Review of Computer Engineering Studies
Vol. 9, No. 1, March, 2022, pp. 1-22
Journal homepage: http://iieta.org/journals/rces

A Review on Web Application Vulnerability Assessment and Penetration Testing

Urshila Ravindran[1], Raghu Vamsi Potukuchi[2*]

[1] Security Associate, Safe Security, Okhla, Delhi 110020, India
[2] Department of Computer Science and Engineering, JIIT University, Noida 201309, India

Corresponding Author Email: raghu.vamsi@jiit.ac.in

https://doi.org/10.18280/rces.090101

Received: 24 October 2021
Accepted: 22 December 2021

Keywords:
vulnerability assessment, penetration testing, web application, security, ethical hacking, burp suite, application security

ABSTRACT

With the increase in the number of internet users, web applications, user data there is an increase in the number of hackers all over the world. It is becoming challenging for organizations to ensure the security of the data of their employees and their customers around the world. Any cyber-attack on the organization will drastically affect the reputation of the organization as well as the loss of trust from the users or customers. Customers will not invest in these organizations who have encountered a cyber threat or attack. Hence, enabling regular security testing and checks by the penetration testers or security analysts help in preparing the organization from any security threat by testing network and applications. Even after performing the Vulnerability Assessment and Penetration Testing (VAPT) of the applications, it is extremely necessary to follow up the security patches to mitigate all the existing flaws and security vulnerabilities in the web applications under the organization. To this end, this paper presents the common web application security vulnerabilities, prior requirements for performing any security assessment of the web application along with the do's and don'ts of the assessment in accordance with each vulnerability. This paper also discusses various types of security testing and how VAPT is essential in every organization.

1. INTRODUCTION

Ensuring the security of using the internet and its resources is the paramount importance due to the increase in the users of this technology. Data leak can affect any organization or an individual in terms of its reputation, money and chances of loss of resources. Vulnerability in the security domain is defined as the loopholes or weaknesses present in the system or network of any organization or individual [1]. The attacker then takes advantage of this weakness and exploits using exploitation techniques. The security attacks on e-commerce platform are not new, and they are happening from starting of Web 2.0 [2]. There are more than 24 million e-commerce websites are present on the internet. According to a Forgenix survey [3, 4], 75% of e-commerce websites are at the risk of some cyber attacks. Even reputed and branded companies have vulnerabilities in their websites taking advantage of them these websites are compromised by attackers. Taking the example of reputed brands which already have hit by any type of cyber attacks is predicted to be more than 350 million dollars during the years 2018-19 [5-7]. The main area to focus to curb these attacks is systematic security testing (or penetration testing or pentesting) and vulnerability assessment.

The system can be compromised because of existing vulnerabilities. The network, application or systems consisting of these vulnerabilities are termed as a vulnerable application or network. Therefore, it is important to perform the Vulnerability Assessment and Penetration Testing (VAPT) of the web applications before releasing to the market. The process of vulnerability assessment is to find out the flaws and weaknesses existing in the system by scanning the system or the whole network. Conducting a vulnerability assessment helps to check the security posture of the organization in the cyber world. The key objectives of conducting a vulnerability assessment in the organization would be risk assessment, system accreditation, compliance checking, and network auditing and continuous monitoring.

