UNIT IV SECURITY TESTING

4.1 Traditional Software Testing:

Traditional software testing is a systematic process used to ensure that software products meet specific requirements and function as expected before being released to users. It involves verifying that the software is free from defects, behaves as designed, and performs efficiently under various conditions. Below is a detailed explanation of its key aspects:

1. Types of Testing

Traditional software testing is often categorized into two main types:

- **Manual Testing**: Testers manually execute test cases without using any automation tools. It includes techniques like:
  - **Exploratory Testing**: Testers explore the software without predefined test cases, using their knowledge and experience to identify potential bugs.
  - **Ad-Hoc Testing**: Informal testing where testers use their intuition to find defects without following a structured approach.
  - **Test Case Execution**: Based on pre-written test cases, testers check if the software behaves as expected for each scenario.
- **Automated Testing**: Test cases are executed using tools and scripts, improving efficiency for repetitive or time-consuming tests. This is typically faster but requires initial setup and is suitable for regression testing, performance testing, etc.

2. Phases in Traditional Software Testing

The testing process is generally divided into the following phases:

- **Requirement Analysis**: The first step involves understanding the software requirements and ensuring the testing team knows the functionality that needs to be tested.
- **Test Planning**: In this phase, a detailed test plan is created, outlining the scope of testing, objectives, tools required, time estimation, and resources. It also includes identifying what will be tested (test coverage), what will not be tested, and assigning responsibilities to the testing team.
- **Test Design**: Test cases and test scripts are written in this phase. A good test case covers both positive and negative scenarios and should be traceable to the requirements it is designed to validate. Test data, which includes the input values and expected results, is also prepared in this phase.
- **Test Environment Setup**: The testing team sets up the hardware and software environments that are required to run the tests, such as test servers, databases, networks, and tools.

- **Test Execution**: Testers begin executing the test cases based on the plan. They mark test cases as “pass” or “fail” based on the actual versus expected results.
- **Defect Reporting**: If a test case fails, the tester logs a defect. The development team will investigate and fix the issue. After the defect is fixed, it is re-tested in a process called re-testing, and the entire suite might be run again for regression testing.
- **Test Closure**: Once the testing process is complete, a test summary report is generated, which includes details such as how many test cases were run, how many passed/failed, and the number of defects found and fixed.

3. Types of Testing Techniques

Several types of testing techniques can be applied during the traditional testing process, depending on the software’s needs:

- **Unit Testing**: The smallest unit of code, such as a function or method, is tested in isolation. This is usually performed by developers.
- **Integration Testing**: After individual units are tested, they are combined to test their interaction. The aim is to detect issues when different components work together.
- **System Testing**: The entire system is tested as a whole to ensure that it meets the requirements. This includes functional and non-functional tests.
- **Acceptance Testing**: This is the final testing phase before the software is delivered to the customer. It ensures that the software meets business needs and requirements.
- **Regression Testing**: Testing conducted to verify that changes in the software, such as bug fixes or new features, have not negatively affected existing functionalities.
- **Performance Testing**: Evaluates the software’s performance under various conditions, such as load, stress, or scalability. It ensures that the system can handle the expected user load and functions within acceptable performance thresholds.

4. Test Artifacts

Artifacts are the deliverables created during the testing process, which include:

- **Test Plan**: Outlines the testing strategy, objectives, schedule, deliverables, and resources.
- **Test Cases**: Detailed steps that outline how to test specific functionality.
- **Test Scripts**: Automated test scripts written for executing tests.
- **Defect Logs**: A record of defects or bugs found during testing.
- **Test Reports**: Summarizes the testing activities, such as the number of test cases executed, the results, and the number of defects found and fixed.

5. Tools Used in Traditional Testing

Various tools can be used to facilitate testing activities, including:

- **Test Management Tools**: Tools like TestRail or HP ALM help organize test cases, track execution, and report results.
- **Defect Tracking Tools**: JIRA, Bugzilla, or Mantis are used to log and track the status of defects.
- **Automation Tools**: Selenium, QTP, or JUnit are used for automating the execution of test cases.
- **Performance Testing Tools**: Tools like JMeter or LoadRunner are used to assess system performance under different conditions.

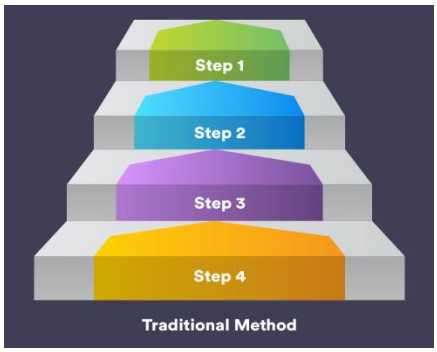
6. Challenges in Traditional Software Testing

- **Time-consuming**: Manual testing, in particular, is time-intensive and requires significant effort.
- **Limited by Human Factors**: Manual testers can make mistakes or miss defects that an automated tool might catch.
- **Repetitive Work**: Regression testing can become cumbersome without automation, especially in larger projects.
- **Scalability Issues**: As the size and complexity of software grow, traditional testing methods may struggle to scale effectively.

4.2 Comparison

Traditional Testing

Traditional testing methodologies have been in existence since the inception of software development. They are primarily based on pre-organized phases/stages of the software testing life cycle. In this case, the testing flow is unidirectional, from testing to maintenance. With time, IT practices have evolved and so have testing approaches, as traditional testing usually fails to address the product's continuous testing needs.



Features Of Traditional Testing

- Performed incrementally.
- The result is only released after all the defects in the software are either resolved or rectified.
- Entirely managed by the project manager.
- Follows a top-down approach where the next phase of testing begins only after completion of the previous stage.
- Predefined steps to execute the process.
- The client's involvement is required only in the initial phase of testing.

Advantages Of Traditional Testing

- It helps in the identification of the maximum number of defects.
- It ensures a quality product.

Disadvantages of Traditional Testing

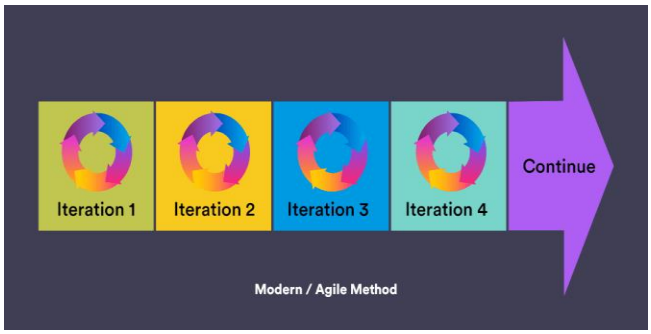
- It is a long-running and taxing process.
- Since the changes are implemented only at the end of testing, product delivery speed is affected.
- The complete set of requirements must be communicated in the initial phase without any chance of modification after the project development has started.
- The approach has minimal to no interactions between different software testers.
- Documentation becomes a high priority in traditional methodology and becomes expensive to create.
- There are minimal chances to implement reusable components.

Traditional testing methodologies are suitable only when the requirements are precise. Although the process is quite useful in identifying defects with the product under test, with the advent of modern or agile testing practices, traditional testing practices have become incompatible.

Modern/Agile Testing

With rapid technological developments and an increasing number of organizations entering into the software testing space, software testers are capable of different testing processes and optimizing these processes at multiple levels of testing by following the modern ways of testing. This modern or agile software testing practice is an iterative and incremental approach. It typically covers all layers and all types of testing. The entire testing team collaborates to find defects in the software while validating its quality, performance, and effectiveness.

In agile testing methodology, both the development and testing tasks are performed collaboratively while ensuring an exclusive tester for testing purposes.



Vital Attributes Of Agile Testing

Continuous interaction with developers

The agile or modern testing approach ensures that the testing and the development processes are closely linked. Testers work as a part of the development team and report on quality issues that can affect end-users, and suggest solutions.

Robust communication with product owners

In this testing methodology, testers continuously interact with product owners to establish project expectations to help software developers align with the overall product roadmap and fulfill customer needs.

Team collaboration in quality assurance

Agile testing promotes team collaboration in maintaining QA. Developers are an equal part of building unit test cases for a superior testing process and enhancing audits' overall quality. Further, developers also follow the recommendations of software testers for various test requirements and code improvements.

Features Of Modern Testing

- Less time-consuming and requires minimum documentation
- Follows an iterative model that is flexible to changes in requirements
- Can be performed using automated tools
- The approach ensures collaboration with end-users

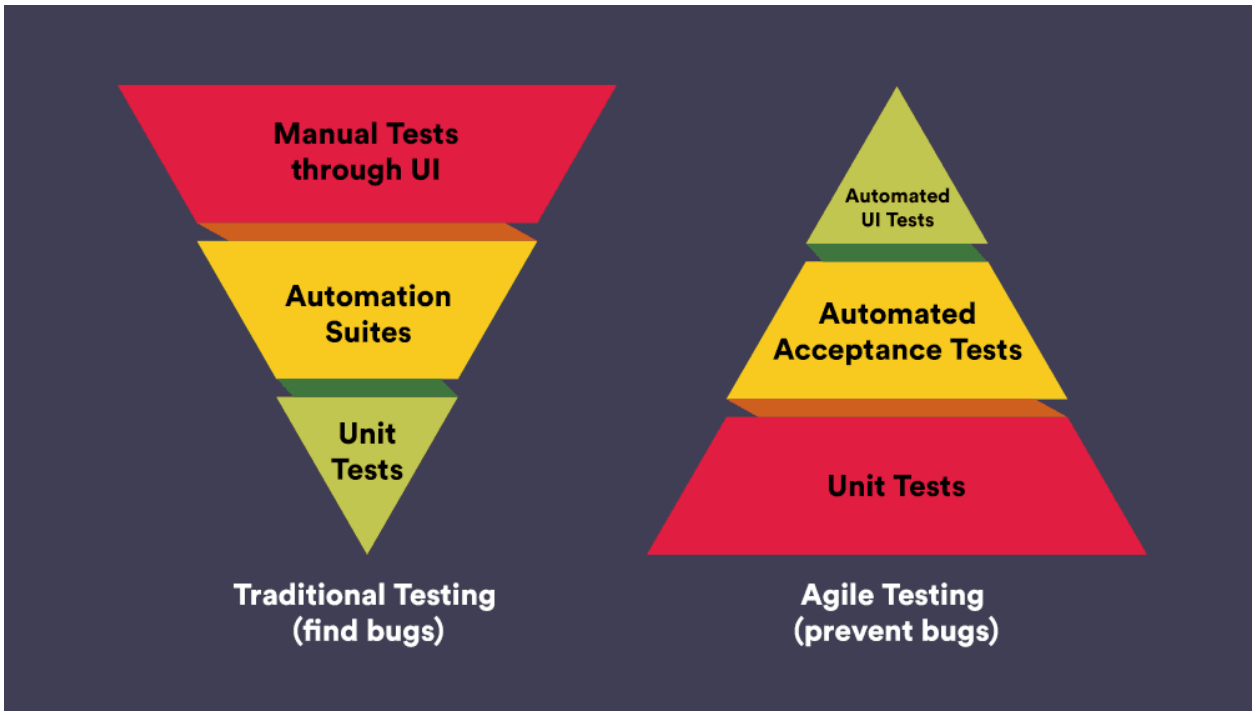
Advantages Of Modern Testing

- Modern or agile testing offers efficient risk management
- Promotes feature driven development and face-to-face interactions

