CYBER SECURITY

LAWS FOR APP SECURITY :

Least Privilege:
Give users and apps only the permissions they need to do their job, no more. This limits what attackers can access.

Defense in Depth:
Use multiple layers of security (like firewalls, encryption, etc.) so if one fails, others can still protect your app.

Fail-Safe Defaults:
Set things to "deny" by default, and only allow access if it's needed and authorized.

Open Design:
Your app's security shouldn't depend on keeping how it works a secret. It should still be secure even if people know its design.

Separation of Duties:
Split important tasks between different people or systems so no one has too much control over everything.

Economy of Mechanism:
Keep security systems simple and easy to check for weaknesses, rather than making them complex.

Security Through Obscurity:
Don't rely only on hiding details of your app to keep it safe. It should be secure even if people know how it works.

Input Validation:
Always check the data coming into your app (like user inputs) to make sure it's safe and not a way to attack your system.

Secure Defaults:
Make sure your app is secure right from the start, with good settings, so users don’t have to fix it later.

Auditability:
Make sure your app keeps logs so you can check what happened if something goes wrong.

Data Encryption:
Protect sensitive data by turning it into unreadable code, so if it gets stolen, it’s useless.

Code Review & Testing:
Regularly check your app’s code for bugs and security holes, and test it to make sure it's safe.

Session Management:
Make sure users' logins are safe by using strong passwords and sessions that expire after a while.

Patch Management:
Keep your app and software up to date with the latest security fixes to protect against new threats.

PENTETING AND FUZZTESTING:

Penetration Testing (Pen Testing):

This is like a "controlled hack" where security experts try to break into your app or system, just like a real hacker would. The goal is to find weaknesses before bad people can exploit them.

Fuzz Testing:

This involves feeding random or unexpected data (called "fuzz") into your app to see if it breaks or behaves unexpectedly. It helps find bugs or security flaws that might happen when the app doesn't handle unusual inputs properly.

TYPES OF PENTESTING:

Black Box Testing:

Definition: In black box testing, the tester has no prior knowledge of the system. They are treated like an external attacker trying to break in from the outside without any inside information.

Goal: To simulate a real-world attack, where the attacker doesn't know the internal workings of the system.

White Box Testing:

Definition: In white box testing, the tester has full knowledge of the system, including access to source code, architecture, and internal systems.

Goal: To find vulnerabilities from an insider's perspective, identifying flaws that may not be obvious from the outside.

Grey Box Testing:

Definition: In grey box testing, the tester has partial knowledge of the system. This could mean having access to some parts of the code or the network but not everything.

Goal: To simulate an attack by someone who has inside knowledge (like an employee) but not full access to everything.

External Penetration Testing:

Definition: This type of test focuses on testing the external-facing parts of a network, such as websites, web apps, and online services that are exposed to the internet.

Goal: To identify vulnerabilities that hackers from outside could exploit to gain unauthorized access to the system.

Internal Penetration Testing:

Definition: This test simulates an attack from within the organization, such as from a malicious employee or someone who has already bypassed physical security.

Goal: To find vulnerabilities that could be exploited if an attacker already has internal access to the network.

Targeted Penetration Testing:

Definition: In this type of testing, both the tester and the organization work together and share information during the test.

Goal: To closely monitor how attacks are conducted and learn more about system vulnerabilities in real-time.

Blind Penetration Testing:

Definition: The tester is given no prior information about the system, and they must rely on their own investigation and tools to discover weaknesses.

Goal: To simulate an attacker with no inside knowledge, testing how vulnerable the system is to an external, unknown threat.

Vulnerability Scanning:

Definition: While not always classified as a full pen test, vulnerability scanning involves using automated tools to check for known vulnerabilities.

Goal: To quickly identify and address weaknesses, though it doesn't always involve the detailed testing of a real pen test.

WHY WE NEED PENTESTIING:

Find Security Holes: Pen testing simulates an attack to discover vulnerabilities (like weak passwords or software flaws) in your system.

Prevent Cyberattacks: By identifying and fixing these weaknesses, pen testing helps prevent real hackers from breaking into your system and stealing data.

Protect Sensitive Information: Pen tests can help secure personal data, financial details, and other sensitive information from being exposed.

Improve Security: Pen testing gives you clear feedback on where your security needs improvement, allowing you to strengthen defenses.

Compliance: Many businesses need pen testing to meet industry standards or regulations (like GDPR or HIPAA), showing they take security seriously.

TOOL FOR PENTESTING:

1. **Kali Linux**

Description: A Linux distribution packed with over 600 security tools, including tools for network scanning, vulnerability assessment, and exploitation.

Used for: All stages of penetration testing, from reconnaissance to exploitation.

2. **Nmap**

Description: A network scanning tool that helps identify active devices, open ports, and services running on a network.

Used for: Network discovery and vulnerability scanning.

3. **Metasploit Framework**

Description: A powerful framework for developing and executing exploit code against a target system. It helps automate attacks and test for vulnerabilities.

Used for: Exploitation, vulnerability testing, and post-exploitation activities.

4. **Burp Suite**

Description: A popular tool for web application security testing, offering features like scanning, interception, and vulnerability discovery.

Used for: Testing and identifying web application vulnerabilities like SQL injection, cross-site scripting (XSS), etc.

5. **Wireshark**

Description: A network protocol analyzer used to capture and analyze network traffic in real-time.

Used for: Packet sniffing and network traffic analysis to find weaknesses in network communication.

6. **Aircrack-ng**

Description: A suite of tools used to crack WEP and WPA-PSK keys in Wi-Fi networks.

Used for: Wireless network security testing, focusing on cracking encryption and breaking into Wi-Fi networks.