1.1 Related work

Security testing and various related issues such as privacy and others in web applications, in general, are articulated in recent survey articles [8-12]. Devi and Kumar [13] introduced various tools such as Sparta, Network mapper (Nmap), Zenmap, Netcraft, IP Address Tracking, Virus Total, etc. for information gathering. During the practical experimentation of using security tools on a particular application, it was observed that after using OWASP ZAP tool, it was able to detect medium high as well as low level risks. The medium high-level severity vulnerabilities that were identified are URL rewriting, Application error disclosure, X-frame-options header and SQL injection. Priyanka et al. [14] presented three web application vulnerabilities such as SQL injection, XSS (Cross-site scripting) and CSRF (Cross-site request forgery). It also discusses the tools that can be used for VAPT. This paper concluded that preventive measures to consider for SQL injection attacks are input validation, white listing and sanitization of user input values, use of prepared statements to pass the user input data as parameters, and avoiding the disclosure of sensitive information through error on the web pages. Amin et al. [15] focus on how the red team performs the security assessments of the applications. There are largely three types of teams: 1) red team, 2) purple team and 3) the blue team. The red team imitates the activities that an attacker would do by making the use of tools and techniques to perform a real-world attack just like an attacker would. The blue team refers to the organization's internal security team that defends against the red team attacks as well as the internal/external attackers. The purple team is a hybrid version of the red team and blue team. The purple team consists of Blue team's defensive techniques and the attacking skills of the Red team. This paper also explained rules of engagement for a red team, some of which are execution of the required engagements, compliance with all the laws, regulations, policies and programs, implementation of the operational methodology of the team, identification of the input to the target environment, research and development of new exploit tools for testing the functionality, perform OSINT, identification and assessment of the actions revealing system vulnerabilities, providing assistance to the red team lead in the development of the final engagement report and also to perform the physical assessment under the leadership of the red team lead. Vats et al. [16] starts with discussing the various pentesting strategies like external pentesting strategy, internal pentesting strategy, blind pentesting strategy, double blind pentesting strategy and targeted pentesting strategy. The different types of pentesting are black box pentesting, gray box pentesting and White box pentesting. This paper discussed some of the common pentesting tools used by testers such as Hping, nmap, httprint, GFI LanGuard Network Security Scanner, Brutus Security tool and others.

Goutam et al. [1] proposed a framework that will give more security to the applications running under any financial institution. The proposed algorithm works in the following manner: first, the user will be required to give his login id and password to this framework. If the user details match with the already registered users, then an OTP is sent to the mobile number as well as the email id. After the OTP is verified, the reference number is asked by the application. If the reference number is not provided, then the user will be redirected back to the login page. The reference number is auto-generated when a user first registers on the application with his/her details. This reference number unique for each customer is not stored in the user details database. The proposed framework was practically tested on a financial application, and it was helpful in detecting vulnerabilities like clickjacking vulnerability (X-frame-options header not included in the HTTP response headers), cross-site scripting protection not enabled, SQL injection vulnerability and private IP disclosure which can be dangerous if accessible to any attacker as it can be used for conducting further attacks on the application. Functionalities of VAPT, testing checklist for assessment of applications, OWASP top 110 security risks and the workflow of penetration testing are presented [17]. Authors presented a checklist that should be followed while practicing secure coding consists of input validation, authentication, password management, output encoding, access control, session management, communication security, cryptographic practices, error handling and logging, system configuration, file management, database security and memory management. Khera et al. [18] performed an analysis of the lifecycle of the VAPT process. It also shortlists some useful VAPT tools for testing and finding the target system vulnerabilities. It explains the need for adoption of vulnerability assessment and penetration testing at various organizational levels to prevent any cyber attacks. It also discussed vulnerability assessment and penetration testing in detail. The main reasons why these vulnerabilities arise are system misconfiguration, weak password combinations, system connected to an unsafe network, poorly designed software and hardware. Hasan and Meva [19] discussed in detail about the VAPT process model, its benefits and the tools used in the process. The paper focuses on the high-risk vulnerabilities like SQL injection, local file inclusion, and cross-site scripting and remote file inclusion. Hasan and Meva [19] also performed a literature survey to perform an analysis of the generalized VAPT process used among all of them and the tools which are actually useful when performing the vulnerability assessment and penetration testing. Yaqoob et al. [20] presented vulnerability assessment and why it needs to perform in every organization. It also discussed common network vulnerabilities, threats to the vulnerable networks and the vulnerability management lifecycle. In addition to this, this paper presented penetration testing process and also performs a comparison with the vulnerability assessment process. Hasan et al. [21, 22] analyzed the different approaches to perform vulnerability assessment and penetration testing in web applications to ensure secure web applications in the ever evolving cyber world. First, the paper [21] discussed the software development life cycle and its phases like planning, analysis, design, implementation, testing, integration and maintenance in detailed manner. Next, the paper divides the vulnerability into two categories: logical vulnerability and technical vulnerability. Paper [22] discussed in detail about penetration testing, how it is done step wise, how it helps in securing the network and what are the tools for performing penetration testing. Penetration testing is an effective way of identifying and assessing the vulnerabilities of the system.