- Includes rigorous planning, analysis, and testing
- Ensures rapid product delivery while ensuring optimum quality

Disadvantages Of Modern Testing

- Difficult to assess the amount of effort required for a particular test
- With limited documentation, it makes it difficult sometimes to specify and communicate individual testing components of large projects



Aspect	Traditional Testing	Agile/Modern Testing
Development Methodology	Sequential (Waterfall model), testing occurs after development	Iterative (Agile, Scrum), testing is integrated with development
Test Planning	Extensive upfront planning with fixed test cases	Adaptive, lightweight planning with evolving test cases
Testing Approach	Manual-heavy, automation used for regression	Automation-centric, tests integrated with CI/CD pipeline
Test Focus	Testing at the end, focusing on the entire product	Continuous testing for each feature in every sprint

Defect Detection	Late defect detection, bugs found at the end of the project	Early defect detection, bugs fixed continuously during development
Collaboration	Testers work independently from developers	High collaboration between developers, testers, and product owners
Documentation	Heavy documentation with detailed test cases	Minimal documentation, test cases evolve as per project needs
Regression Testing	Extensive at the end, since all features are built first	Less regression, testing happens continuously across sprints
Customer Involvement	Customer is involved only at the beginning and end	Customer or product owner provides feedback after each sprint
Risk Management	High risk at the end due to late testing and bug discovery	Low risk, continuous testing and early issue resolution
Tools and Automation	Non-integrated tools, automation used mostly for regression	Automation deeply integrated with tools like Jenkins, Selenium, etc.
Testers' Role	QA team works separately, tests after development	Testers collaborate throughout the process, testing continuously
Flexibility to Changes	Difficult to accommodate changes, scope is fixed early	Highly flexible, can adapt to changing requirements during sprints
Cost of Fixing Defects	Higher, as defects are found late in the development process	Lower, since defects are identified and fixed early in each sprint
End-User Feedback	Feedback comes late, often after product delivery	Regular feedback through iterations, ensuring product aligns with user needs

4.3 Secure Software Development Life Cycle

The Security System Development Life Cycle (SSDLC) is a framework used to manage the development, maintenance, and retirement of an organization's information security systems. The SSDLC is a cyclical process that includes the following phases:

1. **Planning**: During this phase, the organization identifies its information security needs and develops a plan to meet those needs. This may include identifying potential security risks and vulnerabilities, and determining the appropriate controls to mitigate those risks.

2. **Analysis**: During this phase, the organization analyzes its information security needs in more detail and develops a detailed security requirements specification.

3. **Design**: During this phase, the organization designs the security system to meet the requirements developed in the previous phase. This may include selecting and configuring security controls, such as firewalls, intrusion detection systems, and encryption.

4. **Implementation**: During this phase, the organization develops, tests, and deploys the security system.

5. **Maintenance**: After the security system has been deployed, it enters the maintenance phase, where it is updated, maintained, and tweaked to meet the changing needs of the organization.

6. **Retirement**: Eventually, the security system will reach the end of its useful life and will need to be retired. During this phase, the organization will plan for the replacement of the system, and ensure that data stored in it is properly preserved.

The SSDLC is a useful framework for managing the development, maintenance, and retirement of an organization's information security systems. It helps to ensure that security systems meet the needs of the organization and are developed in a structured and controlled manner. This can help organizations to protect their sensitive information, maintain compliance with relevant regulations, and keep their data and systems safe from cyber threats.

Security System Development Life Cycle (SecSDLC) is defined as the set of procedures that are executed in a sequence in the software development cycle (SDLC). It is designed such that it can help developers to create software and applications in a way that reduces the security risks at later stages significantly from the start. The Security System Development Life Cycle (SecSDLC) is similar to Software Development Life Cycle (SDLC), but they differ in terms of the activities that are carried out in each phase of the cycle. SecSDLC eliminates security vulnerabilities. Its process involves identification of certain threats and the risks they impose on a system as well as the needed implementation of security controls to counter, remove and manage the risks involved. Whereas, in the SDLC process, the focus is mainly on the designs and implementations of an information system.

Phases involved in SecSDLC are:

- **System Investigation:** This process is started by the officials/directives working at the top level management in the organization. The objectives and goals of the project are considered priorly in order to execute this process. An Information Security Policy is defined which contains the

descriptions of security applications and programs installed along with their implementations in organization's system.

- **System Analysis:** In this phase, detailed document analysis of the documents from the System Investigation phase are done. Already existing security policies, applications and software are analyzed in order to check for different flaws and vulnerabilities in the system. Upcoming threat possibilities are also analyzed. Risk management comes under this process only.

- **Logical Design:** The Logical Design phase deals with the development of tools and following blueprints that are involved in various information security policies, their applications and software. Backup and recovery policies are also drafted in order to prevent future losses. In case of any disaster, the steps to take in business are also planned. The decision to outsource the company project is decided in this phase. It is analyzed whether the project can be completed in the company itself or it needs to be sent to another company for the specific task.

- **Physical Design:** The technical teams acquire the tools and blueprints needed for the implementation of the software and application of the system security. During this phase, different solutions are investigated for any unforeseen issues which may be encountered in the future. They are analyzed and written down in order to cover most of the vulnerabilities that were missed during the analysis phase.
- **Implementation:** The solution decided in earlier phases is made final whether the project is in-house or outsourced. The proper documentation is provided of the product in order to meet the requirements specified for the project to be met. Implementation and integration process of the project are carried out with the help of various teams aggressively testing whether the product meets the system requirements specified in the system documentation.
- **Maintenance:** After the implementation of the security program it must be ensured that it is functioning properly and is managed accordingly. The security program must be kept up to date accordingly in order to counter new threats that can be left unseen at the time of design.

ADVANTAGES OR DISADVANTAGES:

Advantages of using the Security System Development Life Cycle (SSDLC) framework include:

1. Improved security: By following the SSDLC, organizations can ensure that their information security systems are developed, maintained and retired in a controlled and structured manner, which can help to improve overall security.
2. Compliance: The SSDLC can help organizations to meet compliance requirements, by ensuring that security controls are implemented to meet relevant regulations.
3. Risk management: The SSDLC provides a structured and controlled approach to managing information security risks, which can help to identify and mitigate potential risks.
4. Better project management: The SSDLC provides a structured and controlled approach to managing information security projects, which can help to improve project management and reduce risks.
5. Increased efficiency: By following the SSDLC, organizations can ensure that their resources are used efficiently, by ensuring that the development, maintenance and retirement of information security systems is planned and managed in a consistent and controlled manner.

Disadvantages of using the SSDLC framework include:

1. Cost: Implementing the SSDLC framework can be costly, as it may require additional resources, such as security experts, to manage the process.
2. Time-consuming: The SSDLC is a cyclical process that involves multiple phases, which can be time-consuming to implement.
3. Complexity: The SSDLC process can be complex, especially for organizations that have not previously used this framework.
4. Inflexibility: The SSDLC is a structured process, which can make it difficult for organizations to respond quickly to changing security needs.
5. Limited Adaptability: The SSDLC is a predefined process, which is not adaptable to new technologies, it may require updating or revising to accommodate new technology.

4.4 Risk Based Security Testing

Risk-based testing is a software testing approach that prioritizes testing efforts and resources based on different software application components' potential risks and impacts. The goal of risk-based quality assurance testing is to identify the critical areas of the software that require the most attention and to allocate testing resources accordingly.The process of risk-based testing involves analyzing the software application to identify potential risks. For instance: defects, failures, security breaches, or other types of issues that could negatively impact the end users or the business. Once risks are identified, they are evaluated based on their likelihood of occurring and their potential impact on the system.

Based on this analysis, testing efforts for businesses in the United Kingdom should be focused on the areas with the highest risk. Critical functionalities and features are tested more thoroughly than less important or lower-risk features. This approach allows testing teams to prioritize their efforts to focus on the areas that are most likely to cause problems or issues. The primary goal of risk-based testing is to identify and mitigate the most significant risks that could affect the quality, reliability, or usability of the software. By focused testing efforts on the areas that are most likely to pose risks, risk-based testing can help ensure that the software meets the requirements and expectations of stakeholders and reduces the likelihood of defects or issues that could impact the user experience or the business value of the application.The key outcome of risk-based testing is to ensure that software is of high quality and meets the needs and expectations of stakeholders by identifying and mitigating the most significant risks that could impact the success of the application.

Risk-based testing focuses on identifying and prioritizing testing activities based on the level of risk associated with various features or components of the software system under test. Here are some ways in which risk-based testing differs from other testing approaches:

Test coverage:

The focus is on identifying the most critical areas of the software system that require the most attention and coverage, while other testing approaches may aim for a broader range across the system.

Prioritization:

It prioritizes testing activities based on the level of risk associated with the various features or components of the software system, while other testing approaches may prioritize based on functional requirements or other factors.

Risk assessment:

The risk assessment is an ongoing process that helps identify, assess and mitigate risks throughout the software development lifecycle. At the same time, other testing approaches may not emphasize risk assessment as much.

