Exploiting Insecurity to Secure Software Update Systems

Justin Cappos

Department of Computer Science and Engineering
University of Washington

Introduction

software update system -- a piece of software that installs, updates, removes, or patches software or firmware on a device by retrieving information (software updates) from a trusted, external source (repository)

Software update systems are widely insecure [Bellissimo HotSec 06, Cappos CCS 08]

Software update systems are ubiquitous

But security is simple, right?

Just use HTTPS

- Common errors in how certificates are handled
- Online data becomes single point of weakness

... and add signatures to the software updates

- Attackers can perform a replay attack

... and add version numbers to the software updates

- Attackers can launch freeze attacks

But security is simple, right? (cont.)

..... and add a quorum of keys signature system for the root of trust, add signing by different compartmentalized key types, use online keys only to provide freeze attack protection and bound their trust window, etc. [Thandy software updater for Tor]

We still found 8 design or implementation flaws

Having each developer build their own "secure" software update system will fail

Is there a practical risk?

PlanetLab uses YUM -- updates come both from Fedora 9 and PLC

Lease a server and have it listed as an official Fedora mirror

Ensure that PlanetLab nodes contact only your mirror

Find existing exploit code for an old version of a package that isn't installed

Change the package metadata so the old version of the package is installed with any update

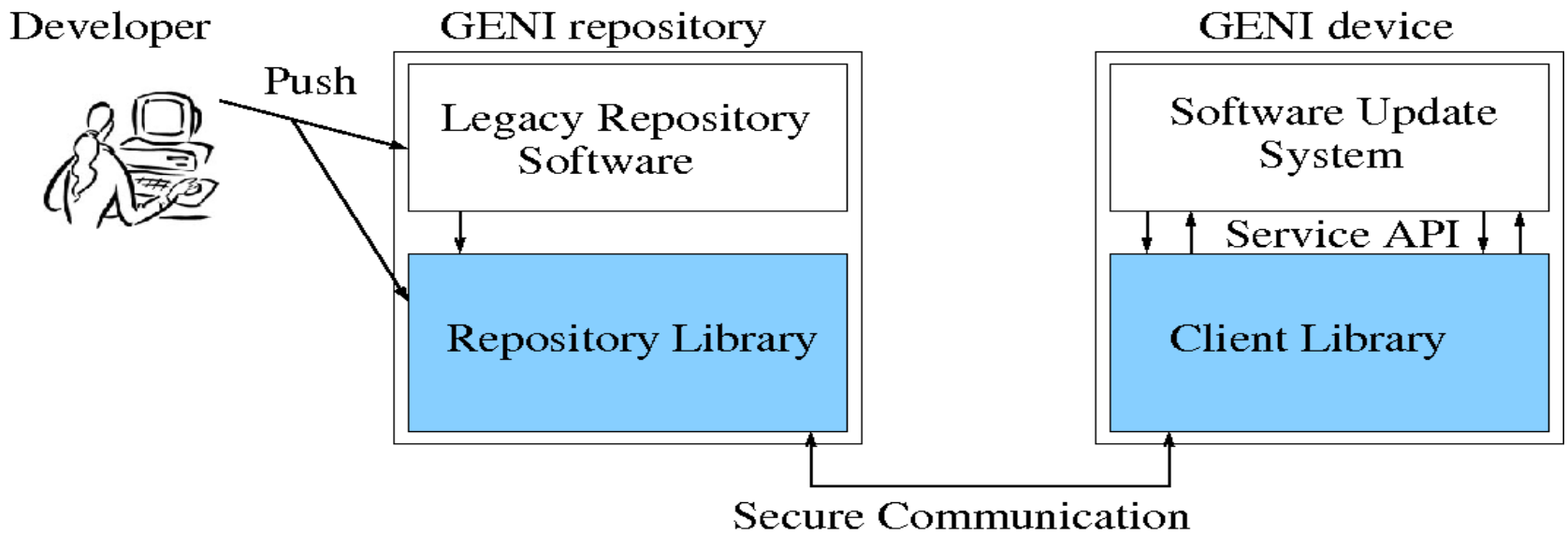
After the PlanetLab node does an update, remotely exploit it

A knowledgable attacker can root any system on PlanetLab today!

