Web Application Security Assessment Using Vulnerability Assessment & Penetration Testing (VAPT) Methodology

ABSTRACT:

Vulnerability assessment and penetration testing-(VAPT) provides a critical observation of organization OS-operating systems, web servers, DB-database servers, access points, and loopholes or back doors. Attackers use these vulnerabilities to exploit the victim's system. It is better to find out these vulnerabilities in advance before the attacker does.

In this report, we proved Vulnerability Assessment and Penetration Testing (VAPT) as a Cyber defense technology, how we can provide active cyber defense using Vulnerability Assessment and Penetration Testing. It gives a more detailed view of threats, loopholes, bugs, back doors so that the information security specialist fixes all these vulnerabilities and back doors will help to provide more security and better protection from malicious attacks.

Vulnerabilities can be found in two ways, internal testing, and external testing. We described the complete life cycle of Vulnerability Assessment and Penetration Testing on systems or networks and proactive action taken to resolve that vulnerability and stop the possible attacks. We have described the complete process of how to use Vulnerability Assessment and Penetration Testing as a powerful Cyber Defense Technology. Meaningful information or data are transferred over the web, cyberattacks are increasing every day with the increased use of Web applications.

Globally, statistics show that more than 70 per- cent of the applications either have vulnerabilities that could potentially be exploited by a hacker, or worse, or they have already been exploited. So it needs to be secure. The best way to secure our web application of the website is to hack ourselves or by conducting penetration testing.

In this report, penetration analysis of web security issues of the website is presented, using Kali Linux, OWASP ZAP, Burp Suite, etc. VAPT ensures that organization applications, web servers, database servers brought back to the initial state.

By taking advantage of vulnerability, Cyber criminals is easily able to steal confidential data of the ICT, results in heavy loss. Vulnerability Assessment and penetration testing is a special approach to eliminate various security threats from the web application. By focusing high risk vulnerability such as SQL Injection, Cross Site Scripting, Local File Inclusion and Remote File Inclusion, in this paper, we have surveyed literatures to study the general mechanics of VAPT process and gather tools which can be useful during VAPT process.

Introduction:

Web Application Security Assessment is a comprehensive practice aimed at ensuring the security and resilience of web applications against a wide range of threats.

Vulnerability assessment is the initial step in the VAPT process. This phase involves systematically scanning and analyzing web applications to identify potential vulnerabilities and weaknesses. Various automated scanning tools and manual techniques are employed to locate security flaws, misconfigurations, and software vulnerabilities. Common vulnerabilities include SQL injection, cross-site scripting (XSS), authentication issues, and more. The identified vulnerabilities are then subjected to a risk assessment, which categorizes them based on their severity, potential impact on the application, and the likelihood of exploitation. This risk analysis guides organizations in prioritizing which vulnerabilities need immediate attention and remediation. After vulnerability assessment, penetration testing takes the process to the next level. This phase involves attempting to exploit the identified vulnerabilities in a controlled and ethical manner. Skilled penetration testers, often referred to as "white hat" hackers, employ various techniques to simulate real-world attacks. The objective is to understand how an attacker might exploit these vulnerabilities and whether the security controls in place can effectively detect or prevent such attacks. Penetration testing also aims to validate the vulnerabilities discovered during the assessment phase, ensuring that they are not false positives and represent genuine security risks. Upon completing both vulnerability assessment and penetration testing, comprehensive reports are generated. These reports detail the vulnerabilities found, their severity, and recommendations for remediation. They serve as a roadmap for organizations to improve their web application security.

Remediation efforts involve collaboration between the development and security teams. Action plans are created to patch or mitigate the vulnerabilities. Organizations may also consider implementing security best practices and measures to prevent future vulnerabilities.

VAPT is an ongoing process because web applications are continuously evolving, and new vulnerabilities emerge over time. Regular security assessments are essential to maintain a strong security posture. In addition, compliance requirements and industry standards often mandate such assessments to ensure the protection of sensitive data, maintain customer trust, and prevent data breaches and other security incidents.

SECURITY RISK MANAGEMENT:

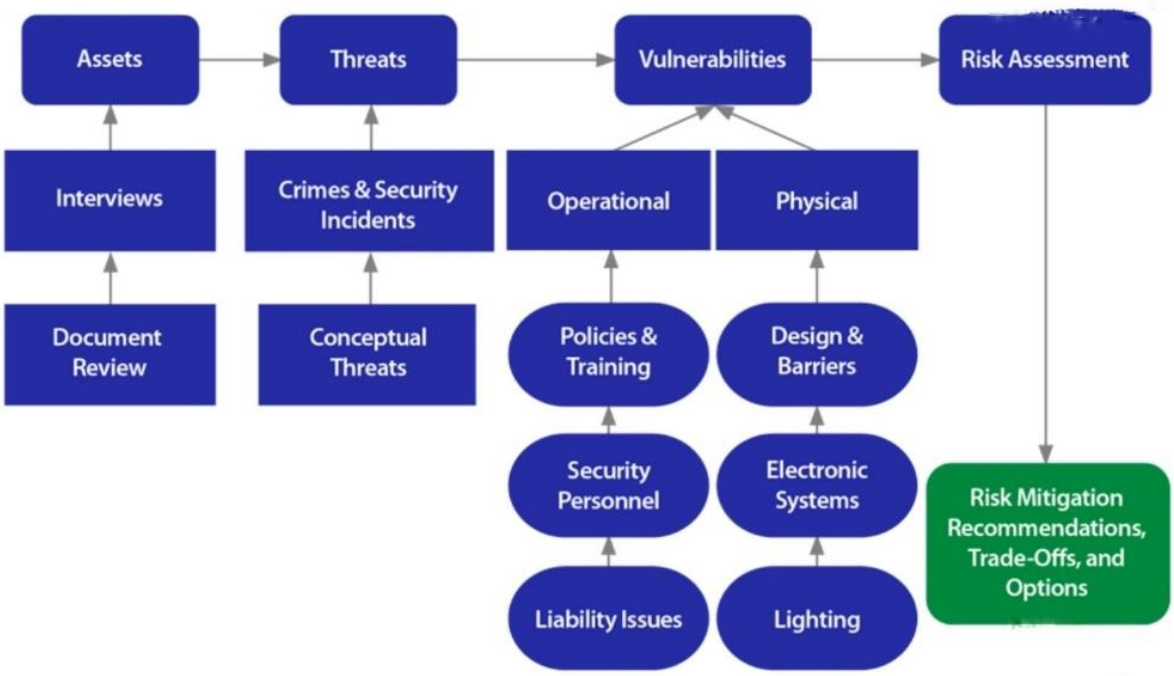


Fig-.1 Security risk management methodology

Given a specific risk, there are five strategies available to security decision makers to mitigate risk: avoidance, reduction, spreading, transfer and acceptance. The goal of most security programs is to reduce risk. Risk mitigation is accomplished by decreasing the threat level by eliminating or intercepting the adversary before they attack, blocking opportunities through enhanced security, or reducing the consequences if an attack should occur. Without question, the best strategy for mitigating risk is a combination of all three elements, decreasing threats, blocking opportunities and reducing consequences.

A logical mitigation strategy ties assets to threats to vulnerabilities to identify risks. Solutions for the identified risks typically enhance three facets of security: Policies, Procedures and Training; Physical/Electronic Security Systems; and Security Personnel. A sound mitigation strategy maximizes existing security resources (optimization) and prioritizes Policies first, Systems second, and Personnel third.

Threat Assessments
A Threat Assessment is a logical process used to determine likelihood of adverse events impacting your assets and to validate security levels. We utilize a number of different data sources to assess real, perceived, and conceptual threats.

Vulnerability Assessments
A vulnerability assessment, sometimes referred to as a security vulnerability assessment, is an analysis of security weaknesses and opportunities for adversarial exploitation in one or more of the above categories. The fundamental method for assessing vulnerabilities is the security survey, which is a tool for collecting information about the facility. The goal of a vulnerability assessment is to identify and block opportunities for attacks against assets. By effectively blocking opportunities, security decision makers can mitigate threats and reduce risk.

During a vulnerability assessment, we can also assess compliance with Crime Prevention through Environmental Design (CPTED). CPTED is a security concept that, through elements of the built environment, attempts to influence offender decisions that precede criminal acts.
CPTED is based upon the theory that the proper design and effective use of the built environment can reduce crime, reduce the fear of crime, and improve the quality of life. Strategies used in support of this concept include natural surveillance, natural access control, and natural territorial reinforcement.

