

# Course 1: Overview of Secure Programming, Section 7

Pascal Meunier, Ph.D., M.Sc., CISSP

May 2004; updated August 12, 2004

Developed thanks to support and contributions from Symantec Corporation, support from the NSF SFS Capacity Building Program (Award Number 0113725) and the Purdue e-Enterprise Center

Copyright (2004) Purdue Research Foundation. All rights reserved.



# Course 1 Learning Plan

- Security overview and patching
- Public vulnerability databases and resources
- Secure software engineering
- Security assessment and testing
- Shell and environment
- Resource management
- **Trust management**

## Learning objectives

- Be able to identify trust relationships and make them explicit
- Understand how to analyze and validate trust
- Understand how trust relationships can be exploited
- Understand the role of authentication, access controls and policies in trust management
- Understand attacks on trust

# Trust Management: Outline

- Trust basis
- Trust analysis and validation
  - Data trustworthiness and flow analysis
  - Formal trust and policies
  - Trust modeling
  - Architecture: client-server paradigm
  - Trusted computing
- Trust verification and limitation
  - Authentication
  - Access control
  - Process privileges

## Basis for Trust

- Who and what do you trust?
- Does trust imply that you're taking a risk?
  - What is at stake?
- Are there assurances that something can be trusted?
  - Why do you trust?
- What tools and techniques can be used to produce or evaluate those assurances?
  - Can you justify (verify/validate) trust?
- What is trusted computing?
  - Who doesn't trust who? (goals, perspectives)

## Why do you trust?

- Reduce costs
  - Economies of scale
  - Fewer tests
  - Simpler procedures
- Benefit from the skills of another
- Delegate and gain
  - Power
  - Focus
- Hopefully, not because you have no choice
  - that's not trust, it's being dependent and at the mercy of ...
- Need to balance risks and benefits
  - What are the threats?

# Data Trustworthiness

- Track the flow of data
  - Are appropriate validation checks made on all data paths?
    - ❖ Esp. flows that can contain data from anonymous sources
- Run-time checks
  - Perl taint mode
- Static analysis
  - Compiler-like analysis
- Data dependence
  - Data channel properties
  - Dependence algebra

# Gross Trust Management Failure

- Trust web browser client to keep data safe
- HTML forms:  
hidden input  
<input type=hidden ...>
- Cookies
- Because the server put the data there, programmer thinks the data is valid
- Threat: users can easily view and change those values

## Real-Life Example: dansie.net shopping cart

- ```
<FORM METHOD=POST ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">
Black Leather purse with leather straps<BR>Price: $20.00<BR>
<INPUT TYPE=HIDDEN NAME=name      VALUE="Black leather
purse">
<INPUT TYPE=HIDDEN NAME=price     VALUE="20.00">
<INPUT TYPE=HIDDEN NAME=sh        VALUE="1">
<INPUT TYPE=HIDDEN NAME=img       VALUE="purse.jpg">
<INPUT TYPE=HIDDEN NAME=return
VALUE="http://www.dansie.net/demo.html">
<INPUT TYPE=HIDDEN NAME=custom1  VALUE="Black leather purse
with leather straps">

<INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">
</FORM>
```
- <http://online.securityfocus.com/archive/1/55172>
- CVE-2000-0253

## Hidden fields in shopping carts

- [http://www.iss.net/security\\_center/static/4621.php](http://www.iss.net/security_center/static/4621.php)  
CAN-2000-0101: Make-a-Store  
CAN-2000-0102: SalesCart  
CAN-2000-0103: SmartCart  
CAN-2000-0104: Shoptron  
CAN-2000-0106: EasyCart  
CAN-2000-0108: Intellivend  
CAN-2000-0110: WebSiteTool  
CAN-2000-0123: Filemaker example  
CAN-2000-0134: Check It Out  
CAN-2000-0135: @Retail  
CAN-2000-0136: Cart32  
CAN-2000-0137: CartIt  
CVE-2000-0253: dansie

## Mini-Lab: Perl's taint mode

- Start a script with:
  - `#!/usr/bin/perl -Tw`
    - ❖ `-T` switch turns on taint mode
- To untaint data, it must go through a regular expression
- Regular expression may do nothing (defeats the purpose)
  - `$tainted =~ /(.*)/;`
    - ❖ `/ /` specifies a regular expression
    - ❖ `()` specifies a match
  - The `$1` variable contains the first match
    - ❖ Example: `$clean = $1;`

## Perl Taint Mode Mini-Lab

- Here is a perl script that will safely run the "wc" (word count) command on any file whose 8.3 format name is passed to it as an argument:

- `#!/usr/bin/perl -wT`
- `$ENV{PATH} = ""; # Try commenting out this line`
- `foreach $file (@ARGV) {`
- `print "Processing $file:\n";`
- `if ($file =~ m/(\^\w{1,8}\.\w{0,3})$) {`
- `system("/usr/bin/wc $1");`
- `} else {`
- `print "non-conformant file name: %file\n";`
- `}`
- `}`

## Perl Taint Mode Mini-Lab

- Run the program and modify it to answer the following questions
  - `perl -wT taint.pl filename.txt2 file2.txt`
- What happens
  1. If you don't purge the PATH environment variable?
  2. If you don't use a regular expression?
  3. If you use `system("/usr/bin/wc", $file)`?
  4. If the regular expression is changed to `m/(.*)/?`

## Formal Trust and Policies

- How to justify (validate/verify) trust
- Company security policies should describe standards, specifications and procedures for securing assets
  - A formally trusted asset complies with all
    - ❖ Asset is used with minimal checking
  - Partial compliance means partial trust
    - ❖ Requires additional safeguards in any application or system making use of the asset
  - Trust is not binary (Yes/No)
    - ❖ Calculated risk

## Question

- An external contractor lets your company access a database but will not let you audit their procedures (auditing wasn't part of the contract). How does that affect the trust put into that database for your applications that use it?
  - § a) Although we "trust" them to do a best effort at security, we design applications by assuming that their database could have been compromised
  - § b) We trust their data, and if their database should get compromised we will sue them for damages
  - § c) Since the contract specifies which procedures they were supposed to follow, we will mostly trust the database but include several basic "sanity checks"

## Trick Question Answer

- The correct answer depends on your company's security policies, security stance, and perhaps just the policies for that project (and its criticality).
- § a) Although we "trust" them to do a best effort at security, we design applications by assuming that their database could have been compromised
- § b) We trust their data, and if their database should get compromised we will sue them for damages
- § c) Since the contract specifies which procedures they were supposed to follow, we will mostly trust the database but include several basic "sanity checks"

# Trust Types

- Direct
  - You perform the validation yourself
  - No delegation
- Transitive
  - Entities with the **same** standards and criteria trust the evaluation results of the others
    - ❖ A trusts B
    - ❖ B trusts C
    - ❖ A trusts C transitively through B
- Assumptive (a.k.a. "spontaneous")
  - Trusting the results of an evaluation conducted with different standards, criteria or procedures

## Example Assumptive Trust

- A friend tells you that Brand Z's mint ice cream is great, so you buy some.
  - Your tastes are somewhat *different* but you trust the evaluation
- Can you provide another example?

## Example Assumptive Trust #2

- PGP chain of trust
  - Signing another's key
  - What are the criteria and procedures?
  - You have to accept the signature or not, even if you require 8 pieces of ID, and someone else may sign just on someone's word
    - ❖ PGP key signing is not quite transitive trust

# Foundations of Trust

- Identification
- Authentication
- Accountability
- Authorization
- Availability

# Trust Modeling

- Who and what do you trust?
  - Trust binds a property or attribute (expected behavior) to an identifiable asset
  - Need authentication to assert identity
  - Need access control to provide just enough authority for desired capabilities

# Trust Model

- A trust model is expressed in relation to a threat model
  - How can you trust that the threats won't materialize?
    - ❖ Specify defense mechanisms
      - e.g., security functional requirements
    - ❖ Change design if necessary
    - ❖ Validate implementation
- Cost/benefit analysis
  - Threats vs
    - ❖ Defense mechanisms
    - ❖ More expensive design options
    - ❖ More expensive assurance requirements and processes

# Threat Modeling

- Identify and enumerate threats
  - Data flow analysis
    - ❖ design
    - ❖ implementation correctness
  - Who could do what?
    - ❖ Insider threats are usually significant
      - Insiders are not necessarily your co-workers
        - » e.g., anyone who can access your internal LAN...
- Analyze possible costs of exploits
- Danger: shape the threat model with what we know how to solve cheaply, instead of what the threats really are
  - Discussion: <http://iang.org/ssl/wytm.html>

# Client-Server Paradigm

- Offload tasks to the clients so that server load "scales" well
- Which tasks can be entrusted to the clients?
- Can you trust data from the clients?
  - Can you trust data that you stored on a client?
    - ❖ How did you store it?
      - Was it encrypted?
      - Was the encryption strong enough?
      - Was it tamper-proof?
    - ❖ Could it be replaced with data you gave another client?
  - What about data computed on a client?
    - ❖ Distributed computations?
    - ❖ Multiplayer games?

