Design Flaw #1: Earn or give but never assume trust

Laurie Williams
williams@csc.ncsu.edu

Computer Science
NC STATE UNIVERSITY

Trust

- Trust, whether it is in external systems, code, people, etc., should always be closely held and never loosely given … and never by default
- Design a software system under the assumption that components running on any platform whose integrity can't be attested are inherently not trustable, and are therefore unsuitable for performing security sensitive tasks.
- Even if users are known, they are susceptible to social engineering attacks and insider threat.

http://www.d00med.net/news.html/_/news/security/social-engineering-the-increasing-treath-r47

NC STATE UNIVERSITY

Some things to consider

- Don't assume that server APIs will always be called in the same order every time.
- Don't assume that the user interface is always able to restrict what the user is able to send to the server (need server-side checks!)
- Don't build build the business logic solely on the client side, or attempt to actually store a secret in the client.
  - Secrets will be found!
  - Intellectual process will be lost!
- Make sure all data received from an untrusted client are properly validated before processing

Computer Science
NC STATE UNIVERSITY

Don't trust "them" to use the gate

http://failblog.cheezburger.com/tag/security/page/5?ref=pagination

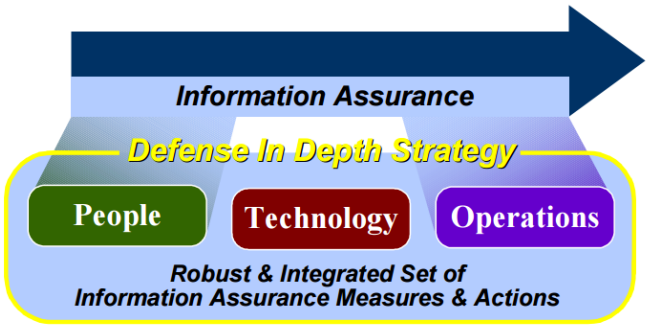
Computer Science
NC STATE UNIVERSITY

Defense in Depth

- A system should have multiple, redundant defensive countermeasures to discourage potential attackers.
- Each component in a path that leads to a critical component should implement its own proper security measures in its own context (as if it was the "last man standing").
- May conflict with simplicity philosophy
- Consistent with secure weakest link principle

http://kellebass.deviantart.com/art/Last-man-standing-146987052

Defense in Depth



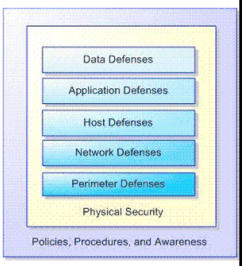
People	Technology	Operations
Policies & Procedures Training & Awareness Physical Security System Security Admin	Network & Infrastructure Computing Environment Segmentation Encryption	Security Management Key Management Readiness Assessments Recovery & Reconstitution

https://www.nsa.gov/ia/_files/support/defenseindepth.pdf

Defense in Depth: Examples

- Security camera + guard + bulletproof glass + combination lock for bank
- Firewall + encryption + Hibernate + white list server side + Javascript checks for important data
- Firewall + an intrusion detection system + strong authentication for important servers

Don't want a single point of failure!



http://kellebass.deviantart.com/art/Last-man-standing-146987052
http://3forward.com/managing-sales-teams/your-sales-team-may-be-a-single-point-of-failure/

Computer Science
NC STATE UNIVERSITY

Never Assume Your Secrets are Safe (Principle of Open Design)

- Security of a mechanism should not depend upon the secrecy of its design or implementation. No "security through obscurity."
- Inevitably information about the internals of a system will be discovered by malicious users.
  - Insider threat
  - Reverse engineering
- By revealing internals of a system (such as open source), you can get evaluation from security reviewers

http://www.ecybercrime.com/2011/02/security-through-obscurity.html

Related Principle: Secure by Default

- Assumptions:
  - Users are likely not educated enough to know about security.
  - Users may not think they care about security (until there is a problem)
- A system's default settings should be as secure as possible. For example:
  - No open network ports
  - No default passwords (e.g. to the database)
- All security functionality should be enabled by default
- All optional features which entail any security risk should be disabled by default.
- Operations that reduce security should be relatively hidden (e.g., in an "advanced" preference panel) and should make the risks readily apparent.