Security Risk Modeling
Threat Analysis Group, LLC has experience developing evidence-based Security Risk Models based on variables (unique vulnerabilities and security posture) for companies with multiple locations. The objective of a Security Risk Model is to develop a model that incorporates the variables to identify risks to people and inform security decisions at each site. The goal of a Security Risk Model is to optimize Security by focusing on the Variables that actually impact Security Risk.
Risk Predictor Variables may include:

- Past Crimes and Threats
- Facility Characteristics (static and dynamic)
- Current Security Measures
- Existing Vulnerabilities

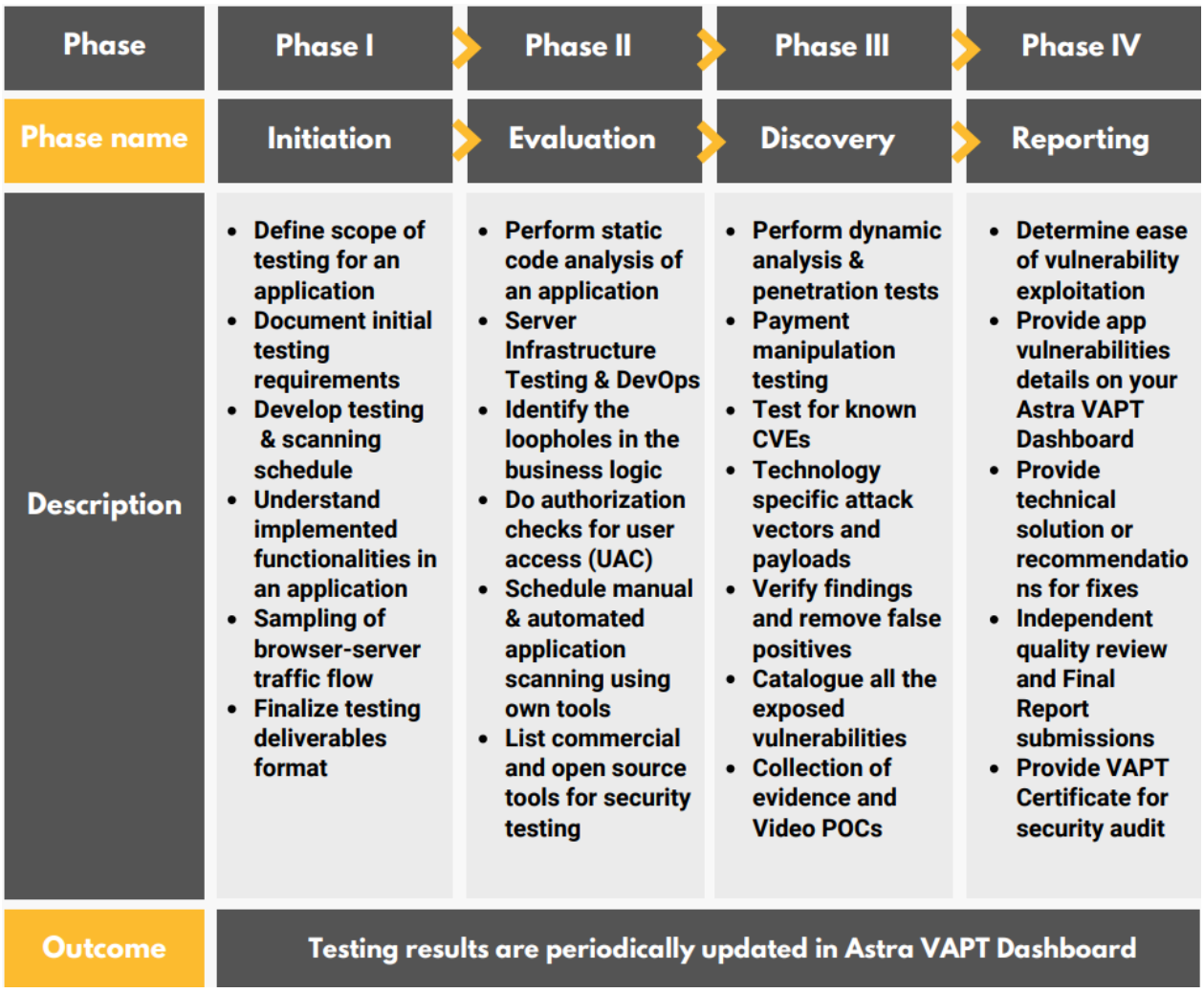


Fig-0.2: phases & their description.

Liability Analysis

Our security consultants evaluate your company's security program with an eye toward reducing liability exposure. Our assessments consist of a detailed analysis of crime including foreseeable crime on the property and in the area; vulnerability identification, risk mitigation strategies, and cost-effective security solutions. We will provide a written report with findings and recommendations for reasonable security measures.

Vulnerability:

What Exactly Vulnerability is?

A vulnerability in the context of cybersecurity is a weakness or flaw in a software application, system, or network that can be exploited by malicious actors. These vulnerabilities can exist at various levels, from code errors to misconfigurations, and they create opportunities for cyberattacks. When a vulnerability is exploited, it can lead to unauthorized access, data breaches, service disruptions, or other security incidents. Organizations must actively identify, assess, and remediate vulnerabilities to protect their systems and data

For example, the most common software security vulnerabilities include missing authentication for critical functions, absent authorization, missing data encryption, OS command injection, unrestricted upload of suspicious file types, and buffer overflow. While various security vendors have their own vulnerability and risk mitigation definitions, vulnerability management is often seen as an open, standards-based effort that uses the Security Content Automation Protocol (SCAP).

At an advanced level, SCAP can be broken down into four components:

- **Common vulnerabilities and exposures (CVE)** – Each CVE represents a specific vulnerability through which a cyber-attack may occur. The Strobes library is already aware of these risks and is capable of identifying them instantly.
- **Common configuration enumeration (CCE)** – A CCE is a list of system security configuration issues used to develop configuration guidance. Strobes is capable of identifying and registering these unique identifiers of system configuration issues that might become vulnerabilities.
- **Common platform enumeration (CPE)** – CPEs are standardized methods of defining classes of applications, operating systems, and devices within your environment. CPEs are used to describe what a CVE or CCE applies to. These are the end-points, i.e., digital systems, devices that are vulnerable.
- **Common vulnerability scoring system (CVSS**) – CVSS works to assign severity scores to each defined vulnerability, which is used to prioritize remediation efforts. Scores range from zero to ten, with ten being the most severe. Strobes' dynamic scoring ensures you're solving the most pressing vulnerabilities first.

Understanding Vulnerabilities:

Understanding vulnerabilities involves recognizing their diversity and the potential impact they can have on the digital landscape. This understanding serves as the foundation for developing effective strategies to mitigate and manage vulnerabilities. The subsequent sections of this document will delve deeper into specific types of vulnerabilities, their disadvantages, and mitigation approaches to bolster cybersecurity.

Vulnerabilities are inherent weaknesses or flaws in software, hardware, systems, and networks that can be exploited by malicious actors to compromise the security and integrity of digital assets. A thorough understanding of vulnerabilities is crucial for effective cybersecurity measures.

Vulnerabilities are essentially weaknesses that exist within the digital environment. These weaknesses can manifest in various forms, including coding errors, misconfigurations, flawed designs, and even human factors. Vulnerabilities create opportunities for attackers to gain unauthorized access, disrupt services, or steal sensitive data.

