

**CYB 309 SYSTEMS SECURITY**

Adeniyi Akanni (PhD)

Adeniyi.akanni@calebuniversity.edu.ng

# CYB 309 Systems Security (L-2; T-0; P-0) 2 Units Prerequisite – CYB201, CYB 202

Security Principles, Account Security, File System Security, Assessing Risk, Risk Analysis, and Encryption. The student’s basic network and operating system skills will be expanded to include planning, implementation, and auditing of a system’s security package. Secure design and secure coding principles, practices, and methods including least privilege, threat modelling, and static analysis. Covers common vulnerabilities suchas buffer overruns, integer overflows, injection attacks, cross-site scripting, and weak error handling.

**SECURITY PRINCIPLES**

The basic principles of cyber security lie in the CIA triad: Confidentiality, Integrity and Availability. On these hinged other security practices to protect and device means of putting at bay every attack attempts.

### Confidentiality

Confidentiality measures are designed to prevent unauthorized disclosure of information. The purpose of the confidentiality principle is to keep personal information private and to ensure that it is visible and accessible only to those individuals who own it or need it to perform their organizational functions.

### Integrity

Consistency includes protection against unauthorized changes (additions, deletions, alterations, etc.) to data. The principle of integrity ensures that data is accurate and reliable and is not modified incorrectly, whether accidentally or maliciously.

### Availability

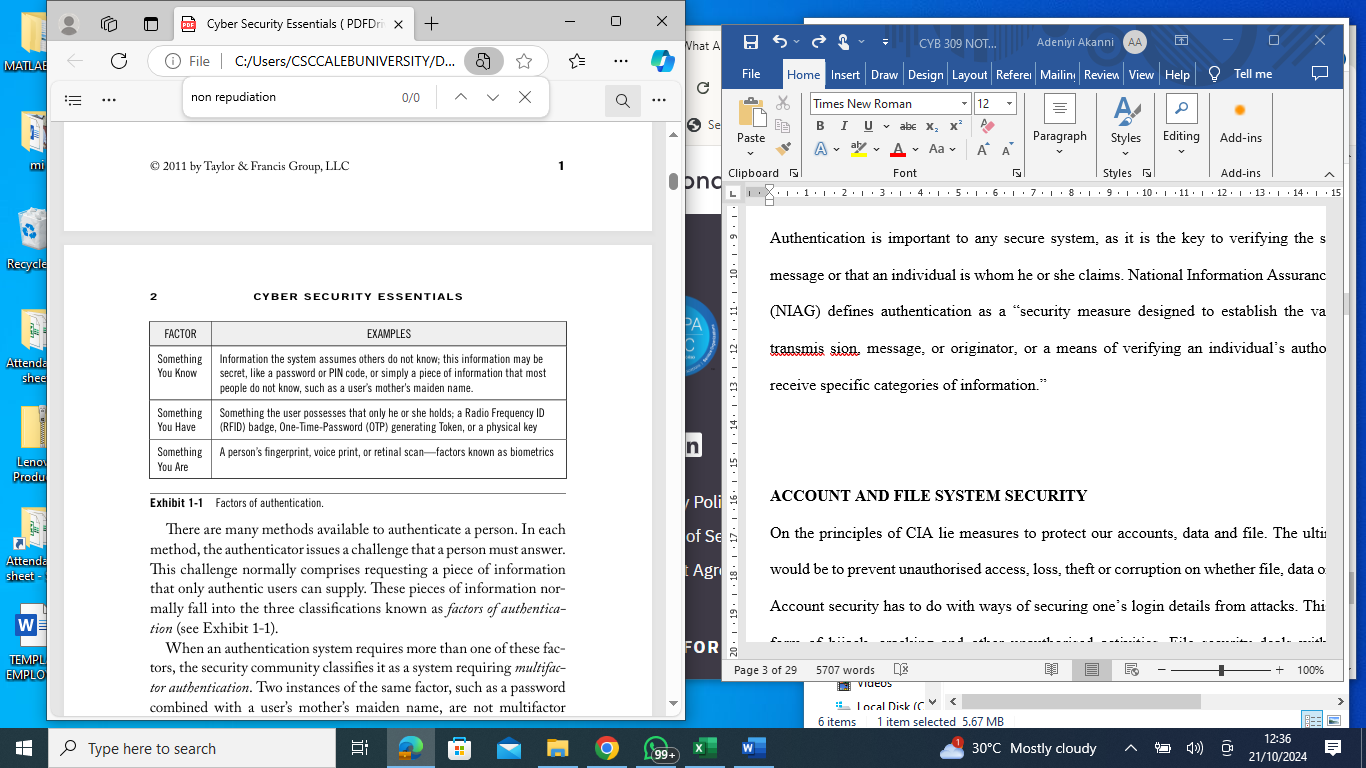
Availability is the protection of a system’s ability to make software systems and data fully available when a user needs it (or at a specified time). The purpose of [availability is to make the technology infrastructure](https://www.imperva.com/learn/availability/high-availability/), the applications and the data available when they are needed for an organizational process or for an organization’s customers.