Haque et al. [23] presented the ways to secure the web services, the challenges faced in the security of web services and the recommendations to overcome those security challenges in web services. It also discusses the 10 most common vulnerabilities and also presented the ways to prevent three of these vulnerabilities such as SQL injection, cross site scripting, session management and broken authentication. Similarly, common vulnerabilities are discussed in the studies [24, 25]. Security controls are presented. To illustrate it, a livestock data center is used for a case study to perform the assessment and testing and to propose the relevant security controls [26]. Singh et al. [27] presented methodology of performing penetration testing. It describes what are penetration testing, its various techniques and the reasons to perform penetration testing. Goel et al. [28-33] presented VAPT lifecycle to be performed on the web application infrastructure and this procedure can be helpful in preventing cyber attacks. A study [34-40] shows that the exploring vulnerabilities depend on the type of programming environments and application specifications.

All the aforementioned studies used automated vulnerability scanning methods and tools. Alternative of the automated testing is the manual testing, which is a best option for modern applications. Further, these studies focused on discussing or presenting very limited web application vulnerabilities.

1.2 Contributions and paper organization

To this end, the main objective of this paper is to identify and provide understanding of how different vulnerabilities exist in the system, how the attackers can perform can exploit them. Papers that provide basic understanding the web application infrastructure, vulnerabilities and exploitation of the application, VAPT process flow, what are the measures the beginners of the VAPT process need to take care are very limited. Further, as the users of the web applications are increasing, new vulnerabilities come in to existence and the literature also grow in accordance with it. This paper attempt to provide all these concepts for easy understanding by the beginners and intermediate VAPT testers. Also provides do's and don'ts for the testers during VAPT process.

Section 2 presents the preliminaries for understanding the VAPT process. Section 3 presents the most commonly used open source tools for conducting VAPT process. This section also presents important add-ons and features available in popularly used Burpsuite tool for web application testing. Section 4 provides a detailed description of the VAPT process in two modes such as active mode testing and passive mode testing. Methods used in each mode are detailed in this section. Section 5 presents the measures to be considered during VAPT process. This section helps the beginners of the VAPT process as a checklist during the testing. Finally, Section 6 concludes the paper with future directions. The list of abbreviations used in this paper is provided at the end of the paper.

2. PRILIMINARIES

2.1 Vulnerability management lifecycle

The vulnerability management lifecycle consists of 6 main steps [1]: 1) discovery, 2) prioritization of assets, 3) assessment, 4) reporting, 5) remediation and 6) verification. The discovery step involves keeping a track of all the vulnerabilities existing in the application or the network on a regular basis. The second step is the prioritization of assets basically involves the categorization and assignment of values to the assets according to their priority. The third step is to create a risk profile based on importance or priority of the reported vulnerabilities. The fourth step is to measure and report the level of business threat with respect to the existing vulnerabilities in the organizational network. The next step is to perform the remediation to fix the vulnerabilities and protect the system from exploitation by the attackers. The final step is the verification to check whether the existing system vulnerabilities have been patched or not.

2.2 Penetration testing

Penetration testing is a type of testing method used by ethical hackers to perform the testing of full integrated and operational system infrastructure or network. Penetration testing is defined as a procedure to find vulnerabilities present in the target system or network infrastructure in order to take certain steps to secure the network from attackers. It helps in checking whether an attacker would be able to penetrate into an organization's network or not. This testing technique is done by an ethical hacker simulated as an unauthorized user who attacks the system or executes the penetration into the system [41, 42].

2.3 VAPT considerations

While performing the VAPT, the CIA (Confidentiality, Integrity and Availability) principles must be considered. Confidentiality refers to the principle of ensuring that only the authorized people are able to access the restricted information. No other unauthorized person should be able to access or read or edit the data in the application. The application should have proper authentication and authorization mechanism to allow only the authorized personnel's to access the data, each authorized role should be allowed with only a specific set of functionalities in the application according to their level of authorization that is, a normal user should not be able to perform the functionalities of what an admin can do. The unauthorized users should not be able to access any data present in the application. The confidentiality factor fails if the attacker is able to perform horizontal or vertical privilege escalation. Horizontal privilege escalation is a type of attack in which the users of the same level of authorization are able to access the data of another user. Vertical privilege escalation is a type of attack in which the normal users are able to access data of users of different levels of authorization, that is, any user of lower level is able to access the privilege rights or functionalities of an admin in the application [43, 44].