Types of Vulnerabilities:

Vulnerabilities can be categorized into different types, each with its own characteristics. Common categories include

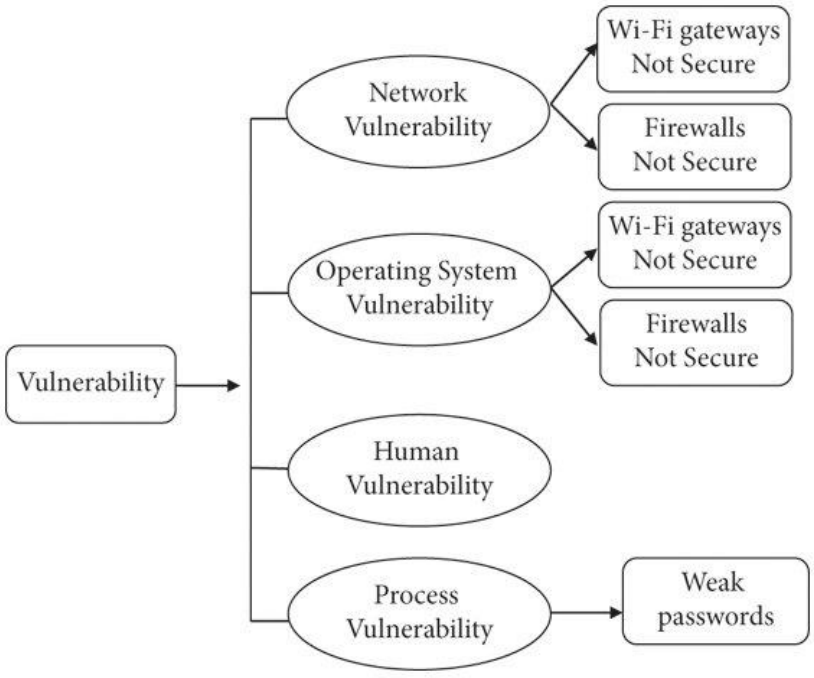


Fig-0.1: Vulnerability types

Software vulnerabilities may include buffer overflows, code injection, and authentication bypass. Network vulnerabilities could involve open ports or unpatched systems. Human-related vulnerabilities encompass social engineering attacks, insider threats, and the impact of human error.

Causes of Vulnerabilities:

Vulnerabilities can emerge from a variety of sources. These causes may be unintentional, such as programming errors, or intentional, as seen in social engineering attacks. Sources of vulnerabilities can include software bugs, misconfigurations, a lack of security awareness, insider threats, and even the complexity of modern technology.

1) Software Bugs and Flaws:

Software vulnerabilities typically result from programming errors and coding flaws. These errors can lead to issues such as buffer overflows, input validation problems, and logical errors within the code.

Impact:

Exploiting software bugs can allow attackers to gain unauthorized access, execute arbitrary code, or manipulate the application's behavior. These vulnerabilities can be unintentional but pose significant security risks.

2) Misconfigurations:

Misconfigurations occur when system settings, software, and security parameters are not appropriately configured. Examples include weak or default passwords, unnecessary services running, or incorrect access control settings.

Impact:

Misconfigurations can provide attackers with unauthorized access to systems or data, making it one of the leading causes of security incidents. Attackers often actively seek out and exploit misconfigurations.

3) **Third-Party Software and Libraries:**

Many applications rely on third-party software components and libraries. Vulnerabilities in these components, which may be integrated into the application, can be inherited and exploited.

Impact:

Exploiting vulnerabilities in third-party software can compromise the security of the entire application. It is crucial to stay updated with security patches for these components.

4) **Legacy Systems:**

Legacy systems are older, often unsupported, or outdated technologies that may lack security updates and patches. They may contain known vulnerabilities that have not been addressed.

Impact:

Attackers can target legacy systems with known vulnerabilities, exploiting them to gain unauthorized access, disrupt operations, or steal data. These systems often pose high risks.

5) **Social Engineering:**

Social engineering involves manipulating individuals to reveal sensitive information or perform actions. Tactics include phishing, pretexting, baiting, and tailgating.

Impact:

Social engineering attacks can lead to data breaches, unauthorized access, and financial losses. Human factors, such as employees being tricked, can result in vulnerabilities being introduced.

Sources of Vulnerabilities:

Top 7 databases to explore vulnerabilities are

- National Vulnerability Database
- Common Vulnerabilities and Exposures
- VULNDB – Vulnerability Intelligence
- DISA IAVA Database And STIGS
- Open Vulnerability and Assessment Language
- National Council of ISACs
- Security Tracker

Top 15 vulnerabilities, Corresponding CWE code, OWASP category, description and their potential business impact:

Fig-0.1: Web Application Vulnerabilities

A vulnerability in the context of cybersecurity is a weakness or flaw in a software application, system, or network that can be exploited by malicious actors. These vulnerabilities can exist at various levels, from code errors to misconfigurations, and they create opportunities for cyberattacks. When a vulnerability is exploited, it can lead to unauthorized access, data breaches, service disruptions, or other security incidents. Organizations must actively identify, assess, and remediate vulnerabilities to protect their systems and data.

1. **Injection Vulnerabilities**

CWE Code: CWE-89

OWASP Category: A1: Injection

Description:

Injection vulnerabilities are a class of security weaknesses in web applications that allow malicious users to exploit the application's input fields and manipulate its behavior. These vulnerabilities often result in the execution of arbitrary code or unintended database queries, which can have severe consequences.

SQL Injection (SQLi) is a well-known type of injection attack. In SQL injection attacks, malicious users manipulate the input data of a web application to inject rogue SQL queries into the application's database. When an application fails to properly validate or escape user inputs, attackers can gain unauthorized access to the database, retrieve, modify, or delete data, and, in some cases, take control of the entire system. SQL injection can lead to data breaches, data loss, and even full system compromise.

Other types of injection vulnerabilities include Command Injection, LDAP Injection, XPath Injection, and OS Command Injection. In Command Injection attacks, attackers insert malicious command strings into input fields that are executed by the server's underlying system, leading to unauthorized system commands execution. LDAP Injection manipulates queries to directory servers, potentially granting unauthorized access to sensitive directory data. XPath Injection involves injecting malicious XPath queries, leading to unauthorized data access within XML documents. OS Command Injection occurs when untrusted data is used to construct and execute operating system commands, compromising the server.

Potential Business Impact:

Injection vulnerabilities can lead to serious financial losses due to data breaches, unauthorized access to sensitive information, and manipulation of application behavior. These attacks can also damage an organization's reputation, erode customer trust, and result in regulatory fines or legal consequences. In cases of successful SQL injection, attackers can exfiltrate, modify, or delete valuable data, causing significant disruption to business operations. In addition to immediate financial and operational setbacks, the long-term impact includes loss of customer confidence, which can be challenging to regain.

2. **Cross-Site Scripting (XSS)**

CWE Code: CWE-79

OWASP Category: A7: Cross-Site Scripting (XSS)

Description:

Cross-Site Scripting (XSS) vulnerabilities allow attackers to inject malicious scripts into web pages viewed by other users. These scripts run in the context of the victim's browser, compromising data and session security. XSS vulnerabilities are pervasive and occur when an application fails to properly sanitize and validate user inputs.

There are three primary types of XSS attacks: Stored XSS, Reflected XSS, and DOM-based XSS. Stored XSS allows attackers to persistently inject malicious scripts into the application, affecting all users who view the compromised page. Reflected XSS involves tricking users into clicking on specially crafted links that execute malicious scripts. DOM-based XSS is an advanced variant where the client-side Document Object Model (DOM) is manipulated, affecting the page's structure and behavior.

XSS attacks can lead to session hijacking, theft of sensitive data (e.g., cookies), defacement of websites, and even distribution of malware. To mitigate XSS vulnerabilities, developers should implement proper input validation and output encoding. Content Security Policy (CSP) headers can help mitigate the impact of XSS by defining which scripts are allowed to run. Security-minded coding practices and regular testing are essential to protect web applications against XSS attacks.

Potential Business Impact:

XSS attacks can result in the theft of sensitive customer information, such as cookies and session tokens, leading to identity theft, fraud, and financial losses. Furthermore, attackers can deface websites, compromising the organization's professional image. The potential spread of malware through XSS can have severe consequences, including reputational damage, loss of customers, legal liabilities, and regulatory fines. The time and resources required to clean up after a successful XSS attack can be substantial, impacting business operations and incurring financial losses.

3. **Broken Authentication**

CWE Code: CWE-285

OWASP Category: A2: Broken Authentication

Description:

Broken authentication vulnerabilities occur when an application's authentication mechanisms are flawed, allowing unauthorized users to gain access to privileged accounts or perform unauthorized actions. These vulnerabilities can result from various issues, including weak password policies, session management flaws, or the absence of proper access controls.

Attackers can exploit broken authentication to gain unauthorized access to user accounts or administrative functions, potentially compromising sensitive data or disrupting the application's normal operation. To mitigate these vulnerabilities, developers should implement strong authentication and session management, enforce access controls, and regularly test the application for authentication-related weaknesses.