Other Fundamental Security Concepts

Authentication, authorization, and nonrepudiation are tools that system designers can use to maintain system security with respect to confidentiality, integrity, and availability.

Authentication

Authentication is important to any secure system, as it is the key to verifying the source of a message or that an individual is whom he or she claims. National Information Assurance Glossary (NIAG) defines authentication as a “security measure designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual’s authorization to receive specific categories of information.”



Authorization

While authentication relates to verifying iden tities, authorization focuses on determining what a user has permission to do. The NIAG defines authorization as “access privileges granted to a user, program, or process.” After a secure system authenticates users, it must also decide what privileges they have. For instance, an online banking application will authenticate a user based on his or her credentials, but it must then determine the accounts to which that user has access. Additionally, the system determines what actions the user can take regarding those accounts, such as viewing balances and making transfers.

Nonrepudiation

Imagine a scenario wherein Alice is purchasing a car from Bob and signs a contract stating that she will pay $20,000 for the car and will take ownership of it on Thursday. If Alice later decides not to buy the car, she might claim that someone forged her signature and that she is not responsible for the contract. To refute her claim, Bob could show that a notary public verified Alice’s identity and stamped the document to indicate this verifica tion. In this case, the notary’s stamp has given the contract the prop erty of nonrepudiation, which the NIAG defines as “assurance the sender of data is provided with proof of delivery and the recipient is provided with proof of the sender’s identity, so neither can later deny having processed the data.” In the world of digital communications, no notary can stamp each transmitted message, but nonrepudiation is still necessary. To meet this requirement, secure systems normally rely on asymmetric (or public key) cryptography. While symmetric key systems use a single key to encrypt and decrypt data, asymmetric systems use a key pair. T hese systems use one key (private) for signing data and use the other key (public) for verifying data. If the same key can both sign and verify the content of a message, the sender can claim that anyone who has access to the key could easily have forged it. Asymmetric key systems have the nonrepudiation property because the signer of a message can keep his or her private key secret.

**ACCOUNT AND FILE SYSTEM SECURITY**

On the principles of CIA lie measures to protect our accounts, data and file. The ultimate goals would be to prevent unauthorised access, loss, theft or corruption on whether file, data or accounts.

Account security has to do with ways of securing one’s login details from attacks. This can be in form of hijack, cracking and other unauthorised activities. File security deals with measures required to protect our details, records and data from unauthorised access.

The Challenges of System security

System security is both fascinating and complex. Some of the reasons are as follows:

1. System security is not as simple as it might first appear to the novice. The requirements seem to be straightforward; indeed, most of the major requirements for security services can be given self-explanatory one-word labels: confidentiality, authentication, nonrepudiation, and integrity. But the mechanisms used to meet those requirements can be quite complex, and understanding them may involve rather subtle reasoning.

2. In developing a particular security mechanism or algorithm, one must always consider potential attacks on those security features. In many cases, successful attacks are designed by looking at the problem in a completely different way, therefore exploiting an unexpected weakness in the mechanism.

3. Because of Point 2, the procedures used to provide particular services are often counterintuitive. Typically, a security mechanism is complex, and it is not obvious from the statement of a particular requirement that such elaborate measures are needed. Only when the various aspects of the threat are considered do elaborate security mechanisms make sense.

4. Having designed various security mechanisms, it is necessary to decide where to use them. This is true both in terms of physical placement (e.g., at what points in a network are certain security mechanisms needed) and in a logical sense [e.g., at what layer or layers of an architecture such as TCP/IP (Transmission Control Protocol/Internet Protocol) should mechanisms be placed].

5. Security mechanisms typically involve more than a particular algorithm or protocol. They also require that participants be in possession of some secret information (e.g., an encryption key), which raises questions about the creation, distribution, and protection of that secret information. There may also be a reliance on communications protocols whose behavior may complicate the task of developing the security mechanism. For example, if the proper functioning of the security mechanism requires setting time limits on the transit time of a message from sender to receiver, then any protocol or network that introduces variable, unpredictable delays may render such time limits meaningless.

6. System security is essentially a battle of wits between a perpetrator who tries to find holes, and the designer or administrator who tries to close them. The great advantage that the attacker has is that he or she needs only find a single weakness, while the designer must find and eliminate all weaknesses to achieve perfect security.