Integrity means ensuring the sanctity of the data when in transit. No one should be able to perform the modification of the data while it is in transit from the client to the server side of the application. To ensure the integrity of the data, data in transit should use HTTPS or the data should be in encrypted form when in transit so that no intruder in the communication can easily read the data. By encrypting the data, sender can prevent man-in-the-middle attacks like spoofing, hijacking and eavesdropping in the communication channel of a network. The last principle is to always ensure the availability of the system or the application. It should be available to its users in the network anytime they access it. Availability can be affected if there is lack of request limiters in the application. Due to lack of request limiters, an attacker can cause a DOS (denial of service) attack and halt the system or application from sending the response back to any of the requests received due to the application server receiving the request more than it can handle it [45].

To conduct a VAPT, the three areas to check the vulnerabilities are the physical structure, logical structure and the architecture of the target network. Network testing is performed by the tester to find the vulnerabilities existing in the physical design of target system.

2.4 Vulnerability assessment vs. Pentesting

The differences between Vulnerability assessment and Pentesting are as follows [35, 36]:

1. Vulnerability assessment is the process of identification and the measurement of the severity level of vulnerabilities in a system. Penetration Testing is generally a goal-oriented exercise.
2. Vulnerability Assessment holds the lists of various security vulnerabilities, often prioritized by their severity level along with the organizational criticality. Penetration Testing is primarily focused on the simulation of a real-time cyber-attack, capability testing the defense mechanisms and figuring out the ways in which a real attacker can conduct an attack on the network or application or the system instead of the identification of the vulnerabilities.
3. VA typically covers the vulnerabilities horizontally, meaning that it provides a breadth wise approach for the

security position of the application whereas penetration testing has a vertical approach, it covers the security vulnerabilities in depth. In other words, vulnerability assessment shows how big a vulnerability is and penetration Testing shows how critical it is.
4. VA can be conducted with the help of automated tools however penetration testing is generally done manually

2.5 Vulnerability assessment methodologies

There are two VA assessment methodologies: 1) automated, and 2) manual. By using automated approach, testers can identify the vulnerabilities present in the application with the help of automated tools and eliminate the false positives. Automated tools are software that interacts with the target sans human intervention. The primary benefit of automated scanners is that they reduce the labor-intensive work required to accomplish the task. Automated scanners like Acunetix will give us an overview of the possible existence of vulnerabilities in the environment. These vulnerabilities are aligned with the following industry-wide accepted standards such as OWASP, SANS, ASVS, WASC.

By using manual approach, the tester will detect every exploitable vulnerability present in the web application. The tester looks out for logical flaws which might compromise the authentication/authorization process, injection attacks, data security, input validations, session management issues, etc. The tester identifies every open port and the service running on the API servers. After that, the tester tests them for security vulnerabilities depending on their level of exploitability and availability in the environment they exist in. Final step involves the verification and validation of these vulnerabilities based on the standard benchmark. The following checks to be performed during manual testing security controls according to industry security standards like OWASP, SANS, MASVS, WASC, etc. are used for testing purposes. They are

- Every API in the application is checked against as many controls as possible by manually fuzzing with various payloads.
- During a security assessment, every open port/service configured on the in-scope asset server is rigorously tested against the respective security control.
- The tester recommended considering the references specifically from online documents/security blogs, and MITRE Common Vulnerabilities and Exposures (CVE) entries [46, 47].

2.6 Web application testing strategies

There are two web application testing strategies: 1) black box testing, and 2) grey box testing. Black box testing is a formal technique of application testing in which the examination of the application functionality without any knowledge of its internal or backend working mechanism is performed. It does not have any requirement of the past knowledge of web application or intervention of the vendor of application. Black Box penetration testing will be performed on the application along with its APIs that are interacting with the application using those APIs.

Grey box testing is a type of software/application testing that is performed by a tester having partial prior knowledge about the internal/backend mechanism of an application which is given by the owners or developers in form of walk-through of applications, application data flow, API documentation, tech stack etc. and can most often include the design and architecture documentation and internal access to the assets. The tester also gets test user credentials to assess the application/software post login functionalities. The main intention behind a Grey box assessment is to provide a more efficient & focused security assessment. This activity helps to simulate an attacker which might act as a threat to the application's sensitive data, assets and eventually to the organization's reputation.