Potential Business Impact:

Broken authentication can lead to unauthorized access to user accounts and critical systems, resulting in financial losses and data breaches. Customers may suffer financial harm and loss of trust in the organization, causing reputational damage and potential legal consequences. Service disruptions, such as accounts being locked due to unauthorized access, can impact operations and lead to customer frustration. In regulated industries, non-compliance due to broken authentication may result in heavy fines.

Moreover, operational disruptions, customer support costs, loss of intellectual property, and a market disadvantage can add to the overall financial burden. Businesses also incur operational overhead to remediate these vulnerabilities, diverting resources from other critical projects. In essence, addressing broken authentication is crucial to avoid a broad spectrum of financial, reputational, and operational consequences.

4. **Sensitive Data Exposure**

CWE Code: CWE-200

OWASP Category: A3: Sensitive Data Exposure

Description:

Sensitive data exposure vulnerabilities occur when web applications fail to adequately protect confidential information, such as passwords or credit card details. Attackers can exploit these vulnerabilities to steal sensitive data, often through techniques like SQL injection or insecure direct object references (IDOR).

To mitigate sensitive data exposure, web applications should use strong encryption to protect data both in transit and at rest. They should also implement access controls and adhere to best practices for secure data storage and handling. Regular security audits are essential to identify and remediate data exposure vulnerabilities.

Potential Business Impact: Exposing sensitive data can have severe consequences, including data breaches that lead to financial losses, regulatory fines, and legal liabilities. The loss of customer trust can result in declining revenues and reputational damage that can affect the business for years. Customers may also experience financial harm, identity theft, and fraudulent activities, which can lead to legal actions against the organization. Moreover, the cost of incident response, notification, and credit monitoring services can be substantial.

5. XML External Entity (XXE) Injection

CWE Code: CWE-611

***OWASP Category**: A4: XML External Entity (XXE) Processing

Description:

XXE vulnerabilities arise when web applications parse XML input from untrusted sources without proper validation. Attackers can exploit these vulnerabilities to manipulate XML data, potentially leading to data exfiltration, server-side request forgery (SSRF), or denial-of-service (DoS) attacks.

To mitigate XXE vulnerabilities, developers should disable external entity processing, validate and sanitize XML inputs, and implement input validation mechanisms. Regular testing can help identify and address XXE issues in web applications.

Potential Business Impact:

XXE vulnerabilities can be exploited to gain unauthorized access to internal resources and sensitive data, which may lead to data breaches and theft of intellectual property. In addition to potential regulatory fines, there can be legal consequences for exposing sensitive customer information. Successful XXE attacks can also lead to denial-of-service (DoS) conditions, causing service disruptions and financial losses. The effort required to fix and secure vulnerable XML processing can be costly and resource-intensive.

6. Security Misconfiguration

CWE Code: CWE-2

OWASP Category: A6: Security Misconfiguration

Description:

Security misconfiguration vulnerabilities occur when web applications have inadvertently exposed security-related settings or have not properly configured access controls. These misconfigurations can enable attackers to exploit the application's weaknesses and gain unauthorized access.

Mitigating security misconfigurations involves a thorough review of the application's configuration settings, ensuring unnecessary services are disabled, implementing strong access controls, and adhering to the principle of least privilege. Regular security audits can help identify and fix security misconfigurations.

Potential Business Impact:

Security misconfigurations can lead to unauthorized access, data breaches, and service disruptions. Businesses may suffer financial losses due to regulatory fines, legal consequences, and damage to their reputation. Customers can lose trust, affecting the company's long-term success. The time and resources needed to address and remediate misconfigurations can also impact business operations and expenses.

Regulatory fines and legal liabilities may also be imposed, further adding to the financial burden. Additionally, operational disruptions can hinder business operations and increase operational overhead. Businesses must proactively address security misconfigurations to avoid a wide array of financial, reputational, and operational consequences.

7. **Cross-Site Request Forgery (CSRF)**

CWE Code: CWE-352

OWASP Category: A8: Cross-Site Request Forgery (CSRF)

Description:

CSRF vulnerabilities occur when an attacker tricks a user into performing actions without their consent within a web application in which they are authenticated. Attackers often use social engineering techniques to lure users into performing unwanted actions.

Mitigating CSRF vulnerabilities requires the use of anti-CSRF tokens, which validate that the requests originate from legitimate sources. Developers should implement proper anti-CSRF defenses to protect users and their data.

Potential Business Impact:

CSRF attacks can result in fraudulent actions on behalf of users, leading to financial losses, compromised data, and reputational damage. Unauthorized actions may disrupt service operations and lead to customer dissatisfaction. In some cases, regulatory fines can be imposed for inadequate protection against CSRF, further impacting the organization's bottom line.

8. **Insecure Deserialization**

CWE Code: CWE-502

OWASP Category: A8: Insecure Deserialization

Description:

Insecure deserialization vulnerabilities occur when untrusted data is deserialized without proper validation, potentially leading to remote code execution or other security issues. Attackers can exploit these vulnerabilities to manipulate data and execute arbitrary code on the server.

To mitigate insecure deserialization, developers should validate and sanitize deserialized data, avoid using serialized objects from untrusted sources, and implement proper security controls. Regular testing is crucial to identify and fix insecure deserialization issues.

Continuing with the descriptions of the remaining vulnerabilities in the next response.

Potential Business Impact:

Insecure deserialization can allow attackers to execute arbitrary code on the server, leading to system compromise, data breaches, and financial losses. The cost of recovering from such attacks, restoring services, and addressing legal and regulatory issues can be substantial. Reputational damage and loss of trust can also harm the long-term success of the business.

9. Insecure Direct Object References (IDOR)

CWE Code: CWE-639

OWASP Category: A4: Insecure Direct Object References

Description:

Insecure Direct Object References (IDOR) vulnerabilities occur when an attacker can manipulate references to objects such as files, database records, or other resources within an application. These vulnerabilities often lead to unauthorized access to sensitive data.

To mitigate IDOR vulnerabilities, developers should enforce proper access controls and implement validation mechanisms to ensure that users can only access objects they are authorized to access.

Potential Business Impact:

IDOR vulnerabilities can result in unauthorized access to sensitive data, leading to data breaches and potential legal consequences. Attackers can gain access to valuable intellectual property, financial data, or personally identifiable information, causing financial losses and reputational damage. Regulatory fines can also be imposed for failing to protect sensitive data.

The integrity of the business's reputation may be compromised, leading to customer mistrust and potential customer churn. Additionally, addressing and remediating IDOR issues can require substantial operational resources and overhead. Mitigating IDOR vulnerabilities is crucial for businesses to protect their sensitive data, avoid reputational damage, regulatory fines, and financial losses, and maintain customer trust.

10. Using Components with Known Vulnerabilities

CWE Code: CWE-937

OWASP Category: A9: Using Components with Known Vulnerabilities

Description:

Using components with known vulnerabilities is a security risk that arises when web applications incorporate third-party libraries, frameworks, or components with known security issues. Attackers can exploit these vulnerabilities to compromise the security of the entire application.

To mitigate this risk, developers should regularly update and patch all components, libraries, and dependencies used in their application. Additionally, they should stay informed about security vulnerabilities associated with these components and apply relevant security updates promptly.

Potential Business Impact:

Using components with known vulnerabilities can expose an organization to security breaches, data theft, and service disruptions. Financial losses may result from addressing security incidents, regulatory fines, and legal liabilities. Loss of customer trust and reputation damage can affect business in the long term.

11. Insufficient Logging and Monitoring

CWE Code: CWE-778

OWASP Category: A10: Insufficient Logging & Monitoring

Description:

Insufficient logging and monitoring vulnerabilities occur when web applications do not adequately log security-related events or fail to monitor these logs.

To mitigate this vulnerability, developers should implement thorough logging and monitoring mechanisms, define what events should be logged, and establish processes for monitoring and responding to security incidents.

Potential Business Impact:

Insufficient logging and monitoring can impede the detection of security incidents, making it challenging to respond to and mitigate breaches in a timely manner. This can

lead to prolonged service disruptions, reputational damage, financial losses, and potential regulatory fines for non-compliance.

12. **Security Headers Not Set**

CWE Code: CWE-16

OWASP Category: A5: Security Misconfiguration

Description:

Security headers are essential for protecting web applications against various attacks. The absence of security headers, such as Content Security Policy (CSP), can leave an application vulnerable to a range of attacks, including Cross-Site Scripting (XSS) and clickjacking.