7. There is a natural tendency on the part of users and system managers to perceive little benefit from security investment until a security failure occurs.

8. Security requires regular, even constant monitoring, and this is difficult in today’s short term, overloaded environment.

9. Security is still too often an afterthought to be incorporated into a system after the design is complete, rather than being an integral part of the design process.

10. Many users and even security administrators view strong security as an impediment to efficient and user-friendly operation of an information system or use of information.

**RISK ASSESSMENT AND ANALYSIS**

Risk: The possibility that the occurrence of an event will adversely affect the achievement of the organization's objectives.

T**he main four types of risk are:**

* **strategic risk** - eg a competitor coming on to the market.
* **compliance and regulatory risk** - eg introduction of new rules or legislation.
* **financial risk** - eg interest rate rise on your business loan or a non-paying customer.
* **operational risk** - eg the breakdown or theft of key equipment.

**Risk Management Framework**

The Risk Management Framework is a template and guideline used by companies to identify, eliminate and minimize risks. It was originally developed by the National Institute of Standards and Technology to help protect the information systems of the United States government.

**Component of Risk Management Framework**

The 5 Components of Risk Management Framework. There are at least five crucial components that must be considered when creating a risk management framework. They include:

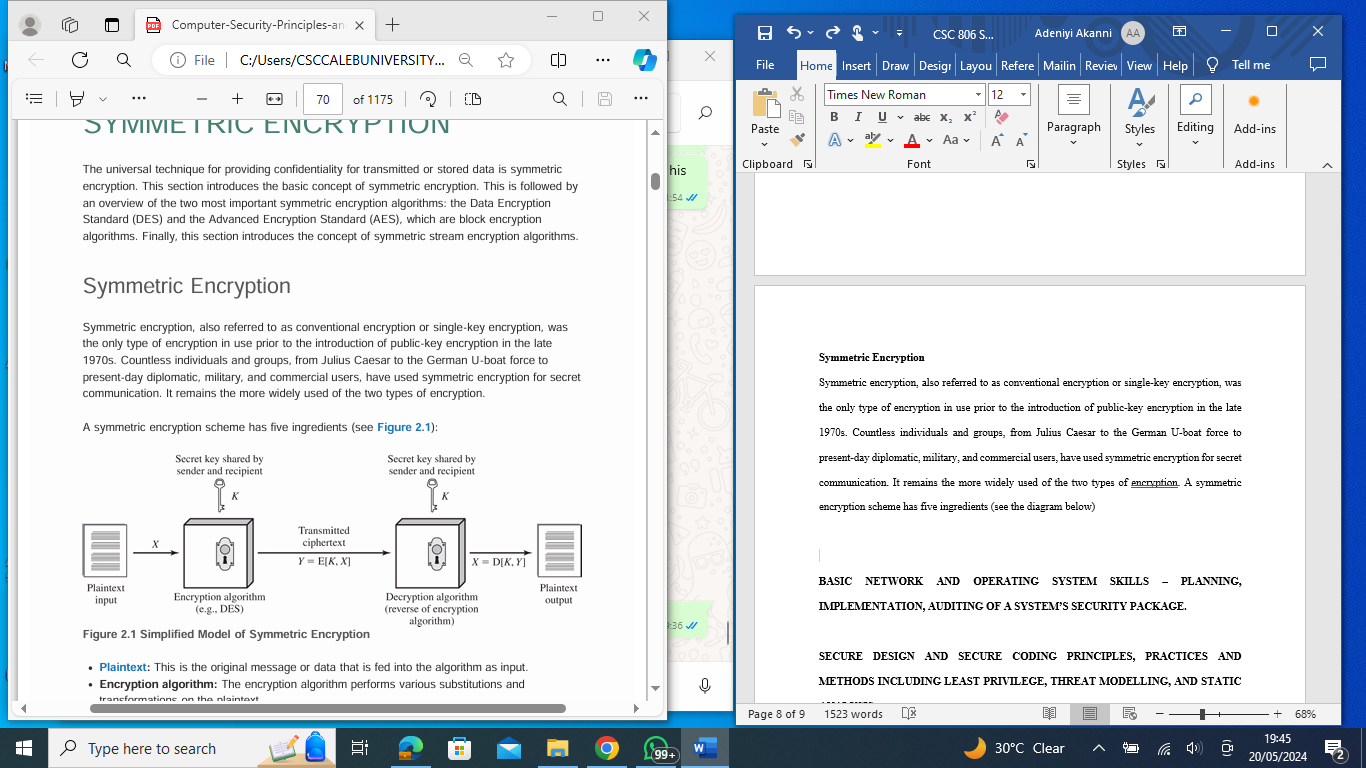
1. Risk Identification. The first step in identifying the risks a company faces is to define the risk universe. The risk universe is simply a list of all possible risks. Examples include IT risk, operational risk, [regulatory risk](https://www.investopedia.com/terms/r/regulatory_risk.asp), legal risk, political risk, strategic risk, and credit risk. After listing all possible risks, the company can then select the risks to which it is exposed and categorize them into core and non-core risks. Core risks are those that the company must take in order to drive performance and long-term growth. Non-core risks are often not essential and can be minimized or eliminated completely.
2. Risk Measurement and Assessment. Risk measurement provides information on the quantum of either a specific risk exposure or an aggregate risk exposure and the probability of a loss occurring due to those exposures. When measuring specific risk exposure it is important to consider the effect of that risk on the overall risk profile of the organization. Some risks may provide diversification benefits while others may not. Another important consideration is the ability to measure an exposure. Some risks may be easier to measure than others. For example, market risk can be measured using observed market prices, but measuring operational risk is considered both an art and a science.
3. Risk Mitigation. Having categorized and measured its risks, a company can then decide on which risks to eliminate or minimize, and how many of its core risks to retain. Risk mitigation can be achieved through an outright sale of assets or liabilities, buying insurance, hedging with derivatives, or diversification.
4. Risk Reporting and Monitoring. It is important to report regularly on specific and aggregate risk measures in order to ensure that risk levels remain at an optimal level. Financial institutions that trade daily will produce daily risk reports. Other institutions may require less frequent reporting. Risk reports must be sent to risk personnel who have the authority to adjust (or instruct others to adjust) risk exposures.
5. Risk Governance. Risk governance is the process that ensures all company employees perform their duties in accordance with the risk management framework. Risk governance involves defining the roles of all employees, [segregating duties](https://www.investopedia.com/terms/s/separation-powers.asp), and assigning authority to individuals, committees, and the board for approval of core risks, risk limits, exceptions to limits, and risk reports, and also for general oversight.