Test planning:

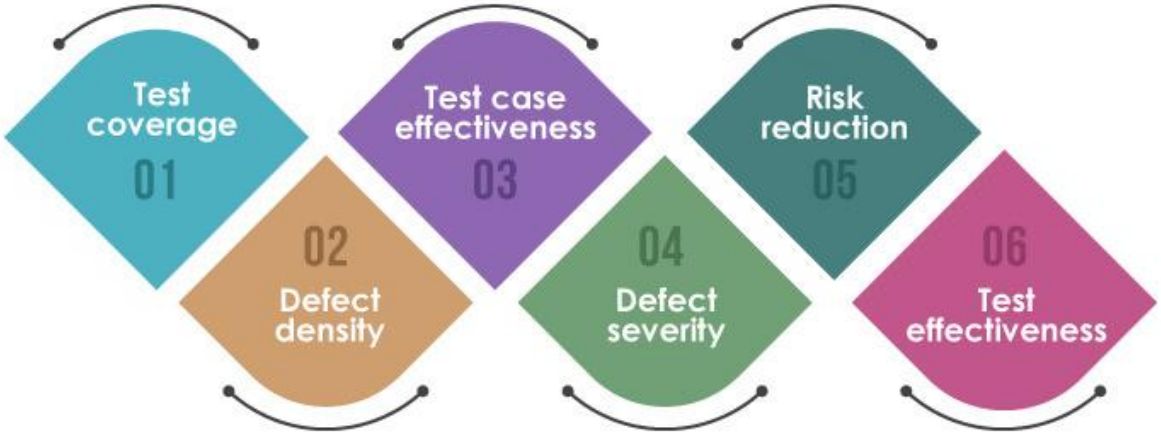
It requires careful planning to identify the most critical areas of the software system and prioritize testing activities based on risk. Other testing approaches may not require such detailed planning.

Test execution:

It focuses on executing tests in areas of the system identified as high-risk, while other testing approaches may run tests based on functional requirements or other criteria.

Risk-Based Testing aims to optimize the testing effort by focusing on areas of the software system that pose the highest risks to quality and functionality. This approach helps ensure that testing efforts are aligned with business objectives and can help reduce the overall testing effort by focusing on areas that are most critical.

Risk-Based Testing Report and Metrics



The main objective of risk-based testing is to identify and mitigate the high-risk areas in the application. The following are some of the results and metrics that can be used to evaluate the effectiveness of risk-based testing:

Test coverage:

Test coverage is the extent to which the software under test has been tested. In risk-based software testing, the focus is on high-risk areas, so the test coverage should reflect this. A higher percentage of test coverage in high-risk areas indicates that the testing effort has been targeted effectively.

Defect density:

Defect density is the number of defects found in a specific area of the software. In risk-based testing, high-risk areas should have a higher defect density. If the defect density in high-risk areas is low, it may indicate that the testing effort was not focused enough on these areas.

Test case effectiveness:

Test case effectiveness measures how well the test cases cover the requirements and identify defects. In risk-based approach to testing, test case effectiveness should be high for high-risk areas.

Defect severity:

Defect severity measures the impact of a defect on the software. In risk-based testing, high-risk areas should have a higher severity level for defects found.

Risk reduction:

Risk reduction measures the effectiveness of the risk-based testing approach. It compares the number of high-risk areas before testing to the number of high-risk areas after testing. The higher the risk reduction, the more effective the testing approach.

Test effectiveness:

Test effectiveness measures the overall effectiveness of the testing effort. In risk-based testing, the test effectiveness should be high for high-risk areas.

Test Report Preparation

Identify the purpose and audience of the report:

Before starting the report, identify the intended purpose and audience. Determine what information needs to be included, and how the report will be used.

Define the scope of the testing:

Specify the scope of the testing that was performed, including the type of testing (risk-based functional testing, risk-based regression testing, risk-based performance testing, risk-based security testing), the testing environment, and the systems or applications that were tested.

Describe the testing process:

Describe the testing process that was used, including the testing methodology, test plan, and test cases. Include any issues that were encountered during the testing process.

Present the results:

Present the results of the testing in a clear and concise manner. Use tables, charts, and graphs to help visualize the data. Include any defects that were found during testing, along with their severity and priority.

Provide recommendations:

Provide recommendations for addressing the defects that were found during testing. These recommendations may include suggested changes to the code, updates to the test plan, or additional testing that may be required.

Conclude the report:

Conclude the report by summarizing the key findings and recommendations. Include any lessons learned during the testing process and highlight any areas that may require additional attention in future testing efforts.

4.5 Prioritizing Security Testing With Threat Modeling

Threat Modeling

Threat modeling is a methodical process that aims to pinpoint and evaluate possible risks and weaknesses in a system or application. This process allows organizations to comprehend the various dangers that could threaten their digital assets, enabling them to prioritize their security measures optimally. The fundamental objective of threat modeling is to establish a robust security plan that diminishes the chances of successful cyber attacks and lessens their impact.

Threat modeling offers numerous benefits to organizations, some of which include the following:

- Proactive Security: By identifying potential threats and vulnerabilities early in the development process, organizations can take proactive measures to address these issues and reduce the risk of security incidents.
- Efficient Resource Allocation: Threat modeling helps organizations prioritize their security efforts by focusing on the most critical vulnerabilities and threats. This ensures that limited resources are allocated effectively and efficiently.
- Improved Incident Response: Understanding the potential attack vectors and having a comprehensive threat model allows organizations to respond more effectively and quickly to security incidents.
- Compliance and Regulation: Many industries have specific security regulations and requirements that must be met. Threat modeling can help organizations demonstrate compliance with these standards by providing a clear understanding of the security measures in place.

Enhanced Communication: A well-structured threat model can serve as a communication tool between various stakeholders, such as developers, security teams, and management. This ensures everyone is on the same page regarding security priorities and efforts.

Steps in Threat Modeling

The threat modeling process generally follows these key steps:

1. Define the Scope and Objectives

- **Identify assets**: Determine what valuable assets need protection, such as user data, system credentials, intellectual property, etc.
- **Understand system architecture**: Create a high-level diagram of the system, including data flows, entry points, external dependencies, and user roles.

- **Set security goals**: Clarify the security objectives (e.g., protecting data confidentiality, ensuring system availability, or maintaining data integrity).

2. Identify Threats

- Use frameworks like **STRIDE** (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) to identify potential threats. These threats can be categorized as:
    - **Spoofing**: Pretending to be another user or entity.
    - **Tampering**: Unauthorized modification of data or code.
    - **Repudiation**: Denying an action without a way to prove it happened.
    - **Information Disclosure**: Unauthorized access to sensitive data.
    - **Denial of Service**: Making a service unavailable to its intended users.
    - **Elevation of Privilege**: Gaining higher access rights than permitted.
- Other threat modeling frameworks like **DREAD** (Damage potential, Reproducibility, Exploitability, Affected users, Discoverability) or **MITRE ATT&CK** may also be used.

3. Assess and Prioritize Threats

- **Risk Analysis**: Assess each identified threat based on:
    - **Likelihood**: How likely is it that the threat will occur?
    - **Impact**: If the threat occurs, what is the potential damage or loss (financial, reputational, legal, etc.)?
- **Risk Matrix**: Create a risk matrix where threats are plotted based on their likelihood and impact. This helps in visualizing which threats should be given priority. High-likelihood, high-impact threats are considered critical and need to be addressed first.

4. Mitigate and Address Threats

- **Determine mitigation strategies**: Once the threats are prioritized, determine how to mitigate them. This could involve:
    - **Security controls**: Implementing firewalls, encryption, authentication mechanisms, etc.
    - **Code changes**: Refactoring vulnerable code sections.
    - **Security policies**: Strengthening access control and data handling policies.
- **Develop test cases**: For each mitigated threat, create specific security test cases that simulate potential attack scenarios.

3. Incorporating Threat Modeling into Security Testing

Once threats have been identified and prioritized using threat modeling, you can guide your security testing efforts more effectively. The key benefits of this integration are:

1. Focused Testing on High-Risk Areas

- By prioritizing the most critical threats (those with high likelihood and impact), teams can focus their testing resources on the most important areas of the system, ensuring that potential security vulnerabilities are mitigated effectively.

2. Developing Targeted Test Cases

- **Threat-based test cases**: Develop test cases directly from the identified threats. For example:
    - If **information disclosure** is a high-priority risk, create test cases for testing whether sensitive data is properly encrypted and transmitted securely.
    - If **elevation of privilege** is identified, develop tests to ensure proper access controls are enforced.

3. Dynamic and Static Testing

- Apply both **static analysis** (reviewing code for security flaws) and **dynamic testing** (simulating attacks in a running system) based on the threat model:
    - **Static Analysis**: Check for security vulnerabilities in the code, such as injection flaws or insecure configurations.
    - **Dynamic Testing**: Simulate real-world attacks like SQL injection, Cross-Site Scripting (XSS), or Denial of Service (DoS) to test for runtime vulnerabilities.

4. Automated Security Testing

- Once prioritized threats are identified, automate security tests (especially for high-risk threats) within your Continuous Integration/Continuous Delivery (CI/CD) pipeline. For example:
    - Automated scans for vulnerabilities like **SQL injection** or **cross-site scripting (XSS)**.
    - **Penetration testing** tools that simulate attacks on prioritized components.

5. Regular Reassessment

- Threat modeling is not a one-time process; it should be reassessed regularly as the system evolves. New features, architecture changes, or newly discovered vulnerabilities in external libraries can introduce new risks that need to be addressed.

4. Tools for Threat Modeling and Security Testing

Several tools can help in automating or facilitating threat modeling and security testing:

Threat Modeling Tools:

- **Microsoft Threat Modeling Tool**: A popular tool that provides templates and automation to guide the creation of threat models based on system architecture diagrams.
- **OWASP Threat Dragon**: An open-source threat modeling tool that provides diagramming and risk assessment functionalities.

Security Testing Tools:

- **Static Application Security Testing (SAST)**: Tools like **SonarQube**, **Checkmarx**, and **Veracode** are used to analyze code for security vulnerabilities without running it.
- **Dynamic Application Security Testing (DAST)**: Tools like **Burp Suite**, **OWASP ZAP**, and **Acunetix** help simulate real-world attacks on a running application.
- **Interactive Application Security Testing (IAST)**: Tools like **Contrast Security** and **Seeker** offer hybrid approaches, combining both static and dynamic analysis while the application is running.

5. Example: Threat Modeling in Action

Suppose a web application processes sensitive financial data and you perform a threat modeling exercise. Some high-priority threats identified might include:

- **Data tampering**: Attackers might manipulate financial data during transmission.
    - **Security Tests**: Perform **man-in-the-middle (MITM)** attack simulations to ensure data integrity is maintained during transmission using secure transport protocols like HTTPS/TLS.
- **SQL injection**: An attacker could exploit poorly written SQL queries to manipulate databases.
    - **Security Tests**: Implement test cases for SQL injection vulnerabilities using tools like **OWASP ZAP** or automated scanners like **SQLMap**.
- **Cross-Site Scripting (XSS)**: Malicious scripts might be injected into web pages, compromising user data.
    - **Security Tests**: Test for XSS vulnerabilities by simulating attacks where user input is injected into web pages without proper validation or sanitization.

---

6. Benefits of Prioritizing Security Testing with Threat Modeling

1. **Focused Efforts**: Security teams can prioritize high-risk threats and ensure critical vulnerabilities are addressed first, optimizing the use of time and resources.