To mitigate this vulnerability, developers should implement and properly configure security headers in their application to prevent common security threats.

Potential Business Impact:

Absence of security headers can render an application vulnerable to a range of attacks, such as XSS or clickjacking. The potential consequences include data breaches, compromised user sessions, financial losses, and damage to reputation. Legal and regulatory implications may also arise, especially if customer data is compromised.

13. Unvalidated Redirects and Forwards

CWE Code: CWE-601

OWASP Category: A10: Unvalidated Redirects and Forwards

Description:

Unvalidated redirects and forwards occur when web applications allow users to navigate to different pages or sites without proper validation. Attackers can manipulate these redirects to trick users into visiting malicious sites.

To mitigate this vulnerability, developers should validate and sanitize redirects, ensuring that they are safe and originate from trusted sources.

Potential Business Impact: Unvalidated redirects and forwards can be exploited by attackers to manipulate user navigation and trick them into visiting malicious sites. This can result in financial losses, reputational damage, and legal consequences. Additionally, users may lose trust in the organization due to these security shortcomings.

14. Improper Error Handling

CWE Code: CWE-209

OWASP Category: A6: Security Misconfiguration

Description:

Improper error handling vulnerabilities occur when web applications disclose excessive information about errors and internal workings, potentially revealing sensitive information or exposing application details to attackers.To mitigate this vulnerability, developers should implement proper error handling mechanisms, avoiding the disclosure of sensitive information, and providing only necessary error details to users.

Potential Business Impact:

Improper error handling can disclose sensitive information, leading to data breaches, reputational damage, and potential legal liabilities. The financial cost of addressing security incidents, implementing proper error handling, and dealing with regulatory fines can be significant.

15. Insecure File Upload

CWE Code: CWE-434

OWASP Category: A8: Insecure Deserialization

Description:

Insecure file upload vulnerabilities occur when web applications allow users to upload files without sufficient validation. Attackers can exploit this vulnerability to upload malicious files, potentially executing code on the server or compromising its security.

Addressing these top 15 common web application vulnerabilities is crucial for maintaining the security and integrity of web applications, protecting sensitive data, and ensuring the trust of users and stakeholders.

Potential Business Impact:

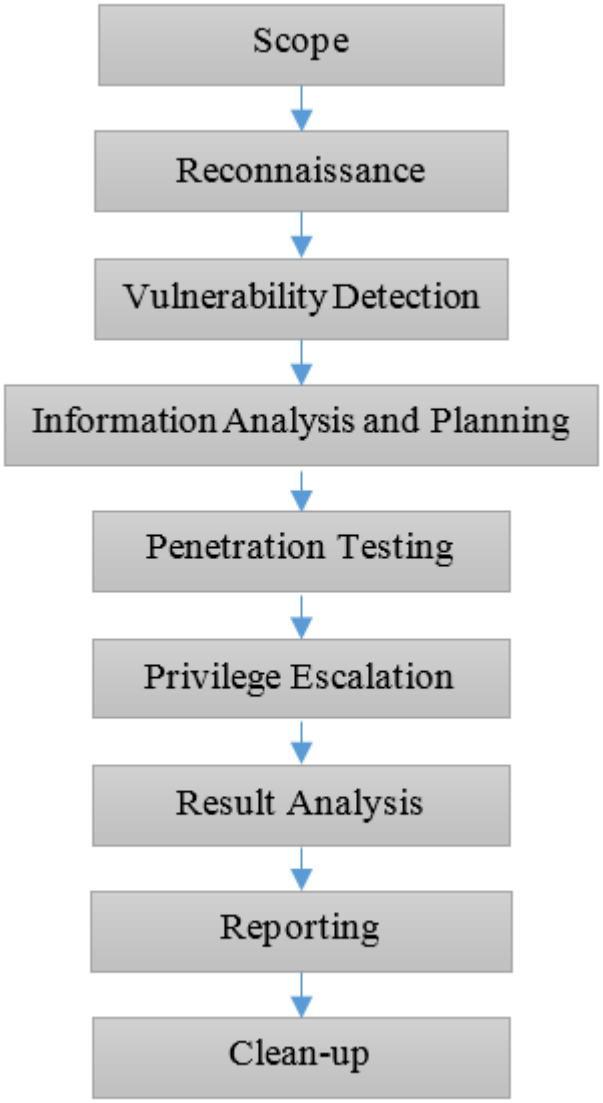
Insecure file uploads can lead to the execution of malicious code on the server, causing system compromise, data breaches, and financial losses. The time and resources required to fix these vulnerabilities, address security incidents, and potentially face legal and regulatory consequences can be substantial. Reputational damage and loss of trust can also affect the business's long-term success.

Number	Vulnerability	Risk
1	OS command injection	Critical
2	Frameable response (potential Clickjacking)	Critical
3	SQL injection	Critical
4	File path traversal	Critical
5	XML external entity injection	Critical
6	LDAP injection	Critical
7	XPath injection	Critical
8	Cross-site scripting (stored)	Critical
9	HTTP header injection	High
10	Cross-site scripting (reflected)	High
11	Cleartext submission of password	High
12	SSL cookie without secure flag set	Medium
13	Session token in URL	Medium
14	Password field with autocomplete enabled	Medium
15	Cookie without HttpOnly flag set	Low
16	File upload functionality	Info
17	Content type is not specified	Info

Fig-01: Vulnerability & their risk

Vulnerability Assessment and Penetration Testing (VAPT) theory:

Vulnerability Assessment and Penetration Testing (VAPT) are two critical components of a robust cybersecurity strategy. VAPT activities aim to identify, assess, and remediate security weaknesses within an organization's digital infrastructure. While often used interchangeably, these processes have distinct objectives.



Vulnerability Assessment:

Vulnerability Assessment focuses on systematically identifying and evaluating potential vulnerabilities within a system, network, or application.

Its core objectives include:

1. **Identification:**
   The process begins with identifying vulnerabilities in software, hardware, configurations, and human-related factors. Common vulnerabilities may include software bugs, misconfigurations, and outdated components.
2. **Scanning and Assessment:**
   Vulnerability scanners are employed to scan the target environment, seeking known weaknesses. The process involves classifying vulnerabilities by severity, impact, and likelihood of exploitation.
3. **Reporting:**
   The results of the assessment are reported, often using a Common Vulnerability Scoring System (CVSS) to rank vulnerabilities. The report helps organizations prioritize remediation efforts based on the level of risk.

Penetration Testing:

Penetration Testing, on the other hand, is a more aggressive and proactive approach. It involves simulating cyberattacks to exploit vulnerabilities, and its core objectives include:

1) **Recreating Real-World Scenarios:**
   Penetration tests aim to replicate the tactics used by malicious actors. This includes exploiting vulnerabilities, gaining unauthorized access, and demonstrating the potential impact of a successful attack
2) **Reporting and Remediation:**
   After the test, a comprehensive report is delivered to the organization, detailing vulnerabilities exploited, the impact of these exploits, and recommendations for mitigation. Organizations use these reports to remediate weaknesses.

Vulnerability assessment:

Fig-0.2: Vulnerability Assessment.

A vulnerability assessment is the testing process used to identify and assign severity levels to as many security defects as possible in a given timeframe. This process may involve automated and manual techniques with varying degrees of rigor and an emphasis on comprehensive coverage. Using a risk-based approach, vulnerability assessments may target different layers of technology, the most common being host-, network-, and application-layer assessments.

There are three primary objectives of a vulnerability assessment.

1. Identify vulnerabilities ranging from critical design flaws to simple misconfigurations.
2. Document the vulnerabilities so that developers can easily identify and reproduce the findings.
3. Create guidance to assist developers with remediating the identified vulnerabilities.

Vulnerability testing can take various forms. One method is Dynamic Application Security Testing (DAST). A dynamic analysis testing technique that involves executing an application (most commonly a Web application), DAST is performed specifically to identify security defects by providing inputs or other failure conditions to find defects in

real time. Conversely, Static Application Security Testing (SAST) is the analysis of an application's source code or object code in order to identify

The two methodologies approach applications very differently. They are most effective at different phases of the software development life cycle (SDLC) and find different types of vulnerabilities. For example, SAST detects critical vulnerabilities such as cross-site scripting (XSS) and SQL injection earlier in the SDLC. DAST, on the other hand, uses an outside-in penetration testing approach to identify security vulnerabilities while Web applications are running.