**ENCRYPTION**

Encryption is a measure for preserving confidentiality. The commonest type is the Symmetric Encryption. The second major type is the Asymmetric Encryption. While the former uses the same key, the latter uses different.

**Symmetric Encryption**

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the introduction of public-key encryption in the late 1970s. Countless individuals and groups, from Julius Caesar to the German U-boat force to present-day diplomatic, military, and commercial users, have used symmetric encryption for secret communication. It remains the more widely used of the two types of encryption. A symmetric encryption scheme has five ingredients (see the diagram below)



**Plaintext:** This is the original message or data that is fed into the algorithm as input.

**Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.

**Secret key:** The secret key is also input to the encryption algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.

**Ciphertext**: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. **Decryption algorithm**: This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

There are two requirements for secure use of symmetric encryption:

1. We need a strong encryption algorithm. The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

2. The sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

**Attack On Symmetric Encryption**

There are two main approaches to attacking Symmetric Encryption:

1. **Cryptanalysis**. Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext, or even some sample plaintext ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used. If the attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.
2. Brute-force attack, is to try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success. That is, if there are x different keys, on average an attacker would discover the actual key after x/2 tries.

The most commonly used symmetric encryption algorithms are block ciphers. A block cipher processes the plaintext input in fixed-size blocks and produces a block of ciphertext of equal size for each plaintext block. The algorithm processes longer plaintext amounts as a series of fixed size blocks. The most important symmetric algorithms, all of which are block ciphers, are the Data Encryption Standard (DES), triple DES, and the Advanced Encryption Standard (AES).

**Asymmetric Encryption Algorithms**

In this subsection, we briefly mention the most widely used asymmetric encryption algorithms

**RSA**

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78]. The RSA scheme has since reigned supreme as the most widely accepted and implemented approach to public-key encryption.

**DIFFIE–HELLMAN KEY AGREEMENT**

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76] and is generally referred to as Diffie–Hellman key exchange, or key agreement. A number of commercial products employ this key exchange technique. The purpose of the algorithm is to enable two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages. The algorithm itself is limited to the exchange of the keys.

**DIGITAL SIGNATURE STANDARD**

The National Institute of Standards and Technology (NIST) published this originally as FIPS PUB 186 (Digital Signature Standard (DSS), May 1994). The DSS makes use of SHA-1 and presents a new digital signature technique, the Digital Signature Algorithm (DSA). The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There were further revisions in 1998, 2000, 2009, and most recently in 2013 as FIPS PUB 186–4. The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange.

**ELLIPTIC CURVE CRYPTOGRAPHY**

The vast majority of the products and standards that use public-key cryptography for encryption and digital signatures use RSA. The bit length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. This burden has ramifications, especially for electronic commerce sites that conduct large numbers of secure transactions. Recently, a competing system has begun to challenge RSA: elliptic curve cryptography (ECC). Already, ECC is showing up in standardization efforts, including the IEEE (Institute of Electrical and Electronics Engineers) P1363 Standard for Public-Key Cryptography.

