

Table of contents

Introduction	3
Secure SDLC	3
Security design principles	4
Secure by default	4
Simplicity and modularity of design	5
Defense in depth	5
Fail-safe defaults	6
Trusted security components	6
Complete mediation	6
Open design	7
Separation of privileges	7
Principle of least privilege	7
Least common mechanism	8
Psychological acceptability	8
Work factor	8
Minimize attack surface area	9
Do not trust services	9
Zero trust	9
Economic security and performance security	10
How can we (HCLTech) help?	10
References	11

Abstract

Security has taken a central place when it comes to the design and development of new products due to the products being connected to the internet and to the ever-increasing security landscape. This paper tries to provide an overview of security design principles that should be adhered to when one starts building or developing a new product.

The different regulatory and standard bodies have defined secure product design principles. Some of them are NIST and OWASP, in addition to Saltzer and Schroeder's early work. This paper takes these security principles and puts them here in the context of product security.

Introduction

Information security has been crucial since the early days of computing devices, but with the advent of the public internet in the 1990s, cyber security emerged as a top priority issue for organizations. According to the FBI Web Crime Report for 2021, cybercrime resulted in financial losses worth \$6.9 billion compared to \$4.2 billion in 2020[1]. This is a significant increase of 1.64 times just in a single year.

Given these risks, it is imperative for companies to take all the necessary steps to secure their digital assets from cyberattacks, information breaches, unauthorized access and other cybersecurity hazards. Adopting a secure Software Development Life Cycle (SDLC) during product development, incorporating security design principles, is key to safeguarding products from increasingly powerful cyberattacks.

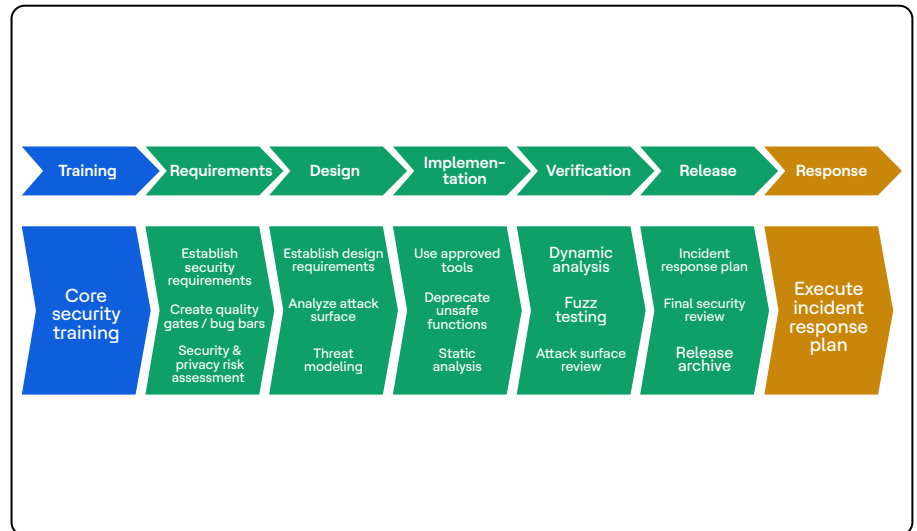
Secure SDLC

In today's interconnected world, the introduction of IoT devices has further increased cyber vulnerabilities, according to the latest Gartner report. Many organizations previously approached cybersecurity as an afterthought, implementing measures only after designing and implementing the products. However, this reactive approach often results in higher costs.

As per a report compiled by Microsoft Corporation and iSec Partners, the cost to correct the fault at design time is 30 times less than fixing the fault after deployment.

Indeed, cybersecurity should be an integral part of SDLC. A secure SDLC ensures that security considerations are prioritized right from the inception phase of product development.

The below diagram illustrated the secure SDLC process. While, this paper will not cover all the steps of secure SDLC, it will focus on the design aspect. Specifically, it will delve into the design part of the below diagram emphasizing system security design principles.



Graphic Source: Microsoft

Security design principles

Early seminal work in this subject was conducted by Saltzer and Schroeder in 1975[2], where they outlined ten guiding design principles in one of the most cited articles, 'The Protection of Information in Computer Systems'. Besides their contribution, various standardization organizations have also established their guiding design principles. This paper is going to refer to NIST (National Institute of Standards and Technology) [3] and OWASP [4] in addition to Saltzer and Schroeder's work. These cybersecurity design principles are essential for the secure development of all computing systems as well as embedded device products.

