Chainiac: End-to-End Software Supply Chain Security and Transparency

Prof. Bryan Ford
Decentralized/Distributed Systems (DEDIS)

HRPC – November 17, 2017
No Security Without Updates…

Much of today’s malware exploits known vulnerabilities in unpatched systems

• Example: WannaCry used old Windows exploit to exploit over 230K computers in 1 day
But updates can go wrong too

Compromised software updates on the rise

Hackers Distribute Malware-Infected Media Player to Hundreds of Mac Users

Yet another software supply-chain attack hits popular applications.

Devs unknowingly use “malicious” modules snuck into official Python repository

Code packages available in PyPI contained modified installation scripts.

DAN GOODIN - 9/17/2017, 12:30 AM
Key Software Supply Challenges

- Dev signing keys may be stolen...or coerced
- Binaries may be compromised after compilation
- Update distribution repositories may be hacked

What about reproducible builds as in Debian?
- Most users don’t have the time, or the source

What about Certificate Transparency for binaries?
- CT can detect only after victim pwned, ie never
Chainiac: Secure, Transparent Software Development & Updates

DEDIS work appearing in [USENIX Security ‘17]

- Development: peer review, signoff workflow
- Build: independent verification of exact binaries
- Distribution: offline-verifiable software updates with proactive transparency via SkipChains
No Central Points of Compromise

Throughout development, build, update pipeline

• Multiple developers review, signoff releases
• Multiple auditors verify, log all workflow steps
  – Even compromised devs can’t make secret release
• Multiple build servers reproduce all binaries
  – Prevent compromise during or after compilation
• Multiple witnesses collectively sign each update
  – Proactively ensure all installable releases are public
WhoActuallyReproducesBuilds?

Most users don’t have the time, knowledge

- Reproducing a browser can take many hours on a typical laptop, forget mobile devices…

Users can’t reproduce if source is proprietary

Chainiac delegates responsibility to multiple independent software update witness-cosigners

- Reproduce exact build, co-sign its validity

Can work for proprietary software too

- Only build verifiers need NDAs/source licenses
Offline-verifiable Transparency

Chainiac logs updates on cryptographically verifiable blockchain or **SkipChain**

- Devices can verify updates offline, peer-to-peer
- But won’t accept anything inadequately co-signed

Closes CT vulnerabilities, avoids need for gossip
A Real Scenario: Apple vs FBI in 2016

FBI: “Sign an iOS with a backdoor.” Apple: “No.”

Invited public debate this time, but what about next time?

Will we know if a software vendor is secretly coerced to sign backdoored image?

- If device uses CT, then FBI simply gives device SCTs in 2 fake logs
- A phone sealed in a forensics lab can’t gossip → we’ll never know!

Only collective signing ensures transparency even if device is isolated or upstream Internet connectivity is persistantly compromised/MITM’d

- Further discussion: see “Apple, FBI, and Software Transparency” in Princeton “Freedom to Tinker” blog
Conclusion: Key Points

We urgently need transparency on what software gets onto our devices, and how it came to be

• Ensure no central points of compromise E2E
• Make reproducible builds work for ordinary users and for closed-source software
• Prevent transparency failures before installation

Chainiac demonstrates how this is achievable. More info: see “Chainiac” in [USENIX Security ‘17]

• Or summaries by Porup/CyberScoop, Colyer