**BASIC NETWORK AND OPERATING SYSTEM SKILLS – PLANNING, IMPLEMENTATION, AUDITING OF A SYSTEM’S SECURITY PACKAGE.**

A computer system security audit is a **systematic assessment of the risks, vulnerabilities, and controls of your IT** infrastructure. It is a way of providing a reasonable assurance on the effectiveness of internal control put in place.

# *TYPES OF IS AUDITS* There are different types of Information Systems Audits. Some of which are discussed below:

# Application Audits

## Application controls are divided into three general areas.

## Input

## Processing

## Output

# Systems Development Audits

## Systems development audits are directed at the activities of the systems analyst and programmers.

## There are three general areas of audit concern in the systems development process.

## Systems development standards

## Project management

## Program change control

# Computer Service Center Audits

## Normally, an audit of the *computer service* *center* is undertaken before any application audits to ensure the general integrity of the environment in which the application will function.

## Audits might be undertaken in several areas.

## Audits of computer service center operations require a high degree of technical training and familiarity with computer operations.

# Computer Service Center Audits

## environmental controls

## physical security of the center

## data release, reports, and computer programs

## management controls

# How do you audit your computer system security?

# Define the scope

The first step is to define the scope of your audit. This means deciding what aspects of your system you want to evaluate, such as hardware, software, networks, data, policies, procedures, and personnel. You also need to determine the objectives, criteria, and standards of your audit, such as legal requirements, industry best practices, or internal benchmarks.

1. **Gather the data**

The next step is to gather the data that will help you assess your system's security. This may include reviewing documents, interviewing staff, observing operations, or conducting tests. You can use various tools and methods to collect data, such as vulnerability scanners, penetration testers, log analyzers, or questionnaires. You should document your data sources, methods, and findings for future reference.

1. **Analyze the data**

The third step is to analyze the data that you have collected and identify any gaps, weaknesses, or issues in your system's security. You can use frameworks and models to help you organize and interpret the data, such as SWOT analysis, risk matrix, or control matrix. You should compare your data with your audit objectives, criteria, and standards, and prioritize the findings based on their impact and urgency.

1. **Report the results**

The fourth step is to report the results of your audit to the relevant stakeholders, such as management, clients, or regulators. You should present your findings in a clear, concise, and accurate manner, using charts, graphs, or tables to illustrate your points. You should also include recommendations for improving your system's security, such as remediation actions, preventive measures, or best practices.

1. **Implement the changes**

The fifth step is to implement the changes that you have recommended or agreed upon in your audit report. You should follow a project management approach to plan, execute, monitor, and control the changes, and assign roles and responsibilities to the team members involved. You should also communicate the changes to the affected parties, and provide training and support if needed.

1. **Evaluate the outcomes**

The final step is to evaluate the outcomes of your audit and the changes that you have implemented. You should measure the effectiveness and efficiency of your system's security, and compare it with the baseline or the target that you have set.

**SECURE DESIGN AND SECURE CODING PRINCIPLES, PRACTICES AND METHODS INCLUDING LEAST PRIVILEGE, THREAT MODELLING, AND STATIC ANALYSIS.**

Secure coding standards govern the coding practices, techniques, and decisions that developers make while building software. They aim to ensure that developers write code that minimizes security vulnerabilities. Development tasks can be solved in many different ways, with varying levels of complexity. Some solutions are more secure than others, and secure coding standards encourage developers and software engineers to take the safest approach, even if it is not the fastest.

For example, secure coding best practices often mandate a “default deny” approach to access permissions. Developers using secure coding techniques create code that denies access to sensitive resources unless an individual can demonstrate that they are authorized to access it.

There are several secure coding standards and coding security guides in widespread use today, including the [OWASP (Open Web Application Security Project) Secure Coding Practices](https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/01-introduction/05-introduction) and the [SEI CERT computer emergency response team (CERT) for the Software Engineering Institute (SEI) Coding Standards](https://wiki.sei.cmu.edu/confluence/display/seccode).

## Secure Coding Best Practices

OWASP provides a secure coding practices checklist that includes 14 areas to consider in your [software development life cycle](https://kirkpatrickprice.com/blog/what-is-a-secure-software-development-life-cycle). Of those secure coding practices, we’re going to focus on the top eight secure programming best practices to help you protect against vulnerabilities.