There are two types of VAPT models [31]: 1) flaw hypothesis model and 2) attack tree model. In the first type, there is a system analysis and penetration prediction technique which consists of compilation of a list of flaws consisting of a hypothesis and performs analysis of the code specification and system documentation. In an attack tree model, the different attack methods or exploits which are possible against a target web application are represented in the form of a tree structure.

2.7 Skill set required for VAPT

Following are some common skills required for Web application assessment:

- Basic understanding of Computer Networking (TCP/IP model, OSI layers, protocols, top ports etc.)
- Basic understanding of Linux commands with hands-on practical knowledge.
- Basic understanding of programming languages like HTML, JavaScript, PHP, MySql etc.
- Good understanding of OWASP Top 10 Web Application Vulnerabilities, CVSS, CVE.
- Basic understanding of Pentesting supporting operating systems [48-51] such as kali-Linux tools, open-source and commercial tools for web application assessment.

2.8 VAPT summary

VAPT should be performed on a regular basis especially in the technology-based firms and organizations as they are more prone to the cyber based attacks. The advantages that can be achieved by performing regular Web application VAPT are

- Understanding the web application infrastructure, functionalities and classifying the assets, resources and the functionalities of the web application in accordance with their significance.
- Timely detection of the vulnerabilities in the web applications of the organizations before any attacker performs an exploitation of the application.
- Assigning value to the resources according to their significance and identify the most common web security vulnerabilities and their potential threats to each web resource using automated and manual testing techniques.
- Mitigating and performing the elimination of the critical vulnerabilities at high priority compared to the others. These vulnerabilities exist in the most important assets, resources and functionalities of the web resources.

3. OPEN SOURCE VAPT TOOLS

There are majorly four types of open source and free VAPT tools that can be useful such as 1) static analysis tools, 2) network testing tools, 3) application testing tools and 4) social Engineering tools [31]. Static analysis tools perform the VAPT by analyzing the source code of the web application. Some of the Static Analysis tools are Flawfinder, Pychecker, Pixy, RATS and OWASP SWAAT. Network Analysis tools are used for scanning the target network for analysis of the loopholes in the target network. Some network analysis tools are nmap, hping, superscan, Xprobe2, Nessus, Brutus and Metasploit. The application testing tools are used to analyze the cyber defence infrastructure of the organisation. Some of the tools used for it are Nmap, Fiddler, Nikto, WebScarab, Arachni and OWASP ZAP. Social Engineering tools are used to check the difficulty level of extraction of the information which is considered confidential, by interaction with the organization's employees. Following are some basic tools that need to be thoroughly and practically understood in order to start the assessment process [39, 52-55].

- *Burp Suite:* It is a combined and one in all platforms used for conducting the security testing of web applications. It works simultaneously together by supporting the entire testing process from its initial mapping stage to the complete vulnerability analysis of a web application [47].
- *Nmap:* It is an open source network scanner tool available for free. It is specifically used for discovering the hosts and services by sending the data packets on the computer network and analysing the responses.
- *Slap:* It is a penetration testing tool available as an in-built tool in Kali Linux operating system that enables the automation process for the detection and exploitation of the SQL injection vulnerabilities and weaknesses to take over the database servers.
- *Dirbuster:* It is an open source, multi-threaded java-based application specially designed for the purpose of brute forcing the files and directory names on the web server.
- *Nikto:* It is a free software tool primarily used as a command-line vulnerability scanner that performs the scanning of the web servers for detecting the exposed sensitive data, dangerous files/CGIs, outdated server software and other problems.
- *XSStrike:* It is an open-source tool used as a penetration testing tool whose purpose is to identify XSS (Cross Site Scripting) vulnerabilities. This tool is equipped with an automated payload generator, parsers, a powerful fuzzing engine and a crawler.
- *Corsy:* It is a python program tool available in Kali Linux which enables the scanning of all CORS misconfiguration that is present in the CORS implementations of the applications.
- *SSLscan*: It is a free penetration testing tool specifically for performing the queries on SSL services, such as HTTPS. This is done for the determination of the weak ciphers that are supported for SSL/TLS versions.
- *Tplmap*: It is a penetration testing tool specifically used for detecting and exploiting the SSTI vulnerabilities in all template engines present in the application in order to get access to the underlying file and operating system. The tool is used to test if the parameters in the URL are vulnerable or not.
- *LFISuite*: It is an open-source VAPT tool having the capability to perform scanning as well as the exploitation of the Local File Inclusion (LFI) vulnerability using different attack techniques.
- *Wireshark*: It is a tool which acts as a network protocol analyzer by capturing the packets such as from any computer or office or the internet via the network connection. It is open source software that performs the analysis real time network traffic and is considered the suitable testing tool for troubleshooting of network issues in organization. Wireshark can help in many issues like latency issues or dropped packets by troubleshooting the network using it.
- *OWASP ZAP*: OWASP ZAP (also popularly known as Zed Attack Proxy) is free web application security scanner. It helps the user to perform modification of all the network traffic passing through its proxy server including the HTTPS traffic.
- *Acunetix*: Acunetix is a tool used as an automated application security testing tool that performs the crawling and auditing of the web application that tester give by checking for the web security vulnerabilities in it like Insecure Deserialization, SQL Injection, Cross-site request forgery, Cross site scripting and other such vulnerabilities.