Our approach for new systems

Build a client library that provides security for software update systems

Build a repository library that correctly signs developer updates

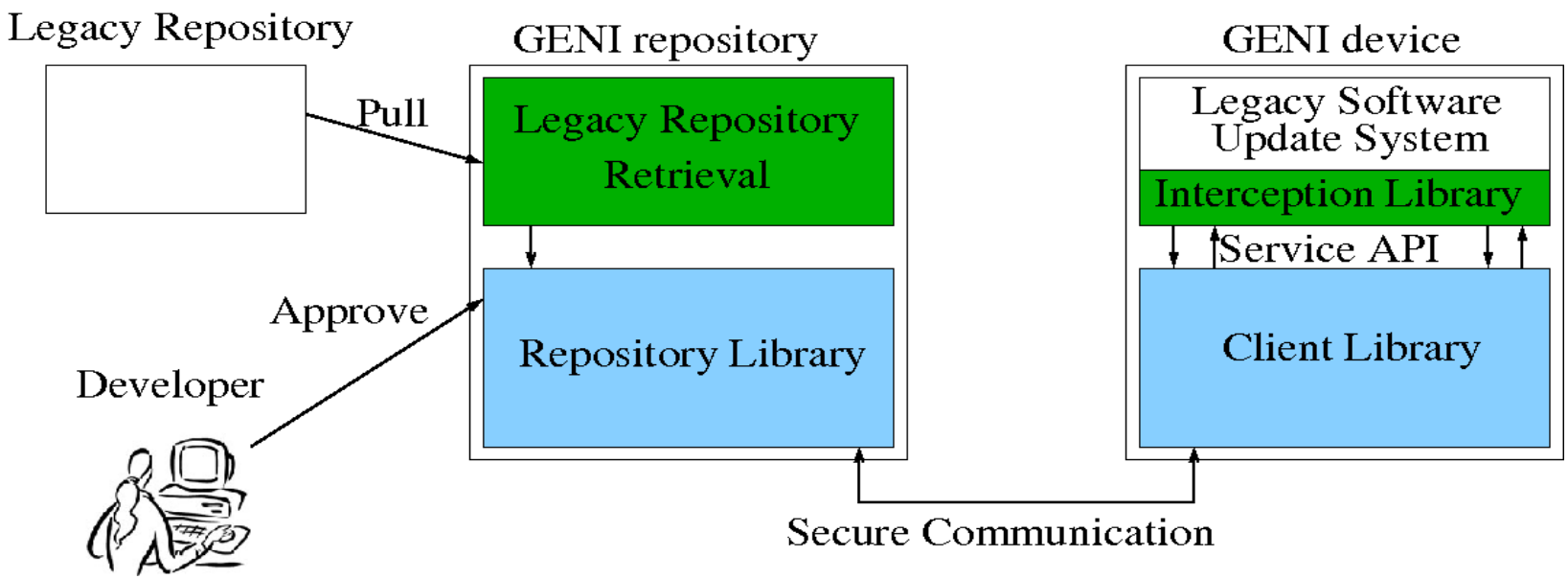


Our approach for legacy systems

Must retain functionality of existing system

Intercept traffic from insecure software update systems to transparently force it through the client library

Provide feedback to the user / system administrator



Proposal Overview

Work with the Tor project

- Many pairs of eyes uncover bugs more easily

Build an artifact early, add security mechanisms gradually

Portability of the client library is key

Focus on supporting the developer / repository interface(s) used by GENI and Tor

Conclusion

Software update systems are extremely vulnerable

Subtle issues in building a secure software update system

We propose to:

- Build a library for securing software update systems
- Secure legacy systems by exploiting their insecurity
- Work with the open source community to ensure quality

Why focus on this threat?

Existing implementations are insecure [Bellissimo 06, Cappos 08]

Software update systems run as root

Traditional defenses don't protect against attacks

Ubiquitous

An attack often appears benign

Attack code can be easily reused [EvilGrade]

Trends show server attacks are on the rise [CERT]