2. **Proactive Risk Mitigation**: By identifying threats early, teams can fix vulnerabilities before attackers exploit them, leading to improved system security and resilience.

3. **Improved Collaboration**: Threat modeling encourages collaboration between developers, testers, and security experts, creating a shared understanding of security risks and a more integrated approach to security testing.

4. **Continuous Improvement**: As threat modeling is an iterative process, it supports continuous reassessment of security risks, ensuring that testing evolves alongside the software.

4.6 Penetration Testing

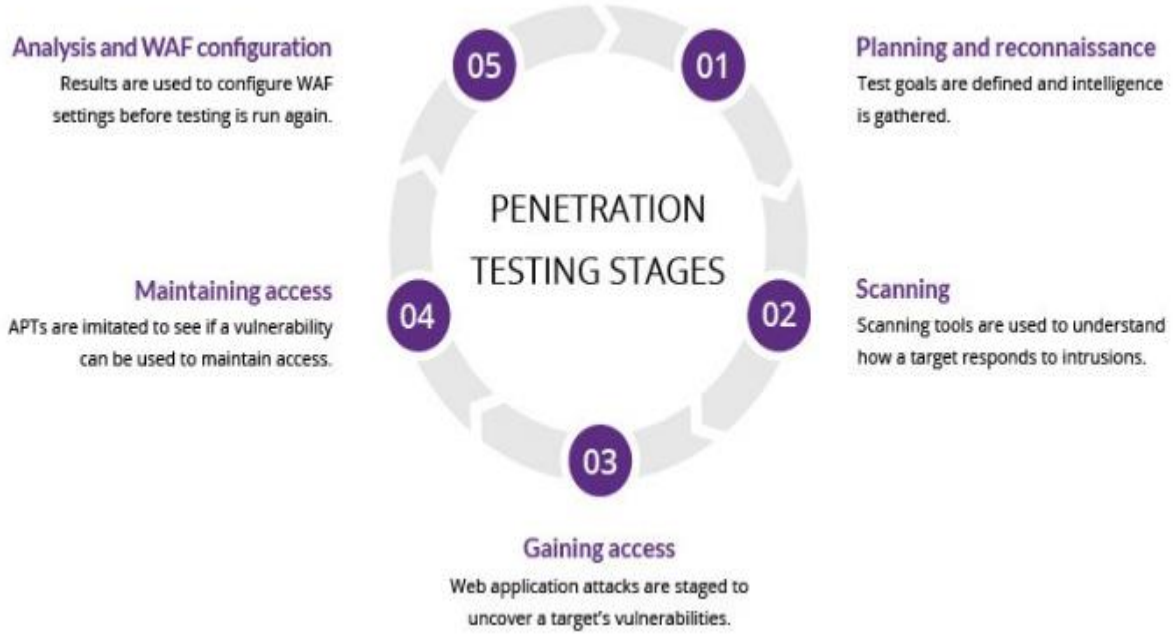
A penetration test, also known as a pen test, is a simulated cyber attack against your computer system to check for exploitable vulnerabilities. In the context of web application security, penetration testing is commonly used to augment a web application firewall (WAF).

Pen testing can involve the attempted breaching of any number of application systems, (e.g., application protocol interfaces (APIs), frontend/backend servers) to uncover vulnerabilities, such as unsanitized inputs that are susceptible to code injection attacks.

Insights provided by the penetration test can be used to fine-tune your WAF security policies and patch detected vulnerabilities.

Penetration testing stages

The pen testing process can be broken down into five stages.



1. Planning and reconnaissance
The first stage involves:

- Defining the scope and goals of a test, including the systems to be addressed and the testing methods to be used.
- Gathering intelligence (e.g., network and domain names, mail server) to better understand how a target works and its potential vulnerabilities.

2. Scanning
The next step is to understand how the target application will respond to various intrusion attempts. This is typically done using:

- **Static analysis** – Inspecting an application's code to estimate the way it behaves while running. These tools can scan the entirety of the code in a single pass.
- **Dynamic analysis** – Inspecting an application's code in a running state. This is a more practical way of scanning, as it provides a real-time view into an application's performance.

3. Gaining Access
This stage uses web application attacks, such as cross-site scripting, SQL injection and backdoors, to uncover a target's vulnerabilities. Testers then try and exploit these vulnerabilities, typically by escalating privileges, stealing data, intercepting traffic, etc., to understand the damage they can cause.

4. Maintaining access
The goal of this stage is to see if the vulnerability can be used to achieve a persistent presence in the exploited system— long enough for a bad actor to gain in-depth access. The idea is to imitate advanced persistent threats, which often remain in a system for months in order to steal an organization's most sensitive data.

5. Analysis
The results of the penetration test are then compiled into a report detailing:

- Specific vulnerabilities that were exploited
- Sensitive data that was accessed
- The amount of time the pen tester was able to remain in the system undetected

This information is analyzed by security personnel to help configure an enterprise's WAF settings and other application security solutions to patch vulnerabilities and protect against future attacks.

Penetration testing methods

External testing

External penetration tests target the assets of a company that are visible on the internet, e.g., the web application itself, the company website, and email and domain name servers (DNS). The goal is to gain access and extract valuable data.

Internal testing

In an internal test, a tester with access to an application behind its firewall simulates an attack by a malicious insider. This isn't necessarily simulating a rogue employee. A common starting scenario can be an employee whose credentials were stolen due to a phishing attack.

Blind testing

In a blind test, a tester is only given the name of the enterprise that's being targeted. This gives security personnel a real-time look into how an actual application assault would take place.

Double-blind testing

In a double blind test, security personnel have no prior knowledge of the simulated attack. As in the real world, they won't have any time to shore up their defenses before an attempted breach.

Targeted testing

In this scenario, both the tester and security personnel work together and keep each other appraised of their movements. This is a valuable training exercise that provides a security team with real-time feedback from a hacker's point of view.

4.7 Penetration Testing: Detailed Overview of Planning and Scoping

Penetration testing (or "pen testing") is a method used to evaluate the security of an organization's IT infrastructure by simulating real-world attacks. The planning and scoping phases of a penetration test are critical because they set the stage for how the test will be conducted, what assets will be tested, and how the results will be evaluated. These phases ensure that both the tester and the client organization have a shared understanding of the goals, limitations, risks, and boundaries of the test.

1. Planning Phase

The planning phase involves defining the goals and strategy for the penetration test. It's a collaborative process where the penetration testing team and the client (organization requesting the test) come together to outline how the test will be executed.

Key Steps in Planning:

a) Define Objectives and Purpose of the Test:

- Business Goals: The first step is identifying what the organization hopes to achieve with the penetration test. Is it to assess overall security, ensure compliance with regulatory requirements (like PCI-DSS or HIPAA), evaluate security after recent upgrades, or validate defenses against specific attack vectors?

- Specific Focus Areas: Are there specific areas of concern? For example, is the organization worried about the security of web applications, internal networks, wireless networks, or cloud infrastructure?
- Risk Tolerance: Understanding how much risk the organization is willing to accept (e.g., how much disruption to systems is permissible during the test).

b) Identify the Type of Penetration Test:

- Black-Box Testing: The tester has no prior knowledge of the target system. This approach mimics an external attacker who has no insider information.
- White-Box Testing: The tester has full knowledge of the systems being tested, including network diagrams, IP addresses, credentials, and configurations. This approach helps simulate an internal attack or an attacker who has breached certain defenses.
- Gray-Box Testing: A hybrid approach where the tester has limited information about the target system. It's used to simulate an attack from someone who has partial insider knowledge, such as a former employee or a contractor**.**

c) Define Rules of Engagement (RoE):

The RoE outlines the parameters under which the penetration test will be conducted. It ensures that the test is controlled, minimizes disruption to the business, and aligns with the organization's legal and operational requirements. Key points include:

- Timing and Duration: When will the test take place? Testing should typically be done during non-peak hours to minimize business disruption.
- No-Go Areas: Identify critical systems that cannot be tested due to their importance (e.g., production systems or life-critical environments).
- Test Limitations: Some tests, such as Denial of Service (DoS) attacks, could crash systems or disrupt services. The organization may want to exclude such tests.
- Handling of Sensitive Data: How will sensitive data encountered during testing be handled? Testers may discover personally identifiable information (PII), confidential business data, or even intellectual property.
- Emergency Protocol: What steps should be taken if the tester finds a serious vulnerability that requires immediate attention?

d) Determine Success Criteria:

Success criteria help define how the test's effectiveness will be measured. They are usually related to:

- The number or percentage of vulnerabilities found.
- The types of vulnerabilities found (e.g., critical, high, medium, low).

- The organization’s ability to detect and respond to the simulated attacks.

e) Establish Communication Protocols:

- Points of Contact: Define who from the organization and the pen testing team will serve as primary points of contact during the test.
- Update Frequency: Agree on how often the organization will be updated on the test’s progress (e.g., daily summaries, immediate notifications for critical findings).
- Emergency Contact: Establish procedures for how to escalate the issue if a severe vulnerability is found that could put the organization at risk.

---

2. Scoping Phase

The scoping phase defines the boundaries and limitations of the penetration test. It involves determining what will be tested, how deeply it will be tested, and what is out of scope. Proper scoping is essential to avoid unnecessary risks or unintended disruptions and to ensure the pen test remains aligned with the business’s needs.

Key Elements of Scoping:

a) Asset Inventory and Identification:

- The organization provides a detailed list of the systems, applications, and network components they want to include in the test. This may include:
    - IP addresses and network segments.
    - Specific web applications or cloud infrastructure.
    - External-facing systems (e.g., public websites) or internal systems (e.g., HR databases).
- This inventory ensures that the penetration tester knows exactly what assets to focus on and prevents testing on unintended targets.

b) Scope of the Penetration Test:

- Internal vs. External Testing:
    - External Testing: Focuses on publicly accessible systems, like websites or external IPs.
    - Internal Testing: Simulates an attack from within the organization's network, assessing internal network security and lateral movement capabilities.
- Network Testing: The pen tester may focus on firewalls, routers, switches, wireless networks, and VPNs.

