.. IoT Security Initiative Security Design Best Practices

Whether designing, implementing, or testing the security of system, one should keep in mind key considerations for core goals and objectives that drive any security posture. These goals and objectives really will apply to the security and defense of any system and its "asset" elements; be it a computer, a building, or a person. Constantly keeping these goals and objectives in mind, together with the security design best practices below, will help ensure a solid security foundation is being established.

VIEW THE PDF

System Security Core Goals

defense objectives

- Increase attack complexity
- Extend compromise duration
- Increase compromise complexity
- Inhibit or slow attacker actions
- Deny useful information to the attacker
- Confuse the attacker

- Increase visibility of an attack or compromise
- Limit impact: operational, system, and data
- Facilitate and promote attack attribution
- Ensure operational resiliency

When designing, implementing, and assessing system security, there are a number of best practices to consider and review that will help ensure robust use of controls. While not to be considered a definitive listing, it is a comprehensive view of key considerations for broad system security design.

Security Design Best Practices

- Implement end-to-end security and privacy of system data and operations; from fielded devices, to the server, to the end-user on a management web portal.
- Restrict system and network communications to only known, authorized system components where able.
- Expect software vulnerabilities & validate secure coding using automated & manuals means.
- Utilize secure coding best practices – and conduct static and dynamic code vulnerability testing.
- Implement reliable and securely managed software/firmware update mechanisms throughout the solution that are link authenticated, encrypted as needed, and verified for authenticity and integrity before implementation on system.
- Fingerprint and validate the integrity of critical system operating thresholds or parameters.
- Utilize two-factor authenticated and encrypted remote management services.
- Use multi-factor authentication and time-bound, out-of-band verification routines to effect critical account/system changes.
- Make use of chip-level security and virtualization capabilities, and utilize crypto coprocessors for key creation and storage.
- Establish a defense-in-depth, or layered-approach, to system security with varying control types applied to the end-to-end solution.
- Apply threat-modeling concepts, continuously, for determining the application of security controls.
- Conduct threat modeling of the end-to-end system solution to baseline system threats and risk.
- Create system designs and conduct security tests relative to trust boundaries both in-system and end-to-end.
- Fuzz test communications stacks and protocols – use pre-assessed stacks and libraries.
- Implement and operate only the system services that are necessary for the function of the system/solution.
- Apply extra security design and testing diligence to all listening network services on a system.
- Plan for system failure – design in redundancy for critical functions - and design to fail secure.
- Do not build your own encryption functions – and have encryption implementations security-reviewed.
- Do not code in "secret" login bypasses/access methods – even if just for seemingly temporary Dev/Test purposes.
- Deploy systems and services based on a least-privilege model.
- Compartmentalize communication IO in system design wherever possible; and run these

services at least-privilege levels.
- Run as much system code as possible at the lowest privilege/permission level possible; and as little as you can in highest privilege/permission level.
- Compartmentalize critical systems, services, communications, resources, and environments.
- Consider restricting or tightly controlling access to system components, firmware, and technical data for critical systems.
- Do not trust data input – sanitize to what is needed and expected for the function on intake.
- Utilize parameterized queries instead of custom code for all SQL statement communications with database management systems (DBMS).
- If creating default credentials, create quality randomized and unique passwords/symmetric-keys.
- All stored secrets are vulnerable to compromise with enough time and/or resources – ALL. Design and mitigate weakness per risk tolerance.
- Ship with, and maintain, security updated open source libraries used in products and services created.
- Validate system security approach and implementation throughout the SDLC.
- Conduct security/vulnerability testing on both software code and finished systems.
- Whitelist and control both ingress and egress of device/system communications where able.
- Use whitelisting methods over blacklisting when feasible.
- Shed technology attack surface whenever and wherever possible in design and development.
- When in question over possible data sensitivity or privacy, just encrypt.
- Implement application data layer encryption in addition to communications link layer encryption for higher risk data communications.
- Utilize trusted platform modules (TPM), secure elements (SE), and other hardware security modules (HSM) for storing and processing cryptographic secrets.
- Make use of secure boot, secure micro-kernels and hardware virtualization capabilities whenever possible.
- Protect the system enclosure and electronics from physical access, probing, and attack.
- Set data constraints and set “whitelist” operating parameters for critical software/physical system functions/operation.
- Utilize mutually authenticated and encrypted RF communications.
- Ensure the identity, authenticity, and integrity of communicated data by authenticating both the communication link and the data communicated.
- Use high-iteration, heavy-salt, key derivation functions such as scrypt/jane, bcrypt and PBKDF2 for storing account passwords.
- Use sufficiently large, as well as high quality, entropy for encryption routines.
- Expire login sessions at a reasonable time limit and use only secure session tokens.
- Implement escalating, timed-lockout of account logins upon a number of failed login attempts.