## As a client, can you trust

- JavaScript to run on your system?
- ActiveX?
- Java?
- Can you trust servers to be fair and uncompromised?
- Emails? When is an email considered code?
- AIM messages?
- How much storage can the server use on your machine (cookies, etc...)?
- Am I running spyware, DRM or otherwise potentially user-hostile code?

## The ActiveX Trust Model

- ActiveX modules are very powerful but foreign pieces of code that run on your system
- There are risks associated with the capabilities of the ActiveX model
- Every ActiveX module poses threats
- A cryptographic signature ties it to its source, so the accountability (through identification and authentication) provides assurance.
  
- Do you trust the module's source? Quick, you are in the middle of something important...

## Problems with Authenticode

- Users are usually not competent to decide on the security of an ActiveX module
  - Users have been conditioned to click through the annoying meaningless stuff
  - XP SP2: No user interface, Active X controls are blocked
- Unsigned ActiveX “controls” could also get executed (e.g., Chaos Computer Club demo)
- You can make an ActiveX “control” that allows all other ActiveX controls to be executed silently
- Anyone can obtain a signing key
- Accountability (liability) unclear and may vary. What recourses do you have (typically none)?

# JavaScript "Security"

- Cross-domain security model: pages from one domain can't access (and modify) pages from another domain
  - Often not implemented correctly (Opera 6 & 7, IE, Mozilla 1.0, 1.1, 1.3, 1.4, Netscape 6 & 7)
  - Cross-domain security violated by other technologies and various loopholes
- Signed scripts (Mozilla)
- Hodgepodge of "hobbles" that still
  - Allowed a javascript to modify ActiveX security settings to accept all ActiveX controls without prompting
  - Allowed malicious plugins to be installed

# Java Security Model

- Code runs inside a virtual machine
- Code properties are audited before execution
  - “Untrusted” applets have limited capabilities
    - ❖ e.g., network connections only to originating host
  - Network applets are untrusted by default
  - “Trusted” applets may do a lot
    - ❖ Limited by file access control list
      - Read and write permissions for given files
  - Signed applets can become “trusted” (here we go again :-)

# Trusted Computing

- Who trusts what?
  - Sounds like users can trust their computers (finally!) but...
- Enables servers to trust clients
- DRM (Digital Rights Management):
  - Allow content providers to trust your computer to manage IP appropriately
  - User loses control, can't trust computer anymore
  - Client is trusted by server
  - Client doesn't trust user
  - Hardware precedent: DVD players
  - Computers need to protect the software

## Question

- Can you trust code running in client web browsers to perform authentication (e.g., JavaScript, Active X)?
  - § a) Yes, why do you ask?
  - § b) Yes, my ActiveX control is signed so users can't modify it
  - § c) No in most cases

## Question

- Can you trust code running in client web browsers to perform authentication (e.g., JavaScript, Active X)?
  - § a) Yes, why do you ask?
    - § How do you know your code is even running?
  - § b) Yes, my ActiveX control is signed so users can't modify it
    - § How do you know they are running your ActiveX control?
  - § **c) No in most cases**
    - § Unless you have assurances that an unmodified browser is running your unmodified code and ... (trusted computing has happened)

## How To Build a Threat Model

- Identify all the assets
  - Which aspect of Confidentiality, Integrity and Availability (CIA) is valuable about them?
- Identify the threats and attackers
  - Not just in-the-wild threats
- Identify solutions, architectures or technologies that allow you to bind a trust property to the asset, even in presence of the threat
- Identify threats to the solutions (sub-threats) and repeat
- Who should do it?
  - Architects and Managers

## Example: Browser Threats

- You want to enable users to trust urls displayed by the browser
- However, urls can be obfuscated (threat)
  - See <http://n3dst4.com/network/obfus>
- What could you do to help users, if you were designing a web browser?
- Additional references
  - "Threat Modeling", Swiderski and Snyder (Microsoft Press)
  - NIST special publication 800-30 on risk management:
    - ❖ <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>

## Exercise: Build a Threat Model

- Point your browsers to the "browser threat model":
  - [http://iang.org/ssl/browser\\_threat\\_model.html](http://iang.org/ssl/browser_threat_model.html)
- Form teams of 2 or 3
  - Each team will take ownership of one (each different) threat and perform trust modeling on it, assuming the scenario where a Symantec user needs to access resources on the Symantec web site
- After 20 minutes, each team will report on its trust modeling effort, for no more than 5 minutes
  - What security requirements (functional, assurance) applied on what would make you trust the site in general?
  - What about the Symantec store?