Secure by default

'Secure by default' implies integrating security into the product from its inception. It ensures that appropriate right security measures are built into both software and hardware during the conceptual phase. A system is considered secure by default when the manufacturer's default settings are secure and usable. The following are examples of the security by default setting:

- Disabling all the unused ports and services
- Complex password rules and password change on the first login
- No use of insecure and deprecated protocols and programming constructs

- Password aging

Some of the above settings may be configurable. If end users wish to change the security settings, they should be informed of the consequences of taking such actions.

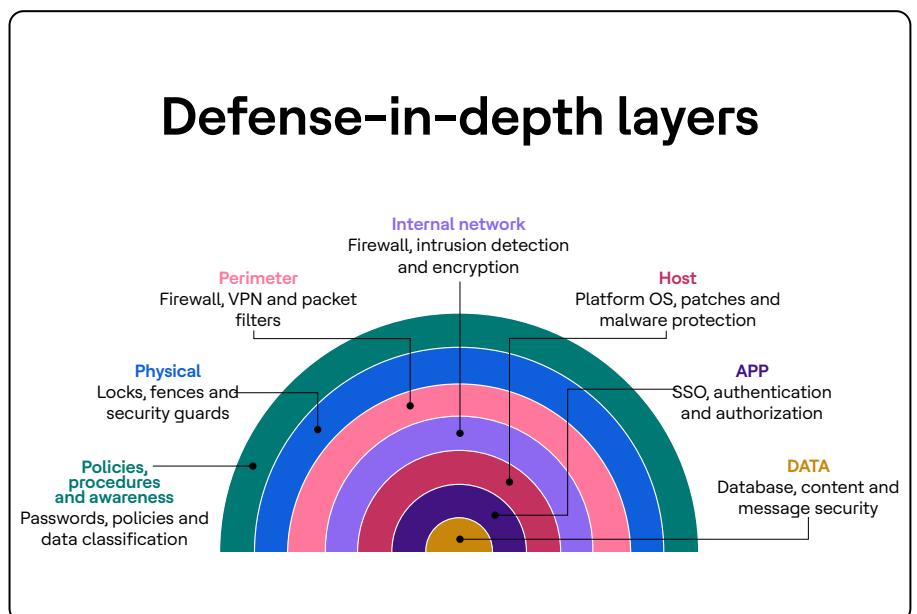
Simplicity and modularity of design

Simplicity over complexity should be favored. A simple and modular design is easier to test, validate and maintain. At runtime, it is easier to address the vulnerabilities in the system.

A complex system is more likely to have undetected access paths which can be challenging and mitigate runtime. To ensure system security, it is crucial to test the system at a micro level, including scanning the source code for each possible flow and physically checking the hardware. Maintaining a simple design facilitates the application of this thorough testing approach.

Defense in depth

Defense-in-depth refers to the concept of layered defense, meaning that the asset should have multiple layers of defense. This approach makes attacks less effective as it requires breaching multiple layers of authentication, authorization and verifications. Multi-Factor Authentication (MFA) is a prime example of defense in depth. MFA requires more than one method for the user to be authenticated such as user password and One-Time Password (OTP) sent to the user's personal phone number. The in-depth defense complicates efforts for hackers to exploit the system as they must overcome multiple controls before gaining access and steal valuable information. The following pic depicts the defense-in-depth layers for an organization's valuable data.



Source- [What's the difference between zero trust vs. defense in depth? \(techtarget.com\)](https://techtarget.com)

Fail-safe defaults

If a product fails to function as per expectations, it should do so in a predefined secure state. The system designers must consider the active security controls of a failed or non-functioning system. That means a failure should not make it easier for attackers to access the system or steal the data. Some examples of fail-safe security mechanisms are listed below.

- Failing should not result in an escalation of privileges for any user
- Failing should not increase the attack surface area
- Failing should not make valuable data accessible to malicious users
- Failing should not allow the user to bypass access controls.