Vulnerability Assessment Types

Several types of vulnerability assessments can be conducted, including:

1. Network-Based Vulnerability Assessment

A network-based vulnerability assessment identifies vulnerabilities in network devices such as routers, switches, firewalls, and other network infrastructure components. The primary goal of a network-based vulnerability assessment is to identify weaknesses in the network that attackers could exploit to gain unauthorized access, steal data, or launch attacks.

Network-based vulnerability assessments typically involve specialized software tools and techniques that scan the network for vulnerabilities. These tools may use various methods to identify vulnerabilities, such as port scanning, vulnerability scanning, password cracking, and network mapping.

2. Application-Based Vulnerability Assessment

An application vulnerability assessment identifies vulnerabilities in software applications, including web applications, mobile applications, and desktop applications.

These assessments typically involve testing the application for common vulnerabilities, such as SQL injection, cross site scripting(XSS), and cross-site request forgery (CSRF). Application vulnerability assessments can be performed using both automated and manual methods.

OWASP consistently compiles a list of the most critical application vulnerabilities, updated periodically. In its latest **OWASP** ranking, the following vulnerabilities demand attention:

- A01:2021-Broken Access Control

- A02:2021-Cryptographic Failures
- A03:2021-Injection
- A04:2021-Insecure Design
- A05:2021-Security Misconfiguration
- A06:2021-Vulnerable and Outdated Components
- A07:2021-Identification and Authentication Failures
- A08:2021-Software and Data Integrity Failures
- A09:2021-Security Logging and Monitoring Failures
- A10:2021-Server-Side Request Forgery

3. API-Based Vulnerability Assessment

API vulnerability assessment is conducted to identify and mitigate potential security risks in APIs. This process identifies vulnerabilities and weaknesses in the API's design, implementation, and deployment. The goal is to ensure that the API is secure, reliable, and resilient to attacks.

The following vulnerabilities require specific attention in vulnerability assessment process to ensure the security and integrity of API interactions:

- API1:2023 Broken Object Level Authorization
- API2:2023 Broken Authentication
- API3:2023 Broken Object Property Level Authorization
- API4:2023 Unrestricted Resource Consumption
- API5:2023 Broken Function Level Authorization (BFLA)
- API6:2023 Unrestricted Access to Sensitive Business Flows
- API7:2023 Server-Side Request Forgery (SSRF)
- API8:2023 Security Misconfiguration
- API9:2023 Improper Inventory Management
- API10:2023 Unsafe Consumption of APIs

4. Host-Based Vulnerability Assessment

A host-based vulnerability assessment identifies vulnerabilities in individual host systems, including servers, workstations, and laptops.

These assessments typically involve scanning the host system for known vulnerabilities, such as missing security patches or outdated software. Host-based vulnerability assessments can be performed using both automated and manual methods.

5. Wireless Network Vulnerability Assessment

A wireless network vulnerability assessment focuses on identifying vulnerabilities in wireless networks, including Wi-Fi networks. These assessments typically involve testing the wireless network for common vulnerabilities, such as weak encryption, default passwords, and rogue access points.

Wireless network vulnerability assessments can be performed using specialized software tools and techniques.

6. Physical Vulnerability Assessment

A physical vulnerability assessment identifies vulnerabilities in physical security measures, such as locks, surveillance cameras, and access control systems. These assessments typically involve physical inspections of the facility and its security measures.

7. Social Engineering Vulnerability Assessment

A social engineering vulnerability assessment identifies vulnerabilities in human point, such as phishing attacks and other social engineering techniques.

This vulnerability assessment type typically involves simulated attacks against employees to test their awareness of security threats and their ability to identify and respond to them.

8. Cloud-Based Vulnerability Assessment

A cloud-based vulnerability assessment identifies vulnerabilities in cloud infrastructure and services, such as Amazon Web Services (AWS) and Microsoft Azure.

These assessments scan the cloud infrastructure for known vulnerabilities and test the security of cloud applications and services.

What Types of Threats Does Vulnerability Assessment Find?

Here are some of the most common types of threats that can be prevented through vulnerability assessment methods:

1. Malware Infections

Malware infections are among the most common cyber threats, which can devastate organizations. Malware is typically delivered through attack vectors such as phishing emails, malicious websites, and software vulnerabilities.

2. Denial of Service (DoS) Attacks

DoS attacks are a type of cyberattack that aims to overwhelm a targeted system or network with traffic or other resources, causing it to crash or become unavailable to legitimate users. Vulnerability assessment can identify vulnerabilities in the network or systems that attackers could exploit to launch DoS attacks.

3. Data Breaches

Data breaches occur when attackers gain unauthorized access to sensitive data, such as personal information, financial data, or intellectual property.

4. Insider Threats

Insider threats are threats that originate from within an organization. These threats could come from current or former employees, contractors, or business partners who can access an organization’s IT resources.

Vulnerability assessment can identify vulnerabilities in applications, systems, and network devices that insiders could exploit to steal data or cause damage to an organization’s IT infrastructure.

5. Phishing Attacks

Phishing attacks are a cyberattack that uses social engineering techniques to trick users into sharing sensitive information, such as login credentials or financial data.

6. Web Application Attacks

Web application attacks are a cyberattack that target web application vulnerabilities, such as SQL injection or cross-site scripting (XSS) attacks. Application vulnerability assessment can identify vulnerabilities in web applications and help organizations prioritize patching these vulnerabilities.

Vulnerability Assessment Methodology:

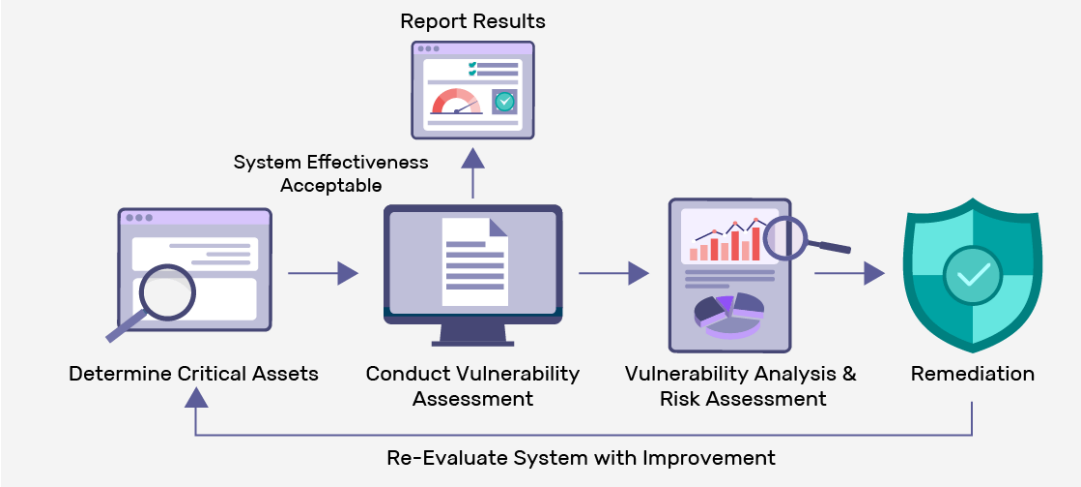


Fig- 0.5: Vulnerability Assessment Methodology Flow Chart.

1. Determine Critical and Attractive Assets

The first step in vulnerability assessment is understanding your entire ecosystem and determining which networks and systems are more critical to your business operation.

The attacker's objectives might vary from your perspective. Review each asset from an attacker's perspective and rank them based on attractiveness.

2. Conduct Vulnerability Assessment

Actively scan your entire network or system through automated tools to identify security flaws and weaknesses. The critical and attractive assets should be termed "targets," which requires further analysis, including testing with real-time scenarios to find and assess perceived security weaknesses. The assessments should rely on vendor vulnerability announcements, asset management systems, vulnerability databases, and threat intelligence feed.

The vulnerability assessment is complete if the overall network or system effectiveness meets the defined security requirements. If vulnerabilities are identified, you should proceed to the next phase.

3. Vulnerability Analysis and Risk Assessment

The next phase in the vulnerability assessment methodology is identifying the source and root cause of the security weakness identified in phase two. It offers a coherent view of remediation. It involves assigning the severity score or rank to each susceptibility based on factors like.