- Application Testing: The test may cover web applications, mobile apps, APIs, and databases.
- Cloud and Virtual Environments: If the organization relies on cloud services (e.g., AWS, Azure, GCP), these services can be part of the scope.
- Social Engineering Testing: Some organizations may include phishing campaigns, physical security breaches, or other social engineering techniques in the scope.

c) Prioritize Critical Assets:

- Not all systems are created equal. Systems that handle sensitive information (e.g., customer data, financial transactions) or are critical to business operations (e.g., e-commerce platforms, customer portals) should be prioritized.
- Assets critical to business continuity should be safeguarded from testing that could lead to outages.

d) Resource Allocation:

- Tools and Technologies: Determine the types of tools that the penetration testing team will use. These may include vulnerability scanners, manual exploitation tools, and network analyzers.
- Personnel: Define the required personnel from both the penetration testing team and the organization. This may include system administrators, network engineers, security personnel, or third-party providers (especially for cloud services).
- Time Commitment: Penetration tests typically run for a predefined period (e.g., two weeks or one month). This ensures that the test is conducted thoroughly but within a reasonable timeframe.

e) Risk Assessment:

- Potential Business Disruptions: Certain tests (like testing for denial-of-service vulnerabilities) could disrupt operations. It's essential to evaluate and accept the risks associated with these tests.
- Mitigation Plans: The organization should have a rollback plan in case something goes wrong (e.g., a system outage). This includes backups, restoration procedures, and any agreed downtime.

f) Legal and Compliance Considerations:

- Legal Authorization: Ensure that the penetration testing team has legal authorization to test the systems (commonly formalized through an authorization letter). Unauthorized testing, even with good intentions, could lead to legal action.
- Compliance Requirements: Some industries have strict regulations around penetration testing (e.g., finance, healthcare). Compliance with frameworks like PCI-DSS, HIPAA, GDPR, or ISO 27001 may dictate the scope and method of testing.

- Third-Party Systems: If third-party systems are involved, ensure that proper permissions are obtained from those vendors. For example, cloud service providers may have specific rules for penetration testing.

---

3. Key Deliverables:

At the end of the planning and scoping phases, the following deliverables are typically produced:

a) Penetration Testing Plan:

- A comprehensive document outlining the agreed-upon goals, scope, methodology, tools, timeline, and resources needed for the test.

b) Rules of Engagement (RoE):

- A formal agreement between the penetration tester and the organization, detailing how the test will be conducted, limitations, and what to do in case of critical findings.

c) Authorization Letter:

- A signed document authorizing the penetration testing team to perform the tests on the agreed-upon assets. This protects both the organization and the testing team from legal liabilities.

PENETRATION TOOLS

Penetration testing requires a variety of tools to simulate real-world attacks and identify security weaknesses in systems, networks, and applications. These tools can help in the discovery, exploitation, and reporting of vulnerabilities. Below are some of the most commonly used tools across different categories of penetration testing:

1. Network Scanning and Discovery Tools

These tools help identify hosts, services, and potential attack surfaces in a network.

- **Nmap (Network Mapper)**:
  - A powerful open-source network scanner used for network discovery and security auditing.
  - Can be used to detect open ports, running services, operating system versions, and network configurations.
- **Angry IP Scanner**:
  - A fast, lightweight, cross-platform IP address and port scanner.
  - Helps in discovering live hosts and services on a network.
- **Netcat**:
  - Known as the “Swiss army knife” of networking, it can be used for port scanning, banner grabbing, and network tunneling.
- **Fping**:

    - A high-performance ping utility for network discovery, allowing bulk pinging of IP addresses.

---

2. Vulnerability Scanning Tools

These tools help identify known vulnerabilities in systems, networks, or applications.

- **Nessus**:
    - A widely-used commercial vulnerability scanner for detecting known vulnerabilities, misconfigurations, and policy violations across networks and applications.
    - Useful for both compliance auditing and vulnerability assessments.
- **OpenVAS** (Open Vulnerability Assessment Scanner):
    - An open-source vulnerability scanner that can detect a wide range of vulnerabilities across network services and applications.
    - Part of the Greenbone Vulnerability Management (GVM) suite.
- **QualysGuard**:
    - A cloud-based vulnerability management and compliance solution that scans for vulnerabilities, misconfigurations, and compliance issues.
    - Popular in enterprise environments due to its extensive vulnerability database.

---

3. Exploitation Tools

These tools are used to exploit identified vulnerabilities to demonstrate real-world impact and provide proof of concept.

- **Metasploit Framework**:
    - The most well-known open-source framework for developing, testing, and executing exploits against target systems.
    - Includes thousands of exploit modules, payloads, and auxiliary tools for post-exploitation activities.
    - Used for automating tasks like privilege escalation and pivoting.
- **BeEF (Browser Exploitation Framework)**:
    - A tool designed to exploit vulnerabilities in web browsers.
    - Allows penetration testers to use hooked browsers as a foothold for further attacks.
- **Social-Engineer Toolkit (SET)**:
    - A tool specifically designed for social engineering attacks, such as phishing, spear phishing, and credential harvesting.
    - Used to simulate attacks that exploit human behavior rather than technical vulnerabilities.
- **SQLmap**:

    - An open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws.
    - Capable of taking over database servers using SQL injection vulnerabilities.

---

4. Password Cracking Tools

These tools are used to crack or recover passwords that have been hashed, encrypted, or obfuscated.

- **John the Ripper**:
    - A widely-used password cracking tool that supports a variety of hashing algorithms.
    - Capable of performing dictionary attacks, brute-force attacks, and rainbow table attacks on password hashes.
- **Hashcat**:
    - A fast and powerful password recovery tool that supports CPU and GPU acceleration.
    - Known for its ability to crack complex password hashes using advanced cracking techniques.
- **Hydra**:
    - A network login password cracking tool that supports brute-force attacks on numerous protocols, including SSH, FTP, HTTP, and RDP.
    - Efficient for conducting password guessing or password spraying attacks.

---

5. Web Application Penetration Testing Tools

These tools focus on finding and exploiting vulnerabilities in web applications.

- **Burp Suite**:
    - A powerful toolset for web application security testing.
    - Includes a proxy, scanner, and several modules for tasks like crawling, intercepting, and modifying HTTP requests.
    - Burp Suite Professional offers automated vulnerability scanning, while the free version is useful for manual testing.
- **OWASP ZAP (Zed Attack Proxy)**:
    - An open-source web application security scanner that helps in finding vulnerabilities such as XSS, SQL injection, and CSRF.
    - Includes an automated scanner as well as tools for manual testing.
- **Nikto**:
    - A web server scanner that detects potentially dangerous files, outdated versions of web servers, and vulnerabilities such as misconfigurations or exposed sensitive information.

- **W3af**:
    - A web application attack and audit framework designed to identify and exploit vulnerabilities in web applications.

---

6. Wireless Network Penetration Testing Tools

These tools are used to test and exploit vulnerabilities in wireless networks (e.g., Wi-Fi).

- **Aircrack-ng**:
    - A popular suite of tools for assessing Wi-Fi network security.
    - Can be used to capture and crack WEP and WPA/WPA2-PSK keys by capturing wireless traffic and performing brute-force or dictionary attacks.
- **Kismet**:
    - A wireless network detector, packet sniffer, and intrusion detection system.
    - Useful for discovering hidden or rogue wireless networks and monitoring wireless activity.
- **Reaver**:
    - A tool used to brute-force Wi-Fi Protected Setup (WPS) PINs, making it possible to recover WPA/WPA2 passwords.

---

7. Post-Exploitation Tools

These tools are used after gaining access to a system to maintain control, escalate privileges, and extract information.

- **Empire**:
    - A post-exploitation framework used to manage compromised systems.
    - It supports stealthy communications with the target using PowerShell and Python agents.
- **Mimikatz**:
    - A tool used to extract credentials and other secrets from memory on Windows systems.
    - Known for its ability to extract plaintext passwords, hash dumps, Kerberos tickets, and other credential data from Windows machines.
- **BloodHound**:
    - A tool designed for Active Directory exploitation.
    - Allows testers to map out an organization’s Active Directory environment and find potential paths to escalate privileges.

---

8. Mobile Application Penetration Testing Tools

These tools are used to assess the security of mobile applications on platforms like Android and iOS.

- **MobSF (Mobile Security Framework)**:
  - An automated mobile app security testing tool that supports Android and iOS.
  - Useful for static, dynamic, and malware analysis of mobile apps.
- **Frida**:
  - A dynamic instrumentation toolkit for iOS and Android, allowing testers to inject JavaScript into native apps for reverse engineering and debugging.
- **Drozer**:
  - A comprehensive security testing framework for Android apps.
  - Helps testers interact with Android devices to identify and exploit vulnerabilities in apps and device configurations.

---

9. Cloud Penetration Testing Tools

These tools help assess the security of cloud environments, like AWS, Azure, or Google Cloud.

- **ScoutSuite**:
  - An open-source multi-cloud security auditing tool.
  - Used to assess cloud environments for security issues, such as misconfigurations or exposed resources.
- **Pacu**:
  - An open-source AWS penetration testing tool.
  - Helps testers discover misconfigurations and security weaknesses in Amazon Web Services (AWS) environments.

---

10. Physical Security Penetration Testing Tools

These tools assist in testing the physical security of a facility, often used in social engineering or physical access testing scenarios.

- **Proxmark3**:
  - A versatile tool for RFID/NFC penetration testing.
  - Allows attackers to sniff, read, and clone RFID access cards used in physical security systems.
- **Lockpicks and Bypass Tools**:
  - Physical lock-picking tools for testing physical access controls (e.g., door locks, padlocks).
  - Used in physical penetration tests to assess the security of entry points and physical defenses.

11. Reporting and Documentation Tools

These tools help in generating reports and documenting the findings of the penetration test.

- **Dradis**:
  - A collaborative reporting tool for penetration testers that integrates with other tools like Burp Suite, Nessus, and Metasploit to streamline the documentation process.
- **Faraday**:
  - An integrated multi-user penetration testing environment that enables real-time collaboration and reporting.
- **PwnDocs**:
  - A modern reporting framework that integrates with popular tools and supports automated report generation.

4.7 Enumeration

Enumeration is defined as the process of extracting user names, machine names, network resources, shares and services from a system. In this phase, the attacker creates an active connection to the system and performs directed queries to gain more information about the target. The gathered information is used to identify the vulnerabilities or weak points in system security and tries to exploit in the System gaining phase.