# Trust Verification and Limitation

- Authentication
- Access controls
- Process privileges

# Authentication

- Trust is specific to a user, asset, protocol, applet or group thereof, so identity must be verified
- Principle of Separation of Privileges
  - Two- or three-factor authentication
  - Problem: proving link between token and person
- Cryptographic hashes
  - Identity of code
  - Verify version numbers
- Trusted Computing
  - How can you trust remote identity?
    - ❖ Use trusted hardware
      - Can't hardware be hacked?
        - » Yes, but it can be made difficult

## Goal of Access Controls

- Limit a user or asset to capabilities that match the trust model and policies
- Provide guarantees about the transfer of capabilities and the properties of assets
- Tradeoff of flexibility vs complexity and administration costs
- Multiple simultaneous mechanisms are possible

# Access Control Types

- All or none (access to system is complete or nil)
- Discretionary a.k.a. Identity-based
  - Object owner can decide access by others
- Mandatory, a.k.a. rule-based
  - Owner can't change or control access by others
- Originator-controlled
  - Current owner can't change access, but creator can
- Role-based
  - Role is a template applied to a user's privileges
- Policy engines
  - Real-time interpretation of policies

# Access Control Management

- Access control lists
- Access control matrix
- Central server and distributed services
  - Kerberos
  - Session ID valid over several related web sites, or services

## Windows Tokens Contain:

- SID
  - Used for access control lists
    - ❖ Read, write, etc... affecting specific objects
    - ❖ e.g., run as administrator to access the registry remotely
- Privileges
  - System-wide
    - ❖ e.g., backup and restore
  - Don't need to run as administrator or other

## Examples of Windows Process Privileges

- Backup and restore privileges
  - SeBackupPrivilege
  - SeRestorePrivilege
  - Note that they override normal file access!
- Act as part of the operating system (SeTcbPrivilege)
- Debug programs (SeDebugPrivilege)
- Replace a process level token
- Load and unload device drivers
- Take ownership of files and other objects

# UNIX Process Permissions

- Permissions are determined based on ID
- UserIDs in UNIX
  - Real
  - Effective
  - Saved
- Group IDs
  - Real
  - Effective
  - Saved
- Various functions to change them
- Setuid and setgid programs set the effective IDs to the owner or group of the program

## Changing IDs in UNIX

- setuid, setreuid, etc...
- The effective userid and groupid are used for ACLs
- Root (super-user) has special capabilities
- Normally:
  - The real user ID can be set to
    - ❖ the effective user ID
  - The effective user ID can either be set to
    - ❖ the saved user ID
    - ❖ the real user ID
- If the effective userid is root:
  - The real user ID and the effective user ID can be set to any legal value

## Windows Exercise (or, What Not To Do)

- Suppose service S1 is started with the default “Log On As” of “Local System”. S1 accepts HTTP connections and authenticates normal Windows users. Once authenticated, it then impersonates the user (which means it gets a different, lower privilege token, not a System token) to mediate access to documents on the filesystem. If the service encounters an executable, it runs the executable by calling `CreateProcess()`.

What is the problem?

- Hint: The executable will be run with which token?

# UNIX Exercise

- Suppose that UNIX user `Alice` starts program `P1` which execs program `P2`. `P1` does not have `setuid` or `setgid` bits set. User `Bob` owns `P2` which has the `setuid` bit set.
  - What are the real user id (RUID), effective user id (EUID), and saved user id (SUID) of `P1` and `P2`? Make a table.
  - What changes can program `P2` make to the RUID and EUID?
  - If `root` owns `P2`, what changes can be made to the RUID and EUID?
  - Explain why the designers of program `P2` might use system calls to change the user id.
  - What happens if `P2` creates a file, closes it, then attempts to open the file?
- Do "`man setuid`" for hints

# Questions or Comments?

§

## About These Slides

- You are free to copy, distribute, display, and perform the work; and to make derivative works, under the following conditions.
  - You must give the original author and other contributors credit
  - The work will be used for personal or non-commercial educational uses only, and not for commercial activities and purposes
  - For any reuse or distribution, you must make clear to others the terms of use for this work
  - Derivative works must retain and be subject to the same conditions, and contain a note identifying the new contributor(s) and date of modification
  - For other uses please contact the Purdue Office of Technology Commercialization.
- Developed thanks to the support of Symantec Corporation

# **Pascal Meunier** **pmeunier@purdue.edu**

Contributors:

Jared Robinson, Alan Krassowski, Craig Ozancin, Tim Brown, Wes Higaki, Melissa Dark, Chris Clifton, Gustavo Rodriguez-Rivera