Trusted security components

Security-related components in the system (software or hardware-based implementation of cryptographic algorithms, TLS stack, Certificates, etc.) should adhere to standards and come from trustworthy sources. Without these criteria, it's impossible to ensure a robust security foundation for the system. For example, cryptographic libraries should be from reputed sources, FIPS certified and support industry-standard cryptographic algorithms.

Complete mediation

Complete mediation mandates that every access to system objects should be validated before any action is allowed. For example, when a user or service tries to access the file, the necessary permission on the file is checked by the system or operating system. If a subject has requisite permission, the access is granted. The system should not rely on cached permission for subsequent access requests. Each time a user needs to read the file, the operating system should revalidate the latest permissions granted to the user on the file.

To illustrate further, consider the Domain Name System (DNS) which caches the mapping between hostnames and IP addresses. A malicious user can poison the cached information by associating a fake IP address with a name, causing the host to route connections to that malicious host controlled by a hacker, resulting in a security breach. The complete mediation principle strictly prohibits relying on cached information and enforces validation of every access request against the current, real-time source of information.

Open design

According to this principle, security should not rely on the complexity or secrecy of the design and implementation. An open and transparent design allows for greater security, thereby increasing the likelihood of identifying and addressing flaws. It is based on Kerckhoff's principle [4] of cryptography stated by Dutch-born cryptographer Auguste Kerckhoff in the 19th century. The principle holds that a cryptosystem should be secure, even if everything about the system, except the key, is in public knowledge. Let us summarize the key takeaways:

- The security design should be open to the world
- The security should depend upon the knowledge of secret keys or passwords possessed by the attacker instead of the knowledge of security design
- Cryptography keys should be kept secret.

Separation of privileges

This principle has different meanings in various contexts and this paper aims to cover all the available definitions.

- According to Saltzer and Schroeder, separation of privileges is a security protection mechanism where access to the system requires more than one key or secret, ideally held by different actors. For example, access to a critical data center might require both biometric or lock keys, each known to two different individuals who do not share their respective access methods.
- The access to a security object should be based on multiple conditions. A historical example from some UNIX OS flavors is the requirement to meet two conditions to get access to the super user command. If either conditions was not met, access was denied.
- In another context, separation of privileges refers to an information technology that best practices where organizations assign different levels based on roles. For instance, a system administrator needs higher privileges compared to a normal user due to their responsibilities.

Principle of least privilege

This principle states that the user or processes should be granted the minimum privileges necessary to perform their duties. Privileges should be elevated only when needed and reverted to the minimal level afterward. If a developer needs admin privileges on a test machine in the production environment to test his or her code, then the user should be granted that access on a temporary basis for the time required to test the code. Once testing is completed, then the developer's elevated privileges from that machine should be revoked.

Least common mechanism

According to this principle, resource access mechanisms should be shared minimally, effectively restricting unnecessary sharing. Information can flow along the channel when access to an object is shared or when the object itself is shared. An example from a banking website can illustrate this concept. When a banking site is accessible to everyone, malicious users can overwhelm it with multiple access requests simultaneously, causing a distributed denial of service attack. This results in legitimately being unable to access the site. The problem arises because the banking site shares the network connection among all the users, which results in availability issues. To mitigate this, mechanisms should be in place to differentiate malicious users from the genuine ones. It may not seem easy to accomplish, but this will ensure that access to the site is not shared between malicious and genuine users.

Psychological acceptability

According to this principle, the security mechanism should not make it difficult for the end users to access the system. In addition to that, the security mechanism should be easily adaptable by end-users. What it means is that the user can easily navigate the system without compromising security. Usability and security work in opposite directions. If we compromise on security to make the system user-friendly, it is going to make the system less secure. On the flip side, very robust security is going to make the system difficult to use. We should balance security and usability. A very simple example is going to make it easy to understand. When a user is unable to access a website because of an invalid password, the informative message should not explicitly mention that the wrong password was entered. This is going to cut the effort in half for the hackers if he or she is trying to hack the site using a brute-force attack.

Work factor

According to this principle, one should compare the cost required to bypass the security system implemented with the resources available at the disposal of the malicious user. The cost to circumvent the security mechanism is called the work factor. It directly relates to the strength of the cipher suite algorithm used.