These tools may be downloaded and installed individually or specialized penetration testing operating systems are available with all necessary security tools. Some popular operating systems are Kali Linux [49], Parrot OS [50], Security Onion [51], etc. Burp Suite [47] extensions can help in the automation of the web application assessment process, and it also reduces the time in scanning and exploiting the web application vulnerabilities if we already have the following Burp Suite extensions installed and setup in our Burp application. The BApp Store consists of various types of Burp extensions for the detection and exploitation of different web or mobile related vulnerabilities that have been specially written by contributors of Burp Suite to expand the capabilities of the Burp application. Testers can directly install the BApps burp extensions from the extender tab present in the Burp application. Brupsuite is available in two editions: 1) community edition and 2) professional edition (Pro version). Community edition is free to use and it comes with limited functions. While the professional version require licence from the vendor and comes with many interesting features. Some of those functionalities that are useful for web application pentesting are as follows [33-35]:

- *Logger++:* Burp Suite Pro provides the functionality to be able to proxy every HTTP request and response tester put through it. It stores all the HTTP requests and responses going through the client-side proxy called Burp Suite, in an easily exported, understandable and sortable table.
- *Active Scan++:* It extends the capability of the active and passive scanning functionality of the Burp Suite Pro. This extension is specifically designed to append a decrease in the network overhead, it detects the possibility of an application to be exploitable as follows: 1) possibility of host header attacks such as password reset poisoning, web cache deception, web cache poisoning, DNS rebinding, and others. 2) possibility of host header attacks such as password reset poisoning, web cache deception, web cache poisoning, DNS rebinding, and others 3) malicious transformation of input, 4) issues related to passive-scanner that take place during the fuzzing process such as installation of 'error message checks' extension for ensuring the effectiveness, etc. 5) edge side includes, and 6) XML input handling
- *Autorize*: It is a burp extension focussed at helping the penetration testers to identify the vulnerabilities existing

in the authorization mechanism during the web application penetration test. It only requires the session cookies of a user with lower privilege rights and navigates to the webpage that highly privileged user can access. The Burp extension repeatedly sends every request with the session of the low privileged user automatically and finds the authorization vulnerabilities.

- *Backslash Powered Scanner*: This Burp extension works in a similar way like the Burp's active scanner using a novel approach which is capable of identifying and confirming both known and unknown cases of server-side injection vulnerabilities.
- *J2EEScan*: This burp extension is used for enhancing the test coverage when the penetration testing of the applications are running in the J2EE- based applications. J2EEScan performs the addition of some new test cases and strategies for discovering the various kinds of J2EE vulnerabilities such as JBoss SEAM Remote Command Execution (CVE-2010-1871), Expression Language Injection (CVE-2011-2730), Java Server Faces Local File Include (CVE-2013-3827 CVE-2011-4367) and others.
- *Retire.js*: This burp extension performs the integration of the Burp Suite Pro with Retire.js GitHub [40] repository to conduct the identification of the vulnerable Bootstrap, WordPress, and JavaScript libraries. During the passive scan, it checks the loaded configuration data and performs the detection of those which are considered vulnerable due to the various signature types such as URL, filename, file content or specific hash, etc. being used or are known to have known vulnerabilities (CVE).