- What data are at risk?
- Which network or system is affected?
- The severity of the possible attacks
- Ease of compromise
- Potential damage if an attack happens

4. Remediation

The main objective of this phase is the closing of security gaps. For each identified vulnerability, determine the remediation actions. Certain remediation actions might include:

- Update all the configuration or operational changes
- Develop and implement vulnerability patches
- Implement new security measures, procedures, or tools

5. Mitigation

Not all vulnerabilities can be resolved completely; this is where mitigation comes into play. Mitigation focuses on lowering the chances of a vulnerability being exploited or minimizing the impact of its exploitation.

A practical approach, known as virtual patching**,** involves promptly applying a patch to the identified vulnerability without making any changes to the actual source code or components.

This virtual patch creates a protective barrier that prevents malicious actors from exploiting the vulnerability, effectively buying time until a permanent patch or code fix can be implemented.

6. Re-Evaluate System with Improvements

During this phase, the system's security posture is reassessed using similar methods as the initial assessment, which may include vulnerability scanning, penetration testing, code reviews, and other relevant techniques. The focus, however, is shifted toward determining whether the previously identified vulnerabilities have been successfully mitigated or reduced to an acceptable level.

The assessment also aims to identify any new vulnerabilities that have emerged due to the applied changes or configurations.

7. Report Results

The final phase in the security vulnerability assessment methodology is reporting the assessment result understandably.

The main goal of reporting is to clearly defining the system's effectiveness and recommending potential solutions if the current security measure seems ineffective.

A comprehensive vulnerability assessment report will include additional factors like:

- Which system is impacted?
- The level of simplicity in attacking or compromising the system
- The potential business consequences resulting from a successful breach
- Whether the vulnerability can be accessed via the Internet or demands physical proximity
- The age of the identified vulnerability
- Any regulatory obligations your organization adheres to
- The expense associated with a data breach in your specific industry

Penetration Testing:

A penetration test, also known as a pen test, is a simulated cyberattack 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 improper inputs that are susceptible to code injection attacks.

What is web application penetration testing?

Web application penetration testing is an essential part of web application security. It is the process of simulating hack-style attacks to identify potential vulnerabilities in web apps using simulated attacks. Web application penetration testing ensures that you are aware of weaknesses before a malicious actor takes advantage of it. This gives you an edge over your competitors and helps you build trust amongst your customers. A constant focus on security is an added feather on your cap. It helps organizations comply with security standards and regulations such as PCI-DSS, GDPR, etc. Web application testing should be performed regularly and periodically to ensure that the web applications are secure and up-to-date.

The aim is to identify all the vulnerabilities on the server side and in all the functionalities and components of the web application (front and backend, APIs, third-party services, etc.), measure their impact and propose corrective measures to strengthen the security of the target system.

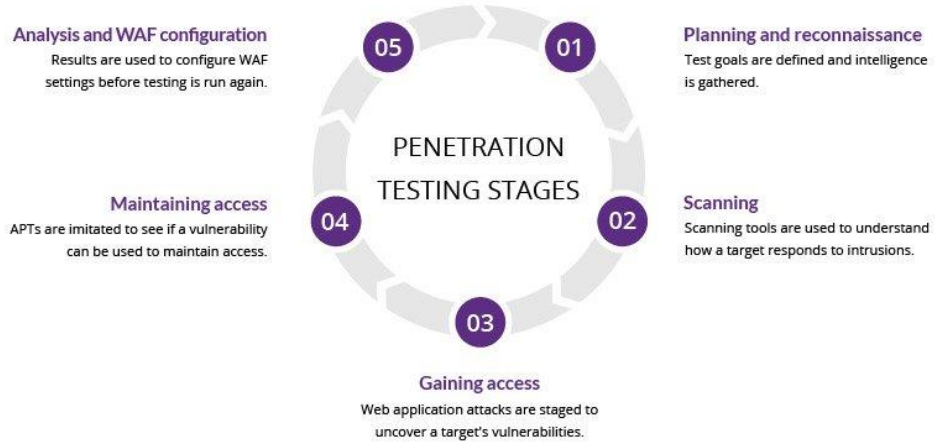


Fig-0.5: Stages in penetration testing.

Phase 1: Planning and Reconnaissance

The first penetration step involves planning to simulate a malicious attack – the attack is designed in a way that helps to gather as much information on the system as possible.

This is possibly one of the most time consuming stages as ethical hackers inspect the system, note the vulnerabilities, and how the organization's tech stack reacts to system breaches. The information searched ranges from names and email addresses of the company's employees to network topology, IP addresses, among others. It should be noted that the type of information or the depth of the investigation will depend on the objectives set for the audit. Some gathering methodologies include social engineering, dumpster diving, network scanning, and domain registration information retrieval.

Phase 2: Scanning

Based on the finding of the planning phase, penetration testers use scanning tools to explore the system and network weaknesses. This pen test phase identifies the system weaknesses that are potentially exploited for targeted attacks. It is essential to obtain all this information correctly, as it will define the success of the following phases.

Phase 3: Gaining System Access

Having understood the system's vulnerabilities, pen testers then infiltrate the infrastructure by exploiting security weaknesses. Next, they attempt to exploit the system further by escalating privileges to demonstrate how deep into the target environments they can go.

Phase 4: Maintaining Access

This pen test step identifies the potential impact of a vulnerability exploit by leveraging access privileges**.** Once they have a foothold in a system, penetration testers should maintain access and hold the simulated attack long enough to accomplish and replicate malicious hackers' goals. Therefore, in this pen test phase, we try to obtain the maximum level of privileges, network information, and access to as many systems as possible by identifying which data and/or services are available to us.

This is the phase in which we have to demonstrate what this security breach could mean for the customer. Gaining access to an old computer that is not even part of the domain is not the same as gaining direct access to passwords or compromised data.

Phase 5: Analysis and Reporting

This is the result of a penetration test. As part of the last stage, the security team prepares a detailed report describing the entire penetration testing process**.** Some information or detail that should appear are:

- The seriousness of the risks emanating from the vulnerabilities discovered
- The tools that can successfully penetrate the system
- Highlighting those points where security had been implemented correctly
- Those vulnerabilities that need to be corrected and how to prevent future attacks (remediation recommendations)

This phase is possibly the most important for both parties. As this report is likely to be read by both IT staff and non-technical managers, it is advisable to separate the report into a general explanation part and a more technical aspect, i.e., the executive report and the technical report.

Types of Penetration Testing:

There are five main types of pentesting, each of which focuses on different security vulnerabilities and uses a unique set of tools. Understanding the different forms of penetration testing is essential in ensuring that you can find the appropriate test to suit your needs.

1. Network Penetration Test

In a network penetration test, the penetration tester audits a network environment for security vulnerabilities. Network penetration tests can be further subdivided into two categories: external tests and internal tests. An external penetration test involves testing public IP addresses, whereas an internal test provides the tester with network access so that they can emulate a hacker who has already penetrated the network's defenses.

Penetration testers focus on the following areas in network penetration tests:

- Firewall configuration
- Firewall bypass testing
- Stateful inspection analysis
- Intrusion prevention system deception
- DNS-level attacks

2. Web Application Penetration Test

In a web application penetration test, testers search for security problems associated with the insecure design, development, or coding of a web app. These types of tests focus on browsers, websites, web applications, and related items, including plug-ins, procedures, and applets.

3. Client-Side Penetration Test

Client-side penetration tests identify security vulnerabilities within an organization. These are often located in the programs and applications the organization uses, such as email platforms, web browsers, and Adobe Acrobat.

Hackers may, for example, gain access to a vulnerable application through a well-crafted email directing an employee to a malicious webpage or load malware onto a USB stick that can execute the malware once it is inserted into a device. Client-side penetration tests aim to identify these risks and address all related internal vulnerabilities.

4. Wireless Network Penetration Test

Wireless network penetration tests focus on vulnerabilities in wireless devices, such as tablets, laptops, notebooks, and smartphones. These tests aim to identify all devices used by an organization that are vulnerable to cyberattacks. These vulnerabilities may include wireless devices' security controls, access point configurations, or weak security protocols.

5. Social Engineering Penetration Test

Social engineering penetration tests focus on the human aspect of an organization's security. In a social engineering test, testers attempt to deceive employees into giving up sensitive information or allowing the tester access to the organization's systems. This enables penetration testers to understand the organization's vulnerability to scams or other social engineering cyberattacks.