Users

http://1000awesomethings.com/2011/02/23/302-grandma-hair/ and http://www.akerstroms.com/en/sesam/abb-robotics__642

Related Principle: Fail Securely

- Application should NOT <u>disclose any data</u> that would not be disclosed ordinarily
- Application should NOT <u>provide any access</u> that would not be given ordinarily
- Application should NOT <u>provide too much information</u> on why the failure occurred.
- Otherwise, an attacker will make your system fail to bypass security mechanisms because your failure mode is insecure.

http://www.jurinnov.com/fail-secure-the-correct-way-to-crash

Design Flaw #2: Use an Authentication Mechanism that Cannot be Bypassed or Tampered With

Laurie Williams
williams@csc.ncsu.edu

Computer Science
NC STATE UNIVERSITY

Authentication

- Authentication is the act of validating an entity's identity.
- Authorize users at the first point of authentication
  - Encrypt stored passwords
  - Use account lockout policies
  - Support password expiration periods
  - Require strong passwords
  - Protect authentication cookies
- Prevent user from changing identity without re-authentication

Computer Science
NC STATE UNIVERSITY

http://www.digitaltrends.com/mobile/crack-this-how-to-pick-strong-passwords-and-keep-them-that-way/

Some ways to bypass, tamper with authentication mechanisms

- The user can access the service by navigating directly to an "obscure" URL … ~~security through obscurity~~
- The authentication mechanism uses assumptions about sole possession of resources (e.g IP address, MAC address) that may actually be shared or spoofed.
- System tokens (such as session IDs) should not be deterministically derived from easy-to-obtain information, such as a user name, then it becomes possible to forge identities, allowing users to impersonate other users.
- Limit the lifetime of an authentication interaction; time out after extended inactivity
  - Balance with usability

Design Flaw #3: Authorize After You Authenticate

Laurie Williams
williams@csc.ncsu.edu

Authorize

- Authorization is the act of checking that the user is allowed or disallowed to perform certain actions based upon who they are
- Authorization of a specially sensitive operation (for example, transferring a sum of money larger than a designated threshold) may require a re-authentication or a higher level of authentication.
- The user interface should allow the user to easily review any active authority relationships that would affect security-relevant decisions.
- The interface should allow accounts/privileges to be easily disabled/revoked.

Computer Science
NC STATE UNIVERSITY

Separation of Privilege

- System should not grant permission based upon a single condition.
- Sensitive operations should require the cooperation of more than one check.
  - Two signatures on a large check.
  - UNIX users cannot change from their account to the root account unless:
    - User knows the root password
    - User is in the "wheel" group (group with GID 0)

Computer Science
NC STATE UNIVERSITY

Access Control

- Access is the ability to do something with a computer resource (e.g., use, change, or view).
- Access control is the process by which use of a system's resources is regulated according to a security policy and is permitted by only authorized entities (users, programs, processes) according to that policy
- Computer- based access controls can prescribe not only who or what process may have access to a specific system resource, but also the type of access that is permitted

http://csrc.nist.gov/groups/SNS/rbac/documents/design_implementation/Intro_role_based_access.htm

Computer Science NC STATE UNIVERSITY

Security Policy

- A security policy is composed of a sequence of rules that specify under what conditions a user/actor can access specified resources. In other words, the policy describes a sequence of rules to decide whether access requests are denied/disallowed (i.e., being illegitimate) or accepted/allowed (i.e., being legitimate).
  - Policy …. Deals with subject, object, action

Computer Science NC STATE UNIVERSITY

Discretionary Access Control (DAC)

- DAC permits the granting and revoking of access control privileges to be left to the discretion of the individual users.
- A DAC mechanism allows users to grant or revoke access, ownership, and delegation of rights to any of the objects under their control.
- Advantage: flexible
- Disadvantage: no control of information dissemination; complete trust of security policy administration given to user

Computer Science NC STATE UNIVERSITY

http://csrc.nist.gov/groups/SNS/rbac/documents/design_implementation/Intro_role_based_access.htm

Mandatory Access Control (MAC)

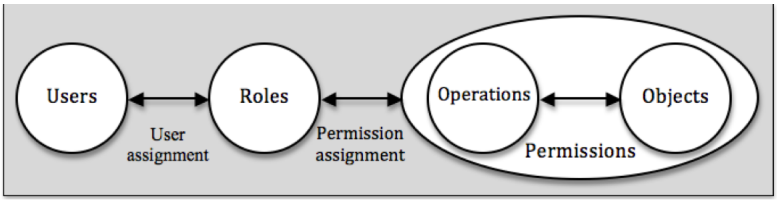
- A means of restricting access to objects based on the sensitivity (as represented by a sensitivity label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity.
- Often used for highly sensitive government and military information

Computer Science NC STATE UNIVERSITY

http://csrc.nist.gov/groups/SNS/rbac/documents/design_implementation/Intro_role_based_access.htm

Role-Based Access Control (RBAC)