For instance, if a user is using a four-letter lower-case password and someone wants to know that password, a hacker will need 264 or 456,976 combinations for an exhaustive search using a brute force attack. Now, if an intruder uses a normal computer, it will be very difficult for him to crack the password. But if an intruder has access to a very powerful computer (trying one million combinations per second), then it will be easy to break the password. The security mechanism should be designed by taking the 'work factor' into consideration.

Minimize attack surface area

An attack surface area is, essentially, all the externally exposed parts of your system. This is the area that hackers can use to gain access to your system to exploit vulnerabilities. An attack surface may be digital or physical, social engineering type or IoT type. If we don't implement the encryption mechanism for data in transit, we are giving malicious users a chance to tamper or read the data. Similarly, if we do not implement protection against phishing attacks, we are giving malicious users a chance to carry out a social engineering attack. An ideally secure system will have minimal entry points in the system to minimize the attack surface.

Do not trust services

Very frequently, web applications rely on external services for retrieving data or to use additional functionality. According to this principle, one should never trust these services from a security perspective. For example, an e-commerce application might call the payment service from a payment gateway to process the payment from the customer. As per this principle, this application should always check the validity of the data as per the predefined protocol. A secure application should not give external service elevated permissions in the app.

Zero trust

NIST special publication 800-207[7] explains the zero trust architecture in detail. As per the United Kingdom National Cyber Security Centre (NCSC)[8], the following principles should be followed for zero trust architecture.

- A single, strong source of user identity
- User authentication
- Machine authentication
- Additional context, such as policy compliance and device health
- Authorization policies to access an application
- Access control policies within an application.

Economic security and performance security

The security measures should not be overkill from both a financial and performance standpoint. It's essential to carefully analyze the trade-off between the level of security and its impact on finances and performance. An optimal security solution needs to be determined—one that addresses the system's security requirements without unduly burdening it by compromising cost-effectiveness, speed, or performance.

How can we (HCLTech) help?

HCLTech has a strong team of Product Security Architects and Engineers who can help in the secure design of your products. Our Product Security Architects and Engineers can help you secure your devices and applications across industry verticals. Automotive, medical devices, aerospace & defense, semiconductor, industrial control systems and telecom and consumer electronics are some of the industry verticals where we help our clients secure the design and development of products (devices and applications).

References

- [1] [2021 Internet Crime Report - Homeland Security Digital Library \(hsdl.org\)](#)
- [2] [A Contemporary Look at Saltzer and Schroeder's 1975 Design Principles | IEEE Journals & Magazine | IEEE Xplore](#)
- [3] [\(PDF\) Examination of security design principles from NIST SP 800-160 \(researchgate.net\)](#)
- [4] [Security by Design Principles - OWASP](#)
- [5] [Kerkhof's principle - Wikipedia](#)
- [6] [Zero trust security model - Wikipedia](#)
- [7] [Zero Trust Architecture \(nist.gov\)](#)
- [8] [Network architectures - NCSC.GOV.UK](#)
- [9] [Engineering Trustworthy Secure Systems \(nist.gov\)](#)

Co-author information



Sachin Kumar

Sachin has more than 25 years of experience in IT industry. He works as Senior Solution Director for HCLTech ERS office – Product Security COE. He has diverse experience in security and worked for long time in Fintech domain for a big Swiss bank in Europe. He holds Bachelor of Technology degree in Electrical Engineering. He is an alumnus of IIM Kozhikode from where he completed his executive MBA with specialization in IT management and Finance. He holds multiple certifications like Microsoft Azure AI Foundation and TOGAF. Lately he is venturing into consulting the organizations on the responsible usage of AI.



Mayank Babu Rastogi

Mayank has 25+ years of industry experience in different technology areas including Product Security, Trustworthy AI, Embedded systems and IoT. He has extensive experience in leading variety of functions like product engineering, technical solutioning, engineering program management and technology partnerships. In product security, he has vast experience in Secure Product Development Life Cycle, Security standards & regulations, Classic as well as Post Quantum Cryptography etc.

HCLTech | Supercharging Progress™

HCLTech is a global technology company, home to 222,000+ people across 60 countries, delivering industry-leading capabilities centered around Digital, Engineering and Cloud powered by a broad portfolio of technology services and software. The company generated consolidated revenues of \$12.3 billion over the 12 months ended December 2022. To learn how we can supercharge progress for you, visit hcltech.com.

hcltech.com