1. Security by Design
2. Password Management
3. Access Control
4. Error Handling and Logging
5. System Configuration
6. Threat Modelling
7. Cryptographic Practices
8. Input Validation and Output Encoding

### **Security by Design**

Security needs to be a priority as you develop code, not an afterthought. Organizations may have competing priorities where software engineering and coding are concerned. Following software security best practices can conflict with optimizing for development speed. However, a “security by design” approach that puts security first tends to pay off in the long run, reducing the future cost of technical debt and risk mitigation. An analysis of your source code should be conducted throughout your software development life cycle (SDLC), and security automation should be implemented.

### **Password Management**

Passwords are a weak point in many software systems, which is why [multi-factor authentication](https://kirkpatrickprice.com/video/pci-requirement-8-3-2-incorporate-multi-factor-authentication-remote-network-access/) has become so widespread. Nevertheless, passwords are the most common security credential, and following secure coding practices limits risk. You should require all passwords to be of adequate length and complexity to withstand any typical or common attacks. OWASP suggests several coding best practices for passwords, including:

* Storing only salted cryptographic hashes of passwords and never storing plain-text passwords
* Enforcing password length and complexity requirements
* Disable password entry after multiple incorrect login attempts

Implementing [logical access](https://kirkpatrickprice.com/blog/webinars-events/logical-access-fundamentals-for-enhanced-security-a-webinar-recap/) controls like password policies can do wonders for strengthening your organization’s security posture.

### Access Control

Take a “default deny” approach to sensitive data. Limit privileges and restrict access to secure data to only users who need it. Deny access to any user that cannot demonstrate authorization. Ensure that requests for sensitive information are checked to verify that the user is authorized to access it.

Learn more about [access controls for remote employees](https://kirkpatrickprice.com/video/soc-2-academy-access-controls-remote-employees/) and [cloud access management](https://kirkpatrickprice.com/blog/aws-iam-security-best-practices/).

### Error Handling and Logging

Software errors are often indicative of bugs, many of which cause vulnerabilities. Error handling and logging are two of the most useful techniques for minimizing their impact. Error handling attempts to catch errors in the code before they result in a catastrophic failure. Logging documents errors so that developers can diagnose and mitigate their cause.

Documentation and logging of all failures, exceptions, and errors should be implemented on a trusted system to comply with secure coding standards.

### System Configuration

Clear your system of any unnecessary components and ensure all working software is updated with current versions and patches. If you work in multiple environments, make sure you’re managing your development and production environments securely.

Outdated software is a major source of vulnerabilities and security breaches. Software updates include patches that fix vulnerabilities, making regular updates one of the most vital, secure coding practices. A [patch management system](https://kirkpatrickprice.com/blog/10-ways-to-conduct-patch-management/) may help your business to keep on top of updates.

### Threat Modeling

Document, locate, address, and validate are the four steps to threat modeling. To securely code, you need to examine your software for areas susceptible to increased threats of attack. Threat modeling is a multi-stage process that should be integrated into the software lifecycle from development, testing, and production.

### Cryptographic Practices

Encrypting data with modern cryptographic algorithms and following [secure key management best practices](https://kirkpatrickprice.com/webinars/encryption-key-management/) increases the security of your code in the event of a breach.

### Input Validation and Output Encoding

It’s important to identify all data inputs and sources and validate those classified as untrusted. You should utilize a standard routine for output encoding and input validation.

## How to Ensure Your Code Is Secure

By patching your systems regularly, you’re taking these secure coding guidelines to the next level. Patch and [vulnerability management](https://kirkpatrickprice.com/blog/notes-from-the-field-center-for-internet-security-control-7-continuous-vulnerability-management/) is focused on identifying risk and enabling systems to stay up to date. Through these methods and security testing, you’re ensuring that your code is properly checked for errors.

**COMMON VULNERABILITIES – BUFFER OVERRUNS, INTEGER OVERFLOWS, INJECTION ATTACKS, CROSS-SITE SCRIPTING WEAK ERROR HANDLING**

## Secure Coding Best Practices

OWASP provides a secure coding practices checklist that includes 14 areas to consider in your [software development life cycle](https://kirkpatrickprice.com/blog/what-is-a-secure-software-development-life-cycle). Of those secure coding practices, we’re going to focus on the top eight secure programming best practices to help you protect against vulnerabilities.

1. Security by Design
2. Password Management
3. Access Control
4. Error Handling and Logging
5. System Configuration
6. Threat Modeling
7. Cryptographic Practices
8. Input Validation and Output Encoding

### **Security by Design**

Security needs to be a priority as you develop code, not an afterthought. Organizations may have competing priorities where software engineering and coding are concerned. Following software security best practices can conflict with optimizing for development speed. However, a “security by design” approach that puts security first tends to pay off in the long run, reducing the future cost of technical debt and risk mitigation. An analysis of your source code should be conducted throughout your software development life cycle (SDLC), and security automation should be implemented.