Testers often use phishing scams as part of social engineering tests. Physical testing may be another aspect of a social engineering test: Penetration testers can attempt to gain access to a secured building or location for which they don't have clearance by taking advantage of employees' ignorance of security protocols.

Different Approaches of Penetration Testing:

1. **Black Box Testing:** In black box testing, the tester adopts the role of an external attacker with no prior knowledge of the target system. This approach is akin to a real-world scenario where the tester must rely solely on external information. It provides a valuable perspective on an organization's security posture from an outsider's viewpoint, helping to identify vulnerabilities that might be exploited by malicious actors. However, due to the limited knowledge of the system, testers may not uncover all potential weaknesses, the testing process can be time-consuming as testers need to explore the system as if they were actual external threats.
2. **White Box Testing:** White box testing takes the opposite approach by providing testers with complete knowledge of the target system, including source code, network architecture, and internal workings. This approach allows for an in-depth assessment of the system's security, making it ideal for identifying and addressing architectural vulnerabilities and other intricate issues. It's highly beneficial for organizations looking to thoroughly assess their security, but it may not fully reflect the perspective of an external attacker. Moreover, white box testing can be resource-intensive because testers need deep technical knowledge.
3. **Gray Box Testing:** Gray box testing strikes a balance between black box and white box testing. Testers have partial knowledge of the system, simulating a scenario where an attacker might possess some insider information. This approach combines the realism of black box testing with the depth of white box testing. It's advantageous because it provides a more holistic view of security, but the extent of tester knowledge can vary, which may lead to differing assessment results.
4. **Internal Testing:** Internal penetration testing evaluates an organization's security from within its trusted network, simulating insider threats or attackers with some level of internal access. This approach assesses an organization's ability to detect and respond to internal threats while identifying vulnerabilities from an internal perspective.
5. **External Testing:** External penetration testing focuses on assessing the security of systems and networks from an external perspective, mimicking the actions of an external attacker. Testers aim to uncover vulnerabilities that real-world external adversaries could exploit, often through the internet. This approach provides insights into the organization's external attack surface and strengthens its defenses against external threats. Nevertheless, it may not uncover internal vulnerabilities, and scoping is crucial to avoid disrupting external services during the assessment.

What Types of Threats Does Penetration Testing Find?

1. **Vulnerabilities in Web Applications:** Penetration testing can uncover vulnerabilities like SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) in web applications. These flaws can be exploited by attackers to compromise sensitive data, deface websites, or gain unauthorized access.
2. **Weak Network Security:** Penetration testers can discover weak network configurations, misconfigured firewalls, and exposed network services. Such vulnerabilities may provide an entry point for attackers, enabling them to gain a foothold within an organization's network.
3. **Inadequate Access Controls:** This threat revolves around the discovery of improper access controls, where users have more privileges than necessary. Penetration testing can reveal unauthorized access to critical systems or data, posing significant security risks.
4. **Outdated Software and Patch Management:** Penetration tests can find systems with outdated software, missing security patches, and unpatched vulnerabilities. Attackers often exploit these weaknesses to launch attacks like ransomware, which can disrupt operations and lead to data loss.
5. **Social Engineering Vulnerabilities:** Testers assess an organization's susceptibility to social engineering attacks, such as phishing and pretexting. Discovering these vulnerabilities is crucial for improving employee awareness and reducing the risk of insider threats.
6. **Mobile Application Weaknesses:** Penetration testing evaluates the security of mobile applications and their interaction with back-end systems. Identifying weaknesses in mobile apps can prevent data breaches and unauthorized access to sensitive data.
7. **Misconfigured Cloud Services:** As organizations increasingly adopt cloud technologies, penetration testing can uncover misconfigured cloud services and storage, which may lead to data exposure. Addressing these issues is vital to maintain the confidentiality and integrity of cloud-hosted data.

Penetration Testing methodologies:

Penetration testing methodology is a specific course of action taken by a pentest provider to conduct the pentest of a target website or network. There are multiple penetration testing methodologies that can be put to use depending on the category of the target business, the goal of the pentest, and its scope.

There are various standards and methodologies that ensure the penetration test is authentic and covers all important aspects. Some of them are mentioned below:

1. OSSTMM
2. OWASP
3. NIST
4. PTES
5. ISSAF

OSSTMM:

OSSTMM is short for Open-Source Security Testing Methodology Manual. It is one of the most widely used and recognized standards of penetration testing. It's based on a scientific approach to penetration testing that contains adaptable guides for testers. You can use this to conduct an accurate assessment.

OWASP:

OWASP stands for Open Web Application Security Project. Widely known, this pentest standard is developed and updated by a community keeping in trend with the latest threats. Apart from application vulnerabilities, this also accounts for logic errors in processes.

NIST:

National Institute of Standards and Technology (NIST) offers very specific pentesting methodology for pentesters to help them improve the accuracy of the test. Both large and small companies, in various industries, can leverage this framework for a penetration test.

PTES:

PTES or Penetration Testing Execution Standards is a pentest methodology designed by a team of information security professionals. The goal of PTES is to create a comprehensive and up-to-date standard for penetration testing as well as to build awareness among businesses as to what to expect from a pentest.

ISSAF:

The Information System Security Assessment Framework (ISSAF) is a pentesting guide supported by the Open Information Systems Security Group. This is one of the security testing methodologies is not updated anymore, hence it is a bit out of data. Nevertheless, it is still in use for its comprehensive nature – it links different steps of the pentest process with relevant tools.

VAPT TOOLS:

NO.	Name	License	Type	Operating System
1	Metasploit	Proprietary	Vulnerability scanner and exploit	Cross-platform
2	Nessus	Proprietary	Vulnerability scanner	Cross-platform
3	Kali Linux	GPL	Collection of various tools	Linux
4	Burp Suite	Proprietary	web vulnerability scanner	Cross-platform
5	w3af	GPL	web vulnerability scanner	Cross-platform
6	OpenVAS	GPL	Vulnerability scanner	Cross-platform
7	Paros proxy	GPL	web vulnerability scanner	Cross-platform
8	Core Impact	Proprietary	Vulnerability scanner and exploit	Windows
9	Nexpose	Proprietary	Entire vulnerability management lifecycle	Linux, Windows
10	GFI LanGuard	Proprietary	Vulnerability scanner	Windows
11	Acunetix WVS	Proprietary	web vulnerability scanner	Windows
12	QualysGuard	Proprietary	Vulnerability scanner	Cross-platform
13	MBSA	Freeware	Vulnerability scanner	Windows
14	AppScan	Proprietary	web vulnerability scanner	Windows
15	Canvas	Proprietary	Vulnerability scanner and exploit	Cross-platform

Fig-0.5: Top 15 VAPT tools.

Conclusion:

Web Application Security Assessment using Vulnerability Assessment and Penetration Testing (VAPT) methodology is an integral defense against the ever-evolving landscape of cyber threats targeting web applications. These applications are highly susceptible due to their internet accessibility, making them prime targets for malicious actors seeking to exploit vulnerabilities. VAPT provides a proactive approach by systematically identifying and mitigating these vulnerabilities, preventing potential security breaches, data loss, and damage to an organization's reputation. In addition to bolstering security, VAPT also assists organizations in complying with regulatory requirements. Many industries have established specific compliance mandates necessitating regular security assessments. For instance, the Payment Card Industry Data Security Standard (PCI DSS) requires organizations handling payment card data to conduct routine penetration testing and vulnerability assessments. By adhering to these regulations, organizations not only avoid legal and financial penalties but also build trust with customers who expect their data to be safeguarded.

It's important to note that VAPT is not a one-time event but a continuous improvement process. Web applications and the threats targeting them are constantly evolving. Regularly repeating the VAPT process ensures that an organization's security measures remain effective and that web applications stay well-protected against emerging threats.

Moreover, VAPT can be employed to assess not only an organization's in-house web applications but also third-party vendor services. This practice ensures that external services do not introduce vulnerabilities into the organization's environment, as third-party breaches can have far-reaching consequences.

Lastly, VAPT methodologies are highly customizable to fit an organization's unique web application and security requirements. This adaptability allows organizations to tailor assessments to their specific needs, addressing the most critical areas of concern effectively.

In today's digital landscape, where cyber threats are persistent and potentially devastating, VAPT is an invaluable practice for organizations committed to data security and the seamless operation of web applications. It goes beyond risk mitigation, enabling organizations to adapt and respond effectively to emerging security challenges, maintain customer trust, and sustain business continuity.