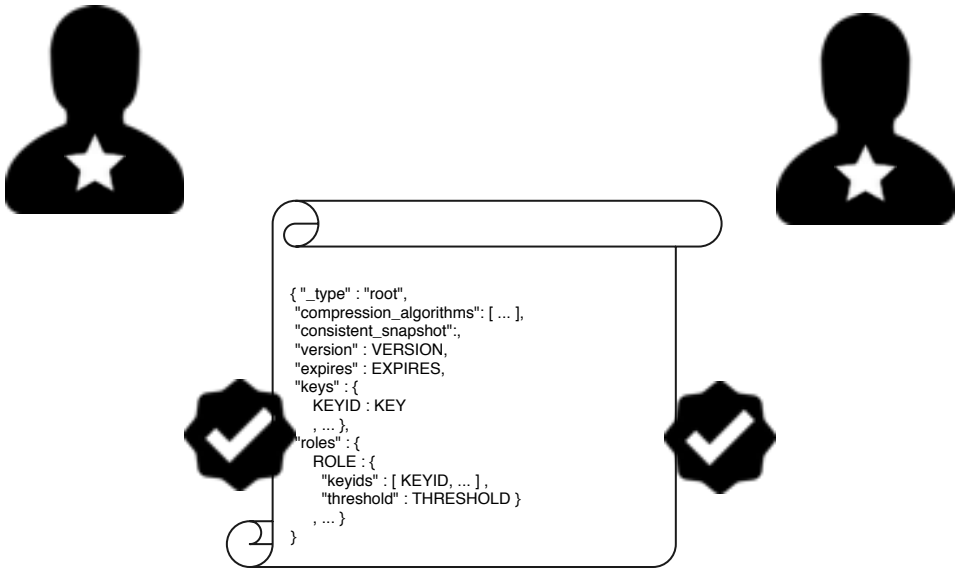
Minimize individual Key and Role Risk

DAMAGE ~= PROBABILITY x IMPACT

High-impact role? ⟶ Highly secure keys

Online keys? ⟶ Low-impact role

Multi-signature Trust (Thresholds)



Explicit and implicit Revocation

Revocation

Expiration

TUF Roles Overview

Root

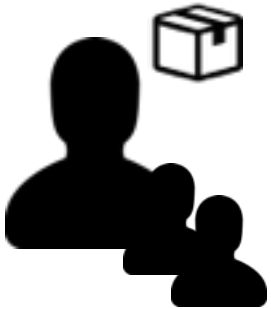
(root of trust)

Timestamp

(timeliness)

Snapshot

(consistency)



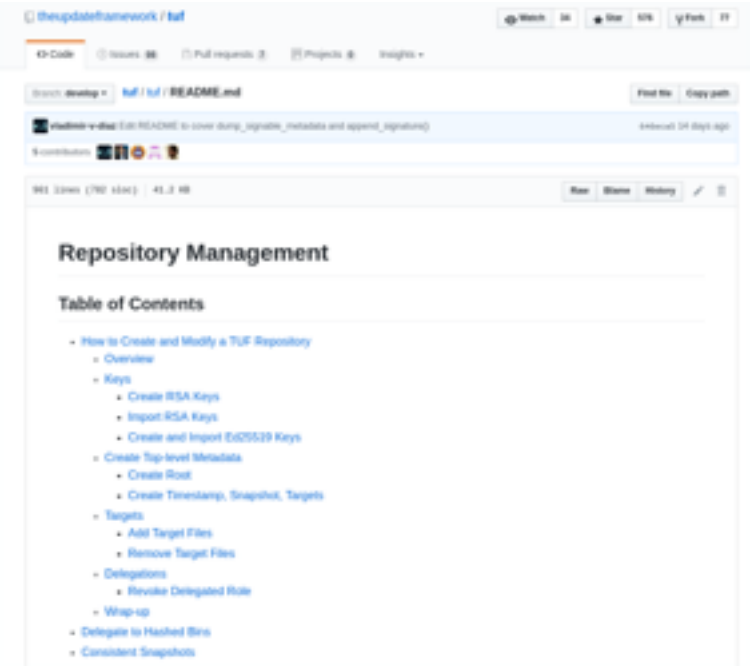
Targets

(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

Interposition Examples

To use interposition, integrators must:

1. Create an interposition configuration file.
2. Import interposition, and load the configuration file with configure().
3. Perform updater urllib calls that may be interposed.
4. Deconfigure interposition.

Option 1

```
from tuf.interposition import urllib_tuf as urllib
from tuf.interposition import urllib2_tuf as urllib2

# configure() loads the interposition configuration file that indicates which
# URLs should be interposed by TUF.  Any urllib calls that occur after
# configure() are subject to interposition.

configuration = tuf.interposition.configure()

url = "http://example.com/path/to/document"

urllib.urlopen(url)
urllib.urlretrieve(url, 'mytarget')
urllib2.urlopen(url)

# deconfigure() is used to stop interposition.  Any urllib calls that occur
# after deconfigure() are not interposed.
tuf.interposition.deconfigure(configuration)
```

Option 2

```
@tuf.interposition.open_url
def instancemethod(self, url, ...):
  ...
```

Note: tuf.interposition.refresh(configuration) may be called to force a refresh of the TUF metadata. Interposition normally performs a refresh of TUF metadata when configure() is called.

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