### **Password Management**

Passwords are a weak point in many software systems, which is why [multi-factor authentication](https://kirkpatrickprice.com/video/pci-requirement-8-3-2-incorporate-multi-factor-authentication-remote-network-access/) has become so widespread. Nevertheless, passwords are the most common security credential, and following secure coding practices limits risk. You should require all passwords to be of adequate length and complexity to withstand any typical or common attacks. OWASP suggests several coding best practices for passwords, including:

* Storing only salted cryptographic hashes of passwords and never storing plain-text passwords
* Enforcing password length and complexity requirements
* Disable password entry after multiple incorrect login attempts

Implementing [logical access](https://kirkpatrickprice.com/blog/webinars-events/logical-access-fundamentals-for-enhanced-security-a-webinar-recap/) controls like password policies can do wonders for strengthening your organization’s security posture.

### Access Control

Take a “default deny” approach to sensitive data. Limit privileges and restrict access to secure data to only users who need it. Deny access to any user that cannot demonstrate authorization. Ensure that requests for sensitive information are checked to verify that the user is authorized to access it.

Learn more about [access controls for remote employees](https://kirkpatrickprice.com/video/soc-2-academy-access-controls-remote-employees/) and [cloud access management](https://kirkpatrickprice.com/blog/aws-iam-security-best-practices/).

### Error Handling and Logging

Software errors are often indicative of bugs, many of which cause vulnerabilities. Error handling and logging are two of the most useful techniques for minimizing their impact. Error handling attempts to catch errors in the code before they result in a catastrophic failure. Logging documents errors so that developers can diagnose and mitigate their cause.

Documentation and logging of all failures, exceptions, and errors should be implemented on a trusted system to comply with secure coding standards.

### System Configuration

Clear your system of any unnecessary components and ensure all working software is updated with current versions and patches. If you work in multiple environments, make sure you’re managing your development and production environments securely.

Outdated software is a major source of vulnerabilities and security breaches. Software updates include patches that fix vulnerabilities, making regular updates one of the most vital, secure coding practices. A [patch management system](https://kirkpatrickprice.com/blog/10-ways-to-conduct-patch-management/) may help your business to keep on top of updates.

### Threat Modeling

Document, locate, address, and validate are the four steps to threat modeling. To securely code, you need to examine your software for areas susceptible to increased threats of attack. Threat modeling is a multi-stage process that should be integrated into the software lifecycle from development, testing, and production.

### Cryptographic Practices

Encrypting data with modern cryptographic algorithms and following [secure key management best practices](https://kirkpatrickprice.com/webinars/encryption-key-management/) increases the security of your code in the event of a breach.

### Input Validation and Output Encoding

It’s important to identify all data inputs and sources and validate those classified as untrusted. You should utilize a standard routine for output encoding and input validation.

## How to Ensure Your Code Is Secure

By patching your systems regularly, you’re taking these secure coding guidelines to the next level. Patch and [vulnerability management](https://kirkpatrickprice.com/blog/notes-from-the-field-center-for-internet-security-control-7-continuous-vulnerability-management/) is focused on identifying risk and enabling systems to stay up to date. Through these methods and security testing, you’re ensuring that your code is properly checked for errors.

Vulnerabilities are the areas left uncovered by the security apparatus that can be exploited by threats. These can be an inroad to attacks. There are common threats which include:

1 Trojan Horse

Systems have mechanisms for allowing programs written by users to be executed by other users. If these programs are executed in a domain that provides the access rights of the executing user, the other users may misuse these rights. A text-editor program, for example, may include code to search the file to be edited for certain keywords. If any are found, the entire file may be copied to a special area accessible to the creator of the text editor. A code segment that misuses its environment is called a Trojan Horse.

1. Trap Door

The designer of a program or system might leave a hole in the software that only he is capable of using. This type of security breach was shown in the movie War Games. For instance, the code might check a specific user ID or password, and it might circumvent normal security procedures. Programmers have been arrested for embezzling from banks by including rounding errors in their code and having the occasional half-cent credited to their accounts. This account credititrg can add up to a large amount of money, considering the number of transactions that a large bank executes.

3 Logic Bomb

Consider a program that initiates a security incident only under certain circumstances. It would be hard to detect because under normal operations, there would be no security hole. However, when a predefined set of parameters were met, the security hole would be created. This scenario is known as a Logic Bomb. A programmer, for example, might write code to detect whether was still employed; if that check failed, a daemon could be spawned to allow remote access, or code could be launched to cause damage to the site.

4.Stack and Buffer Overflow.