7. **John the Ripper**

Description: A password cracking tool used to identify weak passwords by testing various password cracking techniques.

Used for: Cracking password hashes to check the strength of passwords in systems.

8. **Nikto**

Description: A web server scanner that detects potential vulnerabilities such as outdated software, security misconfigurations, and missing security patches.

Used for: Web server scanning and vulnerability discovery.

9. **Hydra**

Description: A fast and flexible brute-force password cracking tool that can target various services like FTP, SSH, and HTTP.

Used for: Brute-forcing passwords on remote services.

10. **Netcat**

Description: A network utility that reads and writes data across network connections using the TCP/IP protocol.

Used for: Creating reverse shells, port scanning, banner grabbing, and testing network connections.

11. **OWASP ZAP (Zed Attack Proxy)**

Description: A tool designed to find security vulnerabilities in web applications. It's open-source and used for automated and manual penetration testing.

Used for: Web application vulnerability scanning and testing.

12. **SQLmap:**

Description: An open-source tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications.

Used for: Automated SQL injection testing and exploitation.

BLOCK CHAIN:

What is Blockchain?

Blockchain is a secure and transparent digital system that stores data in "blocks." These blocks are linked together in a "chain," and once data is added, it cannot be altered. It's commonly used in cryptocurrency, but can be applied to many other uses like contracts and supply chains.

How Does Blockchain Work?

Transaction Initiation: A user starts a transaction (e.g., sending cryptocurrency or transferring data).

Block Creation: The transaction is grouped with other transactions and added to a block.

Validation: The network of computers (called nodes) checks and confirms the transaction is valid through a process (like mining or proof of stake).

Block Addition: Once validated, the block is added to the existing blockchain in a secure and permanent way.

Completion: The transaction is final, and the blockchain remains unchangeable, ensuring no tampering or fraud.

Key Features:

Decentralized: No central authority controls the blockchain; it's managed by many computers.

Immutable: Once a block is added, it cannot be changed or deleted.

Transparent: Anyone in the network can see the data, but it is secure and protected by encryption.

In short, blockchain is a reliable, tamper-proof way to store and transfer data securely across a network.

MALWARE ANALYSIS:

What is Malware Analysis?

Malware analysis is the process of studying malicious software (malware) to understand how it works, what damage it can cause, and how to defend against it. Malware can include viruses, worms, Trojans, ransomware, and spyware.

How is Malware Analysis Performed?

Static Analysis:

The malware is examined without running it. This includes looking at its code, file structure, and other characteristics to see how it behaves.

Tools like disassemblers and decompilers are used to understand the code inside the malware.

Dynamic Analysis:

The malware is executed in a controlled environment (like a sandbox) to see how it behaves when run.

It's monitored for actions like making changes to files, sending data, or infecting other systems.

Behavioral Analysis:

This involves studying what the malware does after it runs, such as what files it creates, changes, or deletes, and what network connections it tries to make.

Reverse Engineering:

Experts disassemble the malware to understand its inner workings. This helps identify how it can be stopped or removed.

Signature-based Detection:

Malware is compared with known patterns or "signatures" in databases to quickly identify it.

Why Perform Malware Analysis?

To understand how malware works and what damage it can do.

To create better defenses (like antivirus software) against malware.

To help organizations recover from an attack and improve security.

Here are some common tools and websites used for **malware analysis**:

Tools for Malware Analysis:

IDA Pro:

A powerful disassembler and debugger used for reverse engineering malware and analyzing its code.

OllyDbg:

A debugger used for dynamic analysis of binary files, helpful in reverse engineering malware.

Wireshark:

A network protocol analyzer used to capture and inspect network traffic generated by malware.

Cuckoo Sandbox:

A popular open-source automated malware analysis system that runs malware in a safe, controlled environment to observe its behavior.

VirusTotal:

A free online tool that scans files with multiple antivirus engines to detect malware.

Procmon (Process Monitor):

A real-time monitoring tool that captures and displays file system, registry, and process/thread activity to analyze malware behavior.

PEiD:

A tool used to detect packers, cryptors, and other protection mechanisms in executable files.

Radare2:

A free open-source tool for reverse engineering, debugging, and analyzing binaries.

x64dbg:

A Windows debugger used to analyze and debug 32-bit and 64-bit applications, including malware.

Websites for Malware Analysis:

VirusTotal (www.virustotal.com)

An online service that scans files and URLs with over 70 antivirus engines to detect malware.

Hybrid Analysis (www.hybrid-analysis.com)

A free malware analysis tool that provides detailed reports on files in a sandbox environment.

Any.Run (www.any.run)

An interactive online malware sandbox that lets you analyze and visualize malware behavior in real-time.

MalwareBazaar (www.malwarebazaar.org)

A repository of malware samples where security researchers can share and download malware for analysis.

Joe Sandbox (www.joesandbox.com)

A platform for analyzing files, URLs, and network traffic to identify potential malware.

ThreatMiner (www.threatminer.org)

A tool for searching and analyzing malware-related information, including indicators of compromise (IOCs).

Cryptography:

The practice of protecting information by transforming it into a secure format that only authorized people can read.

Encryption

The process of converting plain data into a secret code to prevent unauthorized access.

Decryption

The process of converting encrypted data back into its original format, so it can be read by authorized users.

Hashing

A method of turning data (like a password) into a fixed-length string of characters. It's used for verifying data integrity, not for retrieval.

Quantum Cryptography

A type of encryption that uses the principles of quantum mechanics to make data transmission extremely secure, making it nearly impossible to hack.

Symmetric Cryptography

A type of encryption where the same key is used for both encryption and decryption.

Asymmetric Cryptography

A type of encryption where two different keys are used: one for encryption (public key) and another for decryption (private key).