Trusted Component Deployment

Trusted Components
Bernd Schoeller
January 30th, 2006
Code from the Web

➢ Can I trust the software on my machine, knowing that there are “very bad people” out there?

• risk of malicious code
• risk of insecure code
• risk of misbehaving code
Overview

➢ Cryptographic crash course
➢ Component deployment technologies
  • signed binaries
  • public key infrastructure
  • trusted computing
  • processes and virtual machines
➢ Next week: proof-carrying code
Cryptographic Crash Course

➢ Encryption / Decryption

Hello World ➔ xrndctba ➔ asjfijas43 ➔ Hello World
Cryptographic Crash Course

➢ Symmetric Encryption

Hello World ➞ xrundctba asjfijas43 ➞ Hello World

= “shared secret”
Asymmetric Encryption

Hello World = "private key"

xrundctba asjfijas43 = "public key"

Hello World = "key pair"
Hash Function

Hello World ➔ 4711 ➔ “hash function”

Hallo World ➔ 0815 ➔ “hash value”
Digital Signatures (very short)

Hello World
4711
4654

4654

equal?

Hello World
4654

4711
Using Hash Values for Trust
Using Hash Values for Trust
Using Hash Values for Trust
“Man in the Middle” Attack

Request hash value for X

The hash value for X is H

Bad guy™
Signed Binaries

- A signed binary can be authenticated to come from a certain source
  - trust developer, but not deployment channel
  - safe for “man in the middle” attack
  - automatic check
  - liability and damages
The Next Problem

➢ How do we get the **public keys** to the users?
PKI

➢ “Public Key Infrastructure”: an global or local arrangement that provides the vetting of, and the vouching for user identities and the connected distribution of public keys.

➢ Types of PKI
  • Hierarchical
  • Distributed
Hierarchical PKIs

Central Authority

- Key Issuer
  - signs the public keys of
    - people

- Key Issuer
  - signs the public keys of
    - people

- Key Issuer
  - signs the public keys of
    - people

CAs: VeriSign, BSI
Distributed PKIs

➢ “Chain of Trust”

• all participants of a security network start to sign each others keys

• a key is officially seen as valid if
  • signed by myself
  • signed by “enough” other key holders, which are trusted by myself
PGP “Web of Trust”

- Levels of trust
  - untrusted: signatures with this key are ignored
  - marginal: 2 signatures make a key valid
  - complete/unlimited: signatures with this key make a key valid
APT Secure

➢ APT is the standard package distribution tool of the Debian linux distribution
➢ Packages can be distributed world-wide
➢ User are encouraged to use a local package server with good connectivity
➢ Authenticity of packages automatically validated with digital signatures
APT Secure (cont.)

Debian
Developer

Maintainer

User
Trusted Computing

➢ Definition I:
System that lets the user verify/control the software that he is running on his machine

➢ Definition II:
System that lets the manufacturer verify/control the software that the user is running on his machine

Not part of this lecture
Trusted Computing (cont.)

➢ Integrate a chip into computers that can
  • supervise the boot process
  • safely store (write only) keys
  • verify signatures of code before execution

➢ Integrated into many IBM Thinkpad notebooks

➢ Driver available for Linux (tpm)
  • Part of Linux Kernel 2.6.12-rc2 or later
Untrusted Execution

➢ Can we execute code that we do not trust?

• separate the code execution from the rest of the system
• control access to resources (CPU, memory, IO)
• support the user in assessing computational results
Assessing Computational Results

Am I trustworthy?
UNIX Processes

➢ Traditional system separation

  • code execution is associated with a user account
  • control hardware resources (ulimit)
    • IO access controlled by “user mode”
    • CPU resource controlled by scheduler
  • memory access controlled by MMU
Virtual Machines

➢ Hardware support not needed
  • Code JIT compiled or interpreted
➢ Byte-code allows static analysis
➢ Code verification at load time
  • verified code can disable monitoring
➢ JIT compiler can integrate run-time checks
➢ Fine-grained verification
Examples of Virtual Machines

➢ JVM, .NET
➢ JavaScript
➢ Postscript
➢ Online Games: Quake, UT, Doom, Half-Life
Summary

➢ We have to be careful about the code that is executed on our machines

● There are many reasons not to trust code that others give you

● Dynamic systems require complex deployment and verification mechanisms