The stack- or buffer-overflow attack is the most common way for an attacker outside the system, on a network or dial-up connection, to gain unauthorized access to the target system. An authorized user of the system may also use this exploit for privilege escalation.

Essentially, the attack exploits a bug in a program. The bug can be a simple case of poor programming, in which the programmer neglected to code bounds checking on an input field. In this case, the attacker sends more data than the program was expecting. By using trial and error, or by examining the source code of the attacked program if it is available, the attacker determines the vulnerability and writes a program to do the following:

* Overflow an input field, command-line argument, or input buffer-for example, on a network daemon - until it writes into the stack.
* Overwrite the current return address on the stack with the address of the exploit code loaded in the next step.
* Write a simple set of code for the next space in the stack that includes the commands that the attacker wishes to execute-for instance, spawn a shell.

There are threats associated with wireless networks. They include:

* **Accidental association:** Company wireless LANs or wireless access points to wired LANs in close proximity (e.g., in the same or neighboring buildings) may create overlapping transmission ranges. A user intending to connect to one LAN may unintentionally lock on to a wireless access point from a neighboring network. Although the security breach is accidental, it nevertheless exposes resources of one LAN to the accidental user.
* **Malicious association:** In this situation, a wireless device is configured to appear to be a legitimate access point, enabling the operator to steal passwords from legitimate users then penetrate a wired network through a legitimate wireless access point.
* **Ad hoc networks:** These are peer-to-peer networks between wireless computers with no access point between them. Such networks can pose a security threat due to a lack of a central point of control.
* **Nontraditional networks:** Nontraditional networks and links, such as personal network Bluetooth devices, barcode readers, and handheld PDAs pose a security risk both in terms of eavesdropping and spoofing.
* **Identity theft** (MAC spoofing): This occurs when an attacker is able to eavesdrop on network traffic and identify the MAC address of a computer with network privileges.
* **Man-in-the middle attacks:** This type of attack was described earlier in the context of the Diffie-Hellman key exchange protocol. In a broader sense, this attack involves persuading a user and an access point to believe that they are talking to each other, when in fact the communication is going through an intermediate attacking device. Wireless networks are particularly vulnerable to such attacks.
* **Denial of service** (DoS): This type of attack was discussed in detail in Chapter 7. In the context of a wireless network, a DoS attack occurs when an attacker continually bombards a wireless access point, or some other accessible wireless port, with various protocol messages designed to consume system resources. The wireless environment lends itself to this type of attack, because it is so easy for the attacker to direct multiple wireless messages at the target.
* **Network injection:** A network injection attack targets wireless access points that are exposed to nonfiltered network traffic, such as routing protocol messages or network management messages. An example of such an attack is one in which bogus reconfiguration commands are used to affect routers and switches to degrade network performance.

**SECURING WIRELESS TRANSMISSIONS**

The principal threats to wireless transmission are eavesdropping, altering or inserting messages, and disruption. To deal with eavesdropping, two types of countermeasures are appropriate:

* **Signal-hiding techniques**: Organizations can take a number of measures to make it more difficult for an attacker to locate their wireless access points, including turning off service set identifier (SSID) broadcasting by wireless access points; assigning cryptic names to SSIDs; reducing signal strength to the lowest level that still provides requisite coverage; and locating wireless access points in the interior of the building, away from windows and exterior walls. Greater security can be achieved by the use of directional antennas and of signal-shielding techniques.
* **Encryption:** Encryption of all wireless transmission is effective against eavesdropping to the extent that the encryption keys are secured.

**SECURING WIRELESS ACCESS POINTS**

The main threat involving wireless access points is unauthorized access to the network. The principal approach for preventing such access is the IEEE 802.1X standard for port-based network access control. The standard provides an authentication mechanism for devices wishing to attach to a LAN or wireless network. The use of 802.1X can prevent rogue access points and other unauthorized devices from becoming insecure backdoors.

**SECURING WIRELESS NETWORKS**

recommends the following techniques for wireless network security:

1. Use encryption. Wireless routers are typically equipped with built-in encryption mechanisms for router-to-router traffic.

2. Use anti-virus and anti-spyware software, and a firewall. These facilities should be enabled on all wireless network endpoints.

3. Turn off identifier broadcasting. Wireless routers are typically configured to broadcast an identifying signal so that any device within range can learn of the router’s existence. If a network is configured so authorized devices know the identity of routers, this capability can be disabled to thwart attackers.

4. Change the identifier on your router from the default. Again, this measure thwarts attackers who will attempt to gain access to a wireless network using default router identifiers.

5. Change your router’s pre-set password for administration. This is another prudent step.

6. Allow only specific computers to access your wireless network. A router can be configured to only communicate with approved MAC addresses. Of course, MAC addresses can be spoofed, so this is just one element of a security strategy