Architecture and Design Principles

Laurie Williams
williams@csc.ncsu.edu

Software Security Touchpoints

Design Flaws

- ... account for 50% of security problems.
- ... can’t find these by staring at the code or running static analysis
- ... have to have a higher-level view by examining architecture and design
- ... versus established security architecture and design principles
- ... once you understand them, you can do risk analysis, as Software Security advises
- ... started with 1975 paper “The Protection of Information in Computer Systems” by Saltzer and Schroeder

Securing the Weakest Link

- Attackers are more likely to attack a weak spot in a software system than to penetrate a heavily fortified component.
  - For example, some cryptographic algorithms can take many years to break, so attackers are not likely to attack encrypted information communicated in a network. Instead, the endpoints of communication (e.g., servers) may be much easier to attack.

Defense in Depth

- A system should have multiple defensive countermeasures to discourage potential attackers.
- Each component in a path that leads to a critical component implement proper security measures in its own context (as if it was the “last man standing”).
- Firewall + encryption + Hibernate + white list server side + Javascript checks …

Failing Securely

- Application dos not disclose any data that would not be disclosed ordinarily, data cannot be tampered with.
- If not, attacker will know he/she can make your code fail and can bypass security mechanisms because your failure mode is insecure.
- Do not provide a lot of information on why the failure occurred.
Least Privilege

- “Need to know” rule; role-based access
- A subject (user/other system) should be given only those privileges it needs to complete its task.
- The function (or role) of a user (as opposed its identity) should control the assignment of rights.
- Specific action requires a subject’s access rights be augmented, extra rights should be relinquished immediately upon completion of the action.
- Analyze communication and control paths to determine necessary privileges.
  - Process runs as admin, malicious code runs as admin

Separation of Privilege

- System should not grant permission based upon a single condition. Sensitive operations should require the cooperation of more than one check
  - The purchaser should not approve their own large purchase.
  - Similarly, access to sensitive data/operations should require more than one check.
Economy of Mechanism

- Security mechanism should be as simple as possible.
  - KISS (Keep it Simple and Small)
- Complex interactions make verifying security of systems more difficult.
- Isolate, consolidate, and minimize security controls.

Least Common Mechanism

- Mechanisms used to access separate resources should not be shared.
- Avoid the situation where errors of the mechanism while accessing one resource allow compromise of all resources accessible by the mechanism.
- Separate machines, separate networks, virtual machines can help fulfill this principle and avoid cross-contamination.
Reluctance to Trust

• Developers should assume that the environment in which their system resides is insecure.

• Trust, whether it is in external systems, code, people, etc., should always be closely held and never loosely given.

• When building an application, software engineers should anticipate malformed input from unknown users. Even if users are known, they are susceptible to social engineering attacks, making them potential threats to a system.

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, can get
Complete Mediation

- All accesses to entities are checked to ensure they are allowed, irrespective of who is accessing what. Check should ensure attempted access do not violate security properties.
- Fool-proof method to identify the source of every request.
- Security architect can evaluate each possible interaction among components.
- Easier to evaluate if principle of economy of mechanism has been followed.
- Caching permissions can increase the performance of a system, but at the cost of allowing secured objects to be accessed.

Psychological Acceptability

- Security mechanisms should not make the resource more difficult to access for legitimate users.
- Interface of security mechanisms should be designed to match the mental mode of the users.
- Otherwise, users will either attempt to bypass security mechanisms or will use them incorrectly.
Promoting Privacy

• Many users consider privacy a security concern. Try not to do anything that might compromise the privacy of the user. Be as diligent as possible in protecting any personal information a user does give your program.

• If an attacker breaks into a software system and steals private information about a vendor's customers, then their customers may lose their confidence in that software system.


Fail-safe Defaults

• Unless a subject is granted explicit access to an object, it should be denied access to that object.

• Default access to an object is “none.”

• Do not install all features and capabilities by default. Provide an easy mechanism to enable other features.

• Example: Internet browsers requesting a resource on behalf of the users request.
  – Fetching and displaying based upon is a form of granting permission based upon user-selected URL.
  – URL can be specified as absolute or relative path.
  – Not straightforward to reject all invalid URLs.
  – Guidance: Deny all URLs unless their form specifically be verified as being valid.
Minimize Your Attack Surface

- Keep entry points to a minimum and allow users to enable functionality as needed.
  - Number of open sockets (TCP and UDP)
  - Number of open named pipes
  - Number of open remote procedure call (RPC) endpoints
  - Number of services
  - Number of services running by default
  - Number of services running in elevated privileges
  - Number of dynamic content Web pages
  - Number of account you add to administrator group.

References