Solaris Security Design Considerations

Casper Dik
Sun Microsystems, Inc.

Solaris Security Design Principles

Or how ten years changed my perspective on security

- History of fixes and hardening
- Solaris 10
- Look at the future
- My greatest frustration

What was wrong?

- Bugs
- Configuration issues
- Software reuse

Bugs

- Retraining programmers
- Fixing bugs
- Codesweep
- Automated Scanning

Improving code quality

- Security awareness training
- Better programming interfaces
- Different programming languages

Bugs: Optimist's view

- And then you're done!

Bugs: Pessimist's view

- Programmers come and go
  - Continuous training required
- Training doesn't stick
- Much code imported from the outside
- Code evolves to evade automated scanning
- Code increases 10-50 fold
  - And so do bugs
- Where there are bugs, there are security bugs

Bugs: Pessimist's view

- Different programming languages, different security issues
- *You can write* C/FORTRAN *in any language*

Bugs: Open versus Closed source

- Ross Anderson[2002]: *Security in Open vs Closed Systems*
  - Defender *and* attacker helped equally
- So what happens when transitioning?
  - Tested in OpenSolaris
  - Not much, so far

Bugs: Realist's view

- Fixing bugs helps
- Fixing bugs is not sufficient

Configuration

- “Ease of Use” trumped Security
- Services defaulted to on
- Access defaulted to open
- Complaints when defaults changed
  - Remember /etc/hosts.equiv with “+” in SunOS 3 & 4?

Configuration

- Backward compatibility King
- “Like turning a supertanker”
  - File permissions fixed
  - New network services default to off
- Everything defaults to off
  - Except sshd

Configuration

- System must be secure with defaults
- Disabled services must be secure, too!

Changing World

- Everything is connected
- Much is wireless
- Dynamic content
- Webify Everything
  - Controlled Environment -> Internet
- Software reuse?!?

What we have

- Bugs
- Enabled Services
- Users
- System Administrators

What I want

- Security:
  - With bugs
  - Without firewalls
  - While doing useful work
  - Without virusscanners

Design for Resilience

- Tamper proof
- Tamper resistant
- Tamper evident

Security Evolution in Solaris 10

- Cryptographic Framework
- Privileges
- Loopback Credentials
- Zones
- RBAC
- SMF
- BART
- Trusted Extensions

Cryptographic Framework

- Cryptographic Algorithms
  - `encrypt(1), decrypt(1)`
- Digests
  - `digest(1)`
- Random number generator

Cryptographic Framework

- Two software instances of all algorithms
  - One Userland
  - One Kernel
- Completely Pluggable
  - Add accelerator (different implementation)
  - Add new algorithm
- 128-bit crypto standard
  - Import restrictions in some countries

Privileges

- Privileges with a pragmatic twist
- Principle of Privilege Escalation Prevention
  - *"You need as many Privileges as you can get"*
- Basic Privileges
  - Privileges required for previously unprivileged actions
  - Execve, fork, viewing other people's processes
  - Extensible
- Hard privilege limit
  - Privileges processes can never exceed

Privileges

- Privileges needed to control other process
  - Superset of privileges available in that process
- Privileges needed to write to /dev/*mem, /dev/dsk/*
  - All privileges defined in the system
- Users can be prevented from ever performing some tasks

Loopback Credentials

- Loopback server now knows who connects
  - Uids
  - Gids
  - Privileges
  - Audit attributes
  - Zone

Zones

- Virtual OS Instance
- Ease of administration
- Compartmentalize
- Separate namespaces
- Resource controlled
- Observable from the global zone

Service Management Facility (SMF)

- Single set of commands for all services
- Service dependency graph
- Restarts failed services
- Delegation of administrative authorizations

Role Based Access Control (RBAC)

- Allows assigning Authorizations and Roles to users
- Allows running privileged commands by unprivileged users or roles

BART

- Basic Auditing and Reporting Tool
- Verifies file contents and attributes
- To be integrated with online database
  - SunSolve Fingerprint database

Signed Binaries

- All Solaris 10 binaries carry a signature
  - Binaries can be verified off-line
  - Obviously not on a compromised system
- Requirement for export of "*Crypto with a hole*"
  - Crypto plugins must be signed
  - No obvious restrictions on who can get certificate
  - Strong crypto unbundled because of *import* restrictions

Signed Execution (Future)

- Allow restrictions on the executables run
- Allow restrictions on the kernel modules loaded
- *You* are in control!

Secure Boot (Future)

- Verify all binaries while they are loaded
- Hardware assist required for full feature set
  - TPM
  - But system administrator in control

Trusted Extensions (Soon)

- Labeled zones
- Trusted Networking (labeled networking)
- Trusted Window System
- Replaces *Trusted Solaris*

Unbundled Tools

- Hardening toolkits
  - But more and more obsolete
- Findrootkit (to be released)

My Greatest Frustration

- Incompetent Security Auditors
- About as advanced and scientific as
  - Bloodletting/Leeches
  - Animal Sacrifice
  - Palm reading
- Random, Unmotivated, Requirements
  - Known to break systems
  - Inflexible

Relevant Security Pages

- Sun Security Home Page
  - http://www.sun.com/security/
- Solaris Patches & Fingerprint Database
  - http://sunsolve.sun.com/
- Sun Security Coordination Team
  - http://sunsolve.sun.com/security/
- Sun BluePrints for Security
  - http://www.sun.com/security/blueprints/
- Solaris Security Toolkit
  - http://www.sun.com/security/jass/

Relevant Blogs

- Glenn Brunette
  - http://blogs.sun.com/gbrunett
- Alec Muffett
  - http://blogs.sun.com/alecm
- Casper Dik
  - http://blogs.sun.com/casper

Get The Source!

- http://cvs.opensolaris.org
  - Source repository
- http://www.opensolaris.org
  - Discussions, binaries and all the rest
- http://blogs.sun.com/
  - Engineers explaining their bit of Sun Software

Solaris Security Design Considerations

Casper Dik
Sun Microsystems, Inc.
http://blogs.sun.com/casper