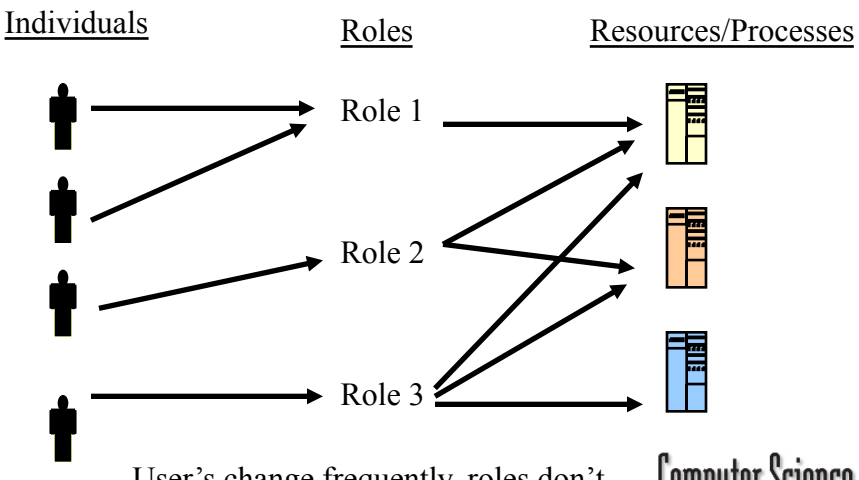
- A user has access to an object based on the assigned role.
- Roles are often defined based on job functions.
- Privileges are defined based on job authority and responsibilities within a job function. ("need to know")
- Operations on a resource are invocated based on the permissions associated with the privilege.
- The object is concerned with the user's role and not the user.



csrc.nist.gov/rbac/alvarez.ppt and http://courses.cs.ut.ee/2010/is/uploads/Main/RBAC-for-UML.pdf

NC STATE UNIVERSITY

Role-Based Access Control



User's change frequently, roles don't

From: csrc.nist.gov/rbac/alvarez.ppt

Computer Science
NC STATE UNIVERSITY

RBAC is Many-to-Many

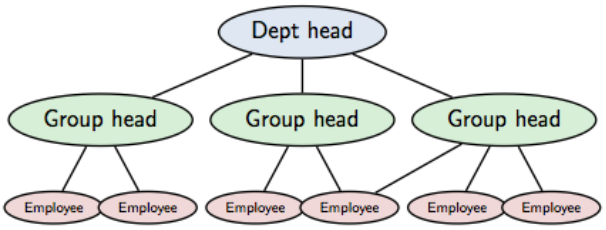
- Users may be assigned many roles (though more likely just one)
- Roles may have many users assigned to them
- Roles may be assigned many permissions
- Permissions may be assigned to many roles
- Permissions may be granted to perform many different types of operations on an object

www.sans.edu/resources/student_projects/200608_002.ppt

Computer Science NC STATE UNIVERSITY

Hierarchies are possible … and can be complicated

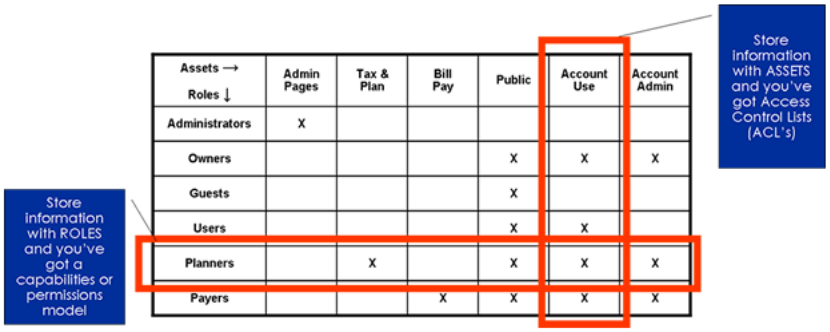
- … and can be complicated
- … and can present conflicts



http://www.icg.isy.liu.se/courses/tsit02/lectures/cseclecture03.pdf

Computer Science NC STATE UNIVERSITY

Access Control Matrix



Assets → / Roles ↓	Admin Pages	Tax & Plan	Bill Pay	Public	Account Use	Account Admin
Administrators	X					
Owners				X	X	X
Guests				X		
Users				X	X	
Planners		X		X	X	X
Payers			X	X	X	X

1. Assign users to roles
2. Provide roles permissions
3. Capture the rules

https://www.owasp.org/index.php/Access_Control_In_Your_J2EE_Application

Computer Science NC STATE UNIVERSITY

Access Control best practices

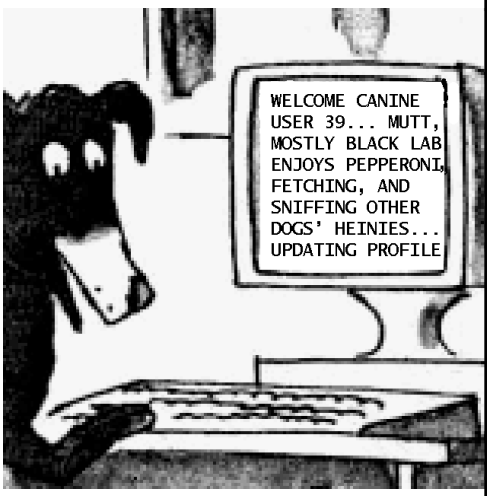
- Policies should be persisted and centralized
- Use a policy language (XACML)
- Have a centralized Access Controller

```
ACLService.isAuthorized(ACTION_CONSTANT)
ACLService.assertAuthorized(ACTION_CONSTANT)
```

- Controller manages conflicts, hierarchies, negative permissions
- Keep user identity in session
- Load entitlements server side from trusted source
- Force authorization checks on all requests
- Deny by default

https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

Computer Science NC STATE UNIVERSITY



http://resound.typepad.com/.a/6a00d8341c074953ef014e8b87928a970d-pi

Computer Science
NC STATE UNIVERSITY

Design Flaw #4: Strictly separate data and control instructions and never process control instructions from untrusted sources

Laurie Williams
williams@csc.ncsu.edu

Computer Science
NC STATE UNIVERSITY

Strictly Separate Data and Control Instructions …

- Co-mingling data and control instructions in a single entity, especially a string, can lead to injection vulnerabilities.
  - Examples of such vulnerabilities include SQL query injection, cross-site JavaScript injection, and shell command injection
- Lack of strict separation between data and code can lead to untrusted data controlling the execution flow of a software system.

Computer Science NC STATE UNIVERSITY

Command injection

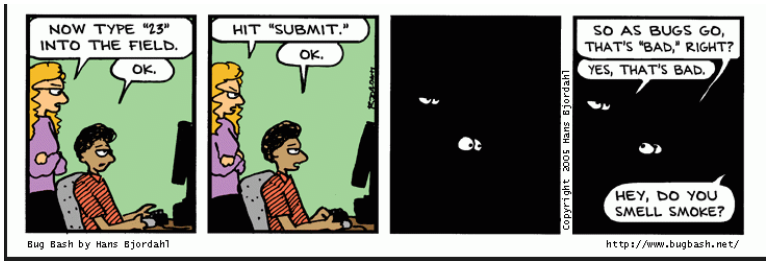
```php
<?php
echo shell_exec('cat '.$_GET['filename']);
?>
```

```
www.mysite.com/viewcontent.php?filename=my_great_content.txt
```

```
www.mysite.com/viewcontent.php?filename=my_great_content.txt;ls
```

https://www.golemtechnologies.com/articles/shell-injection

Computer Science NC STATE UNIVERSITY



www.bugbash.net

Computer Science
NC STATE UNIVERSITY

Design Flaw #5: Define an approach that ensures all data are explicitly validated

Laurie Williams
williams@csc.ncsu.edu

Computer Science
NC STATE UNIVERSITY

32

Define an approach that ensures all data are explicitly validated

- Assume all input is malicious
- Constrain, reject, and sanitize your input
- Centralize your input filtering approach
- Encrypt sensitive cookies
- Do not trust HTTP header
- Use white list

https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

Input validation guidelines - 1

- Design or use centralized validation mechanisms (e.g. frameworks and protocols)
  - Request filter or interceptor provided by web application framework to perform input validation on all request parameters
  - Communication protocols validate all fields of protocol messages before processing takes place
  - Systems using complex data formats (such as XML documents) perform parsing, syntactic validation and semantic validation in a dedicated validation module
- Transform data into a canonical form (e.g. encoded)
  - Ensures validation cannot be bypassed by supplying inputs that are encoded
- Use common libraries of validation primitives (e.g. predicates that recognize well-formed email addresses or URLs)
  - To apply consistent use of validation predicates

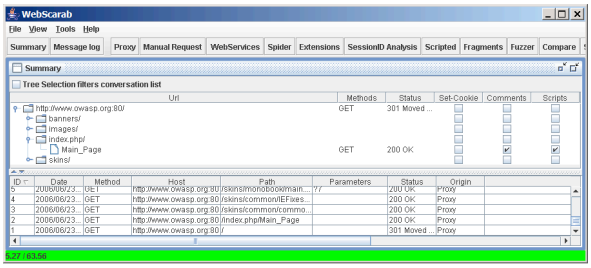
Input validation guidelines - 2

- When necessary, implement input validation that is state-aware
  - Set of valid values may depend upon state
- Explicitly re-validate assumptions "nearby" code that relies on them
  - Developers of another layer may not understand security preconditions at the "nearby" code
  - Precondition states at the entry points are recommended
  - Provide friendly and non-chatty error messages should a precondition fail
- Use implementation-language-level types to capture assumptions about data validity
  - e.g. well formed date and time in ISO 8601 format

Computer Science NC STATE UNIVERSITY

Client side checking is a very thin layer in "defense in depth"

- Turn Javascript off
- Edit HTML
- Use proxy tool (like WebScarab)



https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

Computer Science NC STATE UNIVERSITY

http://dilbert.com/strips/comic/2010-11-24/

Computer Science NC STATE UNIVERSITY

Denial of Service Attacks

- A denial of service (DoS) attack denies service to valid users (e.g. makes a web server temporarily unavailable or unusable)
  - Reasonably easy to achieve
  - Can be anonymous

Computer Science NC STATE UNIVERSITY

DoS Mitigation Techniques

- **Appropriate authentication –** process by which an entity verifies that another entity is who or what it claims to be
- **Appropriate authorization –** check to see if the authenticated entity has authority to access to the resource being requested
- **Filtering –** inspecting data as its received and making a decision to accept or reject the input
- **Throttling –** limiting the number of requests to your system, particularly with anonymous requests; intrusion-detection systems
- **Quality of Service –** provide preferential treatment for specific types of traffic

Howard and LeBlanc, *Writing Secure Code*, Microsoft Press, 2003.

Computer Science
NC STATE UNIVERSITY

Application Crashing

- Inputs that cause an unanticipated error in application
  - Buffer overflow
  - Malformed data, causing parser exception
  - SQL injection (; shutdown --)
- Mitigation: Filtering

Computer Science
NC STATE UNIVERSITY

Data Destruction

- Tampering the data
  - SQL Injection
    - DELETE * from [table]
    - Renders the application useless
- Intentional User Lock
    - Intentionally failing multiple login attempts with each possible username; application users can be locked out
- Mitigation: Filtering, Authentication, Authorization, Throttling

Computer Science
NC STATE UNIVERSITY

Resource Depletion - 1

- CPU consumption:
  - A large forums application
  - Contains millions of messages
  - Allows performing sophisticated regular expression searches
  - Create complicated regular expression which consume a lot of CPU each time search is initiated
  - Write script to launch this request over and over again
  - … or intense nested SQL query … or [infinite] loops
- Mitigation: Throttling

http://www.owasp.org/images/d/da/OWASP_IL_7_Application_DOS.pdf

Computer Science
NC STATE UNIVERSITY

Resource Depletion - 2

- Memory consumption:
  - A web mail application
  - Allows uploading files for attachment
  - Attachments stored in application's memory until the "send" button is pushed
  - No limitation on size or number of attachments
  - Attacker can upload thousands of attachments, consuming all the free memory in the machine
- Mitigation: Throttling; Timeout functionality

http://www.owasp.org/images/d/da/OWASP_IL_7_Application_DOS.pdf

Computer Science NC STATE UNIVERSITY

Resource Depletion - 3

- Disk consumption:
  - Any web application
  - Detailed logging is used for each application error
  - Attacker generates error and repeats until disk is full
  - Application behavior once disk is full is unexpected
    - Might terminate application
    - Might crash entire system
- Mitigation: Throttling; summarizing log errors (e.g. "42 errors in last 10 minutes")

http://www.owasp.org/images/d/da/OWASP_IL_7_Application_DOS.pdf

Computer Science NC STATE UNIVERSITY

Resource Depletion - 4

- Network consumption:
  - Any web application
  - Attacker identifies small request which results in large amounts of data (Display all items)
  - Attacker can launch the request over and over again, causing the database to send large amounts of data back to the web server in each request
- Mitigation: Throttling, authentication, authorization, Quality of Service

http://www.owasp.org/images/d/da/OWASP_IL_7_Application_DOS.pdf

Computer Science NC STATE UNIVERSITY

Distributed Denial of Service

- Flood a large number of messages in an attempt to slow down or stop a site because the site is overwhelmed
  -
- Mitigation: Throttling, checking IP address of sender …

http://cwe.mitre.org/data/definitions/327.html

Computer Science NC STATE UNIVERSITY