Enumeration can be used to gather any of the following information:

- Operating system details
- Network infrastructure details
- Usernames of valid users
- Machine names
- Share names
- Directory names
- Printer names
- Web server details

Description of Enumeration

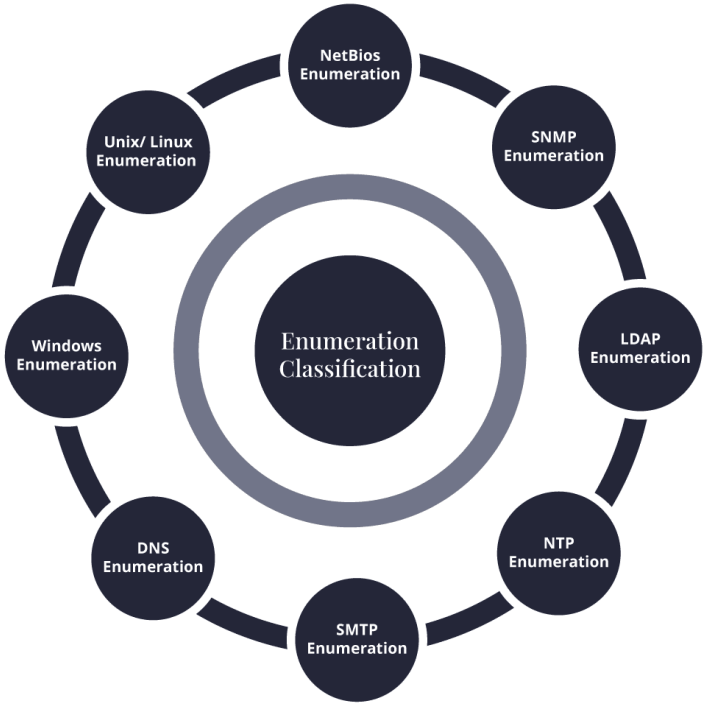
Enumeration is the phase 3 of the Penetration Testing or Ethical Hacking. It is a process of gaining complete access to the system by compromising the

vulnerabilities identified in the first two phases. The Scanning stage only helps to identify the vulnerabilities to a certain extent, but Enumeration helps us learn the complete details such as users, groups and even system level details – routing tables. This phase of the Ethical hacking is to gain end-to-end knowledge of what will be tested in the target environment. Tools are deployed to gain complete control over the system.

Significance of Enumeration

Enumeration is the most critical aspect of Ethical hacking. The metrics, outcomes, results are used directly in testing the system in the next steps of penetration testing. Enumeration helps us to decipher the detailed information – Hostnames, IP tables, SNMP and DNS, Application, Banners, Audit configurations and service settings. The significance of Enumeration is that it systematically collects details. This allows pentesters to completely examine the systems. The pen testers collect information about the weak links during the enumeration phase of ethical hacking. Enumeration helps in finding the attack Vectors and threats.

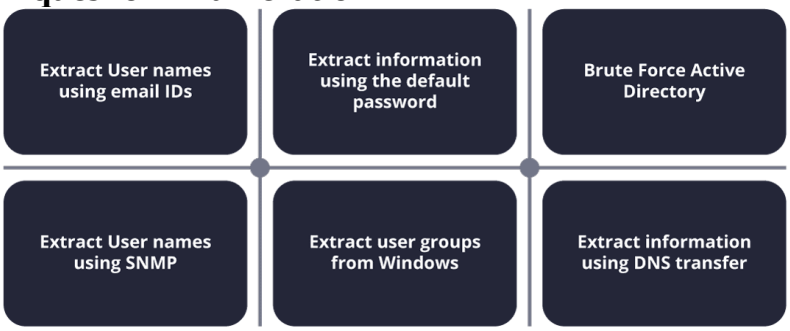
Enumeration Classification



Enumeration and its types – Tool box

Enumeration as a process extracts the user names, machine names, network resources, shares and services from the ecosystem. There is a robust toolbox that helps the enumeration process become scalable. This is a mix of software and hardware systems. There are free and commercial software tools for the enumeration. The hardware tools are mainly the key loggers and special wireless hardware. The pentesters find the right and optimum way to reach the various components of the systems.

Techniques for Enumeration



Types of information enumerated by intruders:

The types of the information enumerated by intruders are the following:

1. Network source
2. Users and groups
3. Routing tables
4. Audit settings
5. Service configuration settings
6. The various machine names
7. Applications
8. Banners
9. SNMP details
10. DNS details

Services and Port to Enumerate

What are the goals of the Enumeration?

Goal 1 – To map the end-to-end details that we need to check after the enumeration step

Goal 2 - The ways to execute the attacks in the upcoming phases

Goal 3 – Identify all the information we need to do the execution in future testing

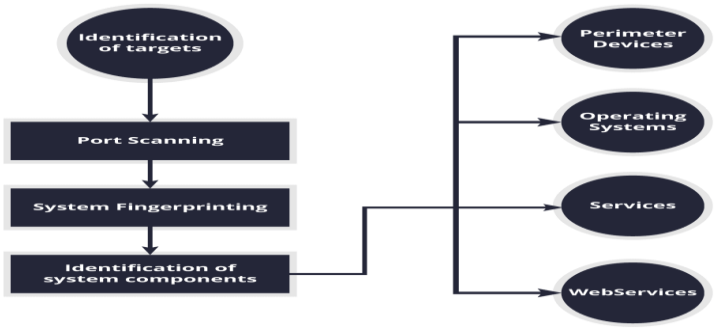
Goal 4 – Compile a list of devices with configuration for testing

Goal 5 – Complete the network map to finalize the steps for testing

Goal 6 – Compile the list of people who support the testing

Goal 7 – Collect even irrelevant information that might still be significant in the future

Process of Enumeration



4.8 REMOTE EXPLOITATION – WEB APPLICATION EXPLOITATION

An exploit (in its noun form) is a segment of code or a program that maliciously takes advantage of vulnerabilities or security flaws in software or hardware to infiltrate and initiate a denial-of-service (**DoS) attack** or install malware, such as **spyware**, **ransomware**, **Trojan horses**, **worms**, or viruses. So the exploit is not the malware itself but is used to deliver the malware. To exploit (in its verb form) is to successfully carry out such an attack.

When developers produce an operating system (OS) for a device, write code for software, or develop an application, bugs often appear due to inherent imperfections. These bugs can create a vulnerability in the system, and an exploit searches out such vulnerabilities and looks for a way to exploit databases and networks or systems.

If the bug is not reported and "patched," it becomes an entryway, so to speak, for cyber criminals to conduct an exploit. With so many devices connected together in the modern world, as in the Internet of Things (IoT), for example, an exploit does not just compromise a singular device, but it can become a security vulnerability for a whole network.

The Different Types of Exploits

Hardware

Hardware, to various degrees, must run on an OS, whether it be a complex OS for a PC or a simpler OS for an edge device. Vulnerabilities in the OS become entry points for an exploit, which can corrupt the memory or cause the device to "freeze."

Software

Software bugs, a normal consequence of software development, can become vulnerabilities open to exploits if not patched or fixed. Some of the common exploit methods include memory safety violations, input validation errors, side-channel attacks, and privilege confusion bugs.

Network

Each of the components of a network offers the possibility of vulnerability, whether hardware, software, or firewall configurations. Some attacks that may be part of an exploit can be domain hijacking, DoS and **distributed denial-of-service (DDoS) attacks**, and malware.

Personnel

Even personnel can be exploited. Cyber criminals may target their devices and credentials by means of **social engineering attacks**, **spear phishing**, and honey trapping. Training and access control are crucial to mitigating this vulnerability.

Physical Site

Exploits can be conducted on-site and if deficient physical security or inadequate access control exists. Just as a thief can break in and steal, a cyber criminal can break in (physically or remotely) and conduct an exploit that compromises an entire network.

1. SQL Injection (SQLi)

- **Description:** Attackers inject malicious SQL queries into input fields to manipulate or access a database.
- **Example:** By entering SQL code in a login form, attackers can bypass authentication or extract sensitive information from the database.
- **Mitigation:** Input validation, prepared statements, and parameterized queries help prevent SQLi.

2. Cross-Site Scripting (XSS)

- **Description:** Attackers inject malicious scripts into webpages, which are then executed in other users' browsers, allowing them to steal data or control interactions.
- **Types:** Reflected, stored, and DOM-based XSS.
- **Mitigation:** Input sanitization, output encoding, and using Content Security Policies (CSPs).

3. Cross-Site Request Forgery (CSRF)

- **Description:** Attackers trick users into performing actions they did not intend to on a web application where they're authenticated.
- **Example:** Submitting a forged request (like a money transfer) while the victim is logged in.
- **Mitigation:** CSRF tokens, user interaction confirmation, and same-site cookies.

4. Remote Code Execution (RCE)

- **Description:** Attackers exploit application vulnerabilities to execute arbitrary code on the server, potentially taking control of the web server.
- **Example:** Unsanitized user inputs leading to OS command injection.
- **Mitigation:** Input validation, limiting permissions, and keeping software updated.

5. File Inclusion Vulnerabilities (LFI/RFI)

- **Local File Inclusion (LFI):** Exploits include sensitive files on the server.
- **Remote File Inclusion (RFI):** Allows inclusion of files from remote servers, enabling malicious code execution.
- **Mitigation:** Restrict file paths, sanitize input, and disable unnecessary functions.

6. Broken Authentication and Session Management

- **Description:** Flaws in authentication or session handling allow attackers to impersonate other users.

- **Example:** Session ID prediction or fixation.
- **Mitigation:** Strong authentication controls, session expiration, and secure handling of session tokens.

7. Insecure Deserialization

- **Description:** Attackers manipulate serialized data to inject malicious code or tamper with serialized objects.
- **Example:** PHP object injection.
- **Mitigation:** Validate data before deserialization, use secure serialization formats, and restrict object types.

8. Path Traversal

- **Description:** Attackers manipulate URL paths to access restricted directories and files on the server.
- **Example:** Using ../ in the URL path to reach sensitive files outside the intended directory.
- **Mitigation:** Restrict directory access and sanitize inputs to disallow traversal characters.

9. Unvalidated Redirects and Forwards

- **Description:** Attackers use unvalidated redirects to lure users to malicious sites or conduct phishing attacks.
- **Mitigation:** Avoid open redirects or validate URLs against a whitelist.

Prevention Strategies

- **Secure Coding Practices:** Use secure coding guidelines and perform code reviews.
- **Input Validation and Sanitization:** Treat all input as untrusted and validate or sanitize it.
- **Regular Patching and Updates:** Update software, frameworks, and libraries to minimize vulnerabilities.
- **Web Application Firewalls (WAFs):** Deploy WAFs to detect and block common attack patterns.
- **Security Testing and Audits:** Perform regular penetration testing, vulnerability scans, and security audits.

4.9 EXPLOITS AND CLIENT SIDE ATTACKS

Client-side exploits are very useful to attackers when the client is behind the firewall or any application security layer. In this situation, it is not possible to directly access the client through the network.

The success rate of finding a vulnerability in the client site is directly proportional to the reconnaissance. If you are performing penetration testing on any application to test is there any client-side exploitation is possible or not you must have an understanding of possible attack scenarios to find and prevent the Client-Side Exploitation on your application.

Attack Scenario's In Client-Side Exploitation:

E-Mails with Malicious Attachments-

In this particular attack scenario, the attacker will send the malicious files such as PDF, exe, or mp3 in the hope that the victim would click on the link and download and execute the attachment. Upon execution, the attacker has a Meterpreter session opened on the victim's machine.
This attack can be a bit difficult to accomplish, as the attacker needs to convince the victim to execute their .exe file. Another major hurdle would be the victim's antivirus, which you need to bypass. Luckily, Metasploit has some built-in encoding mechanisms that, when used effectively, can evade some anti viruses, and if used effectively. However, all this is based on trial and error.

Malware Loaded on USB Sticks-
This method can be used by an attacker when he/she have physical access to the victim's machine. The attacker loads up a malicious PDF file or a malicious executable payload via a USB stick. Once the USB stick is inserted, malicious code will automatically be executed and the attacker would get a meterpreter session opened on the victim's machine. Teensy USB is a device that has the capability to emulate a mouse and keyboard.

E-Mails Leading to Malicious Links-
In this particular attack scenario, the attacker will send malicious links in the hope that victims would click on them. The link could be a fake log-in page or a web server hosted with an attacker's malicious code. Considering the attacker is hosting a web server, the code will be executed in the victim's browser and an attacker will have a meterpreter session opened.
In this scenario, the attacker sends the victim a malicious link, and when the victim clicks on it, we will be able to perform various attacks. Here are some examples:

- An attacker can set up a fake log-in page of any particular website, for example, facebook.com, and ask the victim to log in to the fake log-in page actually located at facebookfake.com

- If the attacker is on the same network as the victim, he can launch a DNS spoofing attack, where we can replace the IP of example.com with that of the attacker's fake log-in page, and as soon as the victim visits example.com, he would log in to our fake page instead
- An attacker can also perform DNS spoofing, where instead of the fake log-in page we can redirect the victim to our malicious webserver that would use relevant browser exploits to compromise the victim's browser.

Browser Exploitation-

Browser-based exploits are one of the most important forms of client-side exploits. Imagine a scenario where you are pen-testing against an organization. If it's an internal pentest, you would already own a box on the LAN. If it's an external pentest you need to somehow gain access to a system. You can set up a malicious web server and ask the victim to visit the server. As soon as he clicks your link, he gets compromised.

Most of the employees of an organization frequently browse on social networking websites like Facebook and Linkedin. We, as penetration testers, can take advantage of this and send malicious links to the employees and compromise them. On an internal network, the attacker could simply use a DNS poisoning attack to redirect victims to his malicious webserver. To sum up, there is a whole lot of attack surface when it comes to browser exploitation.

Compromising Client-Side Update-

In this scenario, an attacker will utilize previously mentioned attack vectors to compromise the client-side updating process. It means that whenever a victim updates a particular software, he will download malicious code instead of updates. In this scenario, the attacker will compromise client-side updates by using a neat tool called Evilgrade, which comes preinstalled with BackTrack. Evilgrade takes advantage of insecure update processes as the user normally does not double-check before an update because they trust that the application is being downloaded from the right place.

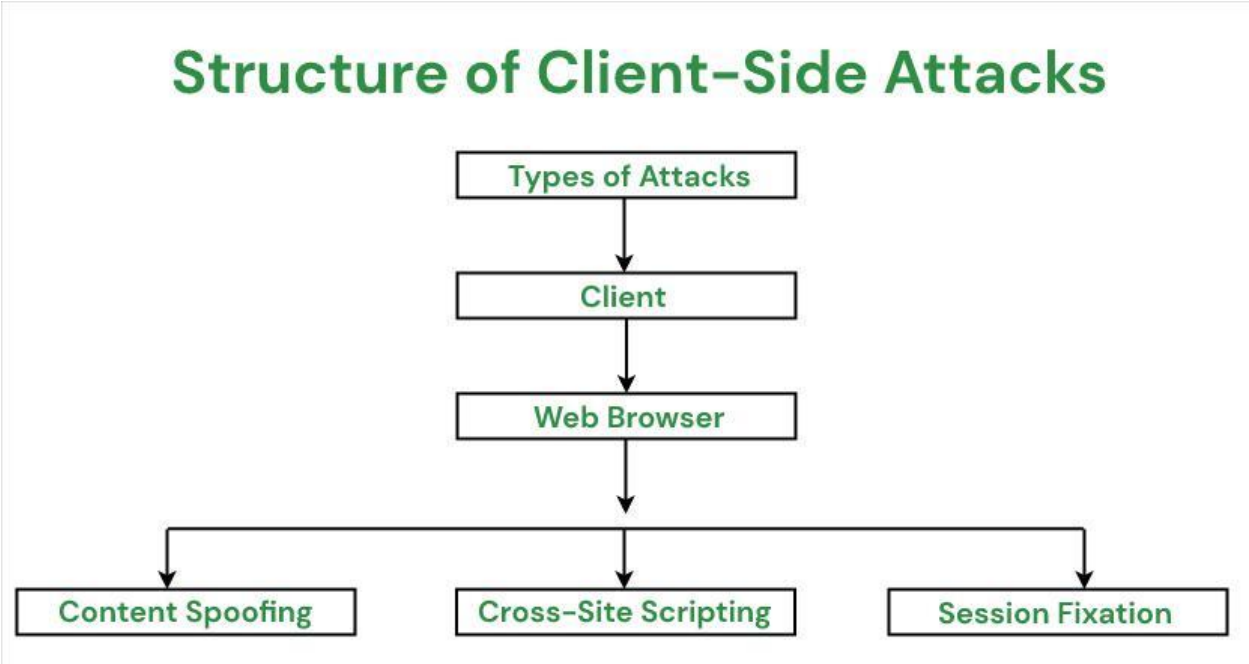
Social Engineering Attack-

In a social engineering penetration test, the organization may ask you to attack its users. This is where you use speared phishing attacks and browser exploits to trick a user into doing things they did not intend to do. A social engineering attack can be part of a network penetration test in some cases.

In the current situation of online work and education, social engineering attack is increasing day by day. Almost every person who has an email address and mobile number may get a victim of social engineering attacks for the attacker. In social engineering attack, the attacker fraudulently obtains confidential data from the users.

How To Protect From Client-Side Exploitation:-

1. In client-side exploitation, attackers take advantage of the weakest link that is clients.
2. To protect yourself from client-side attacks you have to be alert during your everyday Internet surfing.
3. Don’t open any link coming from a malicious or unknown person.
4. After opening any email attachment always make sure that the mail is coming from an authorized source.
5. Avoid downloading .exe attachments of a mail.
6. Always check the confidentiality of the domain of the website after filling in confidential information like username, password, card number, bank account number, etc.



Types of Client-Side Attacks:

- **Content Spoofing**: Content Spoofing is one of the common web security vulnerabilities. It allows the end user of the vulnerable web application to spoof or modify the actual content on the web page. The user might use the security loopholes on the website to inject the content that he/she wishes into the target website. When an application does not properly handle user-supplied

data, an attacker can supply content to a web application, typically via a parameter value, that is reflected back to the user.

- **Cross-site scripting**: In this type of attack, an attacker injects malicious codes into websites that are typically downloaded and displayed by a vulnerable browser. Cross-site scripting (XSS) is a computer security vulnerability typically found in web applications that enable attackers to inject client-side script into web pages viewed by other users.

- **Session Fixation Attack**: A session fixation attack is a type of remote code execution attack which is used to exploit software designed with the web-server Session Management feature. When a website is running an HTTP server, the server's session state information can be stolen and then retrieved by an attacker to take over the browser or use it for further attacks. There are many tools that can help you

- detect session fixation attacks in your organization in order to prevent future attacks. A Session fixation attack is also known as Session Fixation Vulnerability (SFV).

Detection:

The best way to mitigate client-side attacks is through system patching. System patching is the most basic and cost-effective method to lower the attack surface. It is important to keep up with patches and avoid vendor updates that patch vulnerabilities in software that are not relevant to the overall corporate environment. The best way to prevent client-side attacks is through a secure, strong password policy that dictates common passwords or patterns of many modern authentication technologies, including NTLM, MD4, DIGEST-MD5, and SHA1.

4.10 POST EXPLOITATION

Purpose of Post Exploitation

The post-exploitation is used to determine the capabilities and base value of the target system. The main purpose of post-exploitation is to gain access to all parts of the target system without knowing the user or without being detected. If the attacker is detected, it will make all the effects useless and everything null. A penetration tester is used to exploit the target's computer system without any authentication and analyze the data's value presented on the system of the victim. The tester can dig even further to get more information about the target system if they deem the information valuable. A penetration tester can also analyze system configuration settings, communication modes, registry settings, and connectivity methods

by which specific networks are connected to the devices. In this process, the methods and requirements can vary from rules of engagements and situations.

Rules of Engagement

The post-exploitation consists of the set of rules which is used to protect the client and penetration tester. By using these rules, the unnecessary conflicts between the client and tester can be avoided. If anything does not need to be exploited, the tester will not exploit this. Using these engagement rules, we can avoid any unnecessary actions at all costs. There are two types of set of rules, which are as follows:

Protecting Ourselves

Before making any attack, the penetration tester should learn all the necessary details about the victim or victim system. It is really important for a penetration tester to protect their identity anyhow. When the required operation is done, the tester should avoid the risk of leaving traces. A tester should perform all the operations under strict confidentiality. If a tester is detected, due to this, the whole operation will be terminated. If the tester wants to ensure the safety of digital footprinting or personal information or information of the client, the penetration tester should perform the following steps:

- If the client is a company or business, we should sign a service level agreement or contract. This contract is used to break the security of company assets.
- If we want to store the extracted information for a confidential purpose, we should use strong encryption methods.
- If we want to store the information or data of the client, we tester should avoid personal devices.

Protecting the client

If the client is an individual user or a company or business, the safety of their information and data is upon us. Before the initiation of an attack, the penetration tester should have to follow the proper steps. The tester may also have analyzed the attack method's capabilities and effect and determined the best suitable method for the job. If we want to ensure the safety of both clients, the penetration tester should follow the following steps:

The tester should not involve in an exploitation exercise, which is not necessary.

Suppose the client is a company or business. In that case, the tester should not use attack methods as SSL stripping, DDoS (distributed denial of service), network packet sniffing, SQL injection without the client's proper permission. Due to these attacks, daily operators may be disturbed or halted.

Tools used for Post exploitation

Metasploit is the well known and most popular tool that is frequently used for post-exploitation. Under Metasploit, Meterpreter and other sub tools are developed, and it makes the task of post-exploitation easier and faster. The penetration testing toolkit is described by the Metasploit framework, which is used to exploit research tools and development platforms. Various auxiliary modules and pre-verified exploits are included in the framework for a handy penetration test. Metasploit also contains different handlers, encoders, and payloads, which can be mixed up to work on any pen test.

4.11 Bypassing Firewalls and Avoiding Detection

Bypassing Firewalls:

1. Identifying Firewall Rules: Enumerate firewall rules to identify potential weaknesses.
2. Exploiting Firewall Vulnerabilities: Utilize exploits to bypass or disable firewall protections.
3. Using Encryption: Encrypt communication to evade firewall detection.
4. Tunneling: Use tunneling protocols (e.g., SSH, VPN) to bypass firewall restrictions.
5. DNS Tunneling: Utilize DNS to tunnel malicious traffic.

Avoiding Detection:

1. Stealthy Communication: Use covert channels to communicate with C2 servers.
2. Encryption: Encrypt communication to evade detection.
3. Evasion Techniques: Utilize evasion techniques (e.g., code obfuscation) to avoid detection.
4. Living off the Land (LotL): Use native system tools to avoid introducing new software.
5. Fileless Malware: Utilize fileless malware to evade traditional detection methods.

Post-Exploitation Tools:

1. Metasploit: Exploitation framework with post-exploitation modules.
2. Cobalt Strike: Post-exploitation framework with evasion capabilities.
3. PowerShell Empire: Post-exploitation framework with stealthy communication.
4. Mimikatz: Tool for extracting credentials and bypassing authentication.
5. Burp Suite: Tool for web application testing and evasion.

Techniques for Maintaining Access:

1. Backdoors: Establish persistent access through backdoors.
2. Scheduled Tasks: Use scheduled tasks to maintain access.

3. Service Creation: Create malicious services to maintain access.
4. Registry Modifications: Modify registry settings to maintain access.
5. Kernel-Mode Rootkits: Utilize kernel-mode rootkits to maintain stealthy access.

Evasion Techniques:

1. Code Obfuscation: Obfuscate code to evade detection.
2. Anti-Debugging Techniques: Evade debugging and analysis.
3. Sandbox Evasion: Evade sandbox detection.
4. Memory Injection: Inject malware into memory to evade detection.
5. DLL Hijacking: Hijack legitimate DLLs to evade detection.

Best Practices for Defense:

1. Implement robust firewall rules.
2. Utilize intrusion detection and prevention systems.
3. Monitor network traffic for suspicious activity.
4. Implement endpoint detection and response.
5. Conduct regular security audits and penetration testing.

Resources:

1. OWASP Post-Exploitation Guide
2. PTES (Penetration Testing Execution Standard)
3. SANS Institute: Post-Exploitation and Persistence
4. Black Hat Conference: Evasion Techniques
5. Cybersecurity and Infrastructure Security Agency (CISA): Guidance on Post-Exploitation

Certifications:

1. OSCP (Offensive Security Certified Professional)
2. GPEN (GIAC Penetration Tester)
3. CEH (Certified Ethical Hacker)
4. PTCP (Penetration Testing Certified Professional)
5. OSCE (Offensive Security Certified Expert)

4.12 TOOLS FOR PENETRATION TESTING

Penetration testing requires a variety of tools to simulate real-world attacks and identify security weaknesses in systems, networks, and applications. These tools can help in the discovery, exploitation, and reporting of vulnerabilities. Below are some of the most commonly used tools across different categories of penetration testing:

1. Network Scanning and Discovery Tools

These tools help identify hosts, services, and potential attack surfaces in a network.

- Nmap (Network Mapper):
    - A powerful open-source network scanner used for network discovery and security auditing.
    - Can be used to detect open ports, running services, operating system versions, and network configurations.
- Angry IP Scanner:
    - A fast, lightweight, cross-platform IP address and port scanner.
    - Helps in discovering live hosts and services on a network.
- Netcat:
    - Known as the “Swiss army knife” of networking, it can be used for port scanning, banner grabbing, and network tunneling.
- Fping:
    - A high-performance ping utility for network discovery, allowing bulk pinging of IP addresses.

---

2. Vulnerability Scanning Tools

These tools help identify known vulnerabilities in systems, networks, or applications.

- Nessus:
    - A widely-used commercial vulnerability scanner for detecting known vulnerabilities, misconfigurations, and policy violations across networks and applications.
    - Useful for both compliance auditing and vulnerability assessments.
- OpenVAS (Open Vulnerability Assessment Scanner):

    - An open-source vulnerability scanner that can detect a wide range of vulnerabilities across network services and applications.
    - Part of the Greenbone Vulnerability Management (GVM) suite.
- QualysGuard:
    - A cloud-based vulnerability management and compliance solution that scans for vulnerabilities, misconfigurations, and compliance issues.
    - Popular in enterprise environments due to its extensive vulnerability database.

---

3. Exploitation Tools

These tools are used to exploit identified vulnerabilities to demonstrate real-world impact and provide proof of concept.

- Metasploit Framework:
    - The most well-known open-source framework for developing, testing, and executing exploits against target systems.
    - Includes thousands of exploit modules, payloads, and auxiliary tools for post-exploitation activities.
    - Used for automating tasks like privilege escalation and pivoting.
- BeEF (Browser Exploitation Framework):
    - A tool designed to exploit vulnerabilities in web browsers.
    - Allows penetration testers to use hooked browsers as a foothold for further attacks.
- Social-Engineer Toolkit (SET):
    - A tool specifically designed for social engineering attacks, such as phishing, spear phishing, and credential harvesting.
    - Used to simulate attacks that exploit human behavior rather than technical vulnerabilities.

- SQLmap:
    - An open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws.
    - Capable of taking over database servers using SQL injection vulnerabilities.

---

4. Password Cracking Tools

These tools are used to crack or recover passwords that have been hashed, encrypted, or obfuscated.

- John the Ripper:
    - A widely-used password cracking tool that supports a variety of hashing algorithms.
    - Capable of performing dictionary attacks, brute-force attacks, and rainbow table attacks on password hashes.
- Hashcat:
    - A fast and powerful password recovery tool that supports CPU and GPU acceleration.
    - Known for its ability to crack complex password hashes using advanced cracking techniques.
- Hydra:
    - A network login password cracking tool that supports brute-force attacks on numerous protocols, including SSH, FTP, HTTP, and RDP.
    - Efficient for conducting password guessing or password spraying attacks.

---

5. Web Application Penetration Testing Tools

These tools focus on finding and exploiting vulnerabilities in web applications.

- Burp Suite:
    - A powerful toolset for web application security testing.

    - Includes a proxy, scanner, and several modules for tasks like crawling, intercepting, and modifying HTTP requests.
    - Burp Suite Professional offers automated vulnerability scanning, while the free version is useful for manual testing.
- OWASP ZAP (Zed Attack Proxy):
    - An open-source web application security scanner that helps in finding vulnerabilities such as XSS, SQL injection, and CSRF.
    - Includes an automated scanner as well as tools for manual testing.
- Nikto:
    - A web server scanner that detects potentially dangerous files, outdated versions of web servers, and vulnerabilities such as misconfigurations or exposed sensitive information.
- W3af:
    - A web application attack and audit framework designed to identify and exploit vulnerabilities in web applications.

---

6. Wireless Network Penetration Testing Tools

These tools are used to test and exploit vulnerabilities in wireless networks (e.g., Wi-Fi).

- Aircrack-ng:
    - A popular suite of tools for assessing Wi-Fi network security.
    - Can be used to capture and crack WEP and WPA/WPA2-PSK keys by capturing wireless traffic and performing brute-force or dictionary attacks.
- Kismet:
    - A wireless network detector, packet sniffer, and intrusion detection system.
    - Useful for discovering hidden or rogue wireless networks and monitoring wireless activity.

- Reaver:
    - A tool used to brute-force Wi-Fi Protected Setup (WPS) PINs, making it possible to recover WPA/WPA2 passwords.

---

7. Post-Exploitation Tools

These tools are used after gaining access to a system to maintain control, escalate privileges, and extract information.

- Empire:
    - A post-exploitation framework used to manage compromised systems.
    - It supports stealthy communications with the target using PowerShell and Python agents.
- Mimikatz:
    - A tool used to extract credentials and other secrets from memory on Windows systems.
    - Known for its ability to extract plaintext passwords, hash dumps, Kerberos tickets, and other credential data from Windows machines.
- BloodHound:
    - A tool designed for Active Directory exploitation.
    - Allows testers to map out an organization's Active Directory environment and find potential paths to escalate privileges.

---

8. Mobile Application Penetration Testing Tools

These tools are used to assess the security of mobile applications on platforms like Android and iOS.

- MobSF (Mobile Security Framework):
    - An automated mobile app security testing tool that supports Android and iOS.
    - Useful for static, dynamic, and malware analysis of mobile apps.

- Frida:
    - A dynamic instrumentation toolkit for iOS and Android, allowing testers to inject JavaScript into native apps for reverse engineering and debugging.
- Drozer:
    - A comprehensive security testing framework for Android apps.
    - Helps testers interact with Android devices to identify and exploit vulnerabilities in apps and device configurations.

---

9. Cloud Penetration Testing Tools

These tools help assess the security of cloud environments, like AWS, Azure, or Google Cloud.

- ScoutSuite:
    - An open-source multi-cloud security auditing tool.
    - Used to assess cloud environments for security issues, such as misconfigurations or exposed resources.
- Pacu:
    - An open-source AWS penetration testing tool.
    - Helps testers discover misconfigurations and security weaknesses in Amazon Web Services (AWS) environments.

---

10. Physical Security Penetration Testing Tools

These tools assist in testing the physical security of a facility, often used in social engineering or physical access testing scenarios.

- Proxmark3:
    - A versatile tool for RFID/NFC penetration testing.

    - Allows attackers to sniff, read, and clone RFID access cards used in physical security systems.

- Lockpicks and Bypass Tools:
    - Physical lock-picking tools for testing physical access controls (e.g., door locks, padlocks).
    - Used in physical penetration tests to assess the security of entry points and physical defenses.

---

11. Reporting and Documentation Tools

These tools help in generating reports and documenting the findings of the penetration test.

- Dradis:
    - A collaborative reporting tool for penetration testers that integrates with other tools like Burp Suite, Nessus, and Metasploit to streamline the documentation process.
- Faraday:
    - An integrated multi-user penetration testing environment that enables real-time collaboration and reporting.
- PwnDocs:

A modern reporting framework that integrates with popular tools and supports automated report generation.