Advanced Systems Security: Introduction

Trent Jaeger  
Systems and Internet Infrastructure Security (SIIS) Lab  
Computer Science and Engineering Department  
Pennsylvania State University

January 19, 2010
Cross-site Scripting (XSS)

- **Aim:** *Get a client to run an attackers’ code at higher privilege (Privilege Escalation)*

- **Attack details**
  - [http://www.sans.org/top-cyber-security-risks/#tutorial](http://www.sans.org/top-cyber-security-risks/#tutorial)

- **Resolutions**
  - Where are problems?
  - Can we eliminate them?

- **This is our paradigm: penetrate-and-patch**
  - Alternatives?
Systems Enable Interaction

- If it was solely about isolating processes, security would be easy
- However, process interaction is fundamental to operating systems
  - Interactions in XSS?
  - How does OS manage?
  - Other enforcement?

- Challenge: ensure security goals are met given all means of interaction
Secure Operating System

• *Provides security mechanisms that ensure that the system’s security goals are enforced by trusted components despite threats from attackers*

  ‣ Security goals?
  ‣ Trust model?
  ‣ Threats?

• Can we build a truly secure operating system?
Security Goals

• Lots of unsatisfying definitions
  ‣ Users can perform only authorized operations (safety)
  ‣ Processes perform only their necessary operations (least privilege)
  ‣ Operations can only permit information to be written to more secret levels (MLS)

• We’ll discuss these
  ‣ Defining practical and achievable security goals is a difficult task
Trust Model

- For operating system
  - Trust model == TCB
- What’s in a TCB?
- What are we trusting?
  - Aim: as little as possible
Threat Model

- Threats are means that an attacker can use to violate security goals
  - Where do threats come from?
  - What mechanisms enable threats?
  - What do threats threaten?

- Secure OS must protect TCB against threats
  - Is this sufficient?
Security Model Against XSS

• Components
  ‣ Web server, Web client, Media player
  ‣ Client OS, Root processes, Attacker

• Trust Model
  ‣ Which are trusted?

• Threat Model
  ‣ What are threats?

• Lots of (Possible) Enforcers
  ‣ Prevent unauthorized remote access, control user processes, prevent root escalation
Effective Security Models

- Can we develop an effective approach?
- What security principles can we leverage?
• Historically, OS treats applications as black boxes
  ‣ OS controls flows among applications
  ‣ Security requirements determined by allowed flows
Application policy enforcement: databases, JVM, X Windows, daemons, browsers, email clients, servers
Multi-Layered Enforcement

- Application (Appl)
  - Policy
- Application (Appl)
- Application (Appl)
- Operating System
  - Policy
- Virtual Machine Monitors
  - Policy
- Network
Network Layer

- Network Access Control == Firewall
  - Protect a network from external malice
  - This is a beneficial view of the world
  - But, is the internal network (hosts) ready for the approved messages?
Virtual Machine Layer

- Key technology: Isolation
  - We can isolate the bad stuff from one another!

- How do we get a productive computation from isolated versions of bad stuff?
  - E.g., How can we use VM isolation to improve security for web applications?
Application Layer

• Do not trust applications
  ‣ Why not?

• But, we need to depend on some application enforcement
  ‣ Many root processes
  ‣ Have more semantics
  ‣ May be able to break system

• Cannot treat apps as black boxes anymore
Security Enforcement

- Several applications include access control
  - Databases, window servers, web servers, browsers, …

- Some programming systems include access control to system resources
  - Java, Safe-Tcl, Ruby, Python, Perl – Jif, Flow Caml (information flow);

- Some systems recognize that programs may contribute to access control
  - User-level policy server for SELinux
  - Integrity: CW-Lite, UMIP, PPI
  - Distributed Information Flow Control (Flume, HiStar)

- Requirement: Ensure that all layers are using their authority in a manner consistent with system security goals
Whose Responsible?

- What should we expect from the parties involved in the computation?
  - Programmers (may be multiple groups)
  - OS Distributors
  - Administrators
  - Users
  - Service Providers
  - Content Providers
Programmers

• What are their goals?
  ▸ Function
  ▸ Ease of distribution
  ▸ ??

• Security Decisions
  ▸ Trust Model: themselves and OS (admin)
  ▸ Threat Model: bugs (inject), viruses (illegal modification),…
  ▸ Security Model+: test apps

• Problems: testing is incomplete, deployment unknown
OS Distributors

• What are their goals?
  ‣ Usable configuration
  ‣ Lots of people can use
  ‣ ??

• Security Decisions
  ‣ Trust Model: OS and system TCB
  ‣ Threat Model: applications and network subjects
  ‣ Security Model+: test OS and confine apps

• Problems: broken TCB programs, non-std configs
Administrators

- What are their goals?
  - Function
  - Ease of management
  - ??

- Security Decisions
  - Trust Model: VMM and OS (some)
  - Threat Model: network and untrusted code and users
  - Security Model+: firewall

- Problems: testing is incomplete, deployment unknown
Users

• What are their goals?
  ‣ Function
  ‣ Function
  ‣ ??

• Security Decisions
  ‣ Trust Model: entire system
  ‣ Threat Model: what threats?
  ‣ Security Model+: ignore some dialog boxes, spam, …

• Problems: too many security decisions at runtime
Put It Together

- How do we define what is necessary (for success)?
- How do we define enforcement for individual layers comprehensively?
- How do we compose the enforcement of all the layers into a coherent security architecture?
- How do we prove success?
- How do we do it without much (any) user intervention?
- Is this enough?
Take Away

• Identify core security approach
  ‣ Goals, trust model, threat model

• Map these to the system components
  ‣ What software must be enlisted in the trust model

• Enable participants to customize, refine security in practical ways
  ‣ Requires understanding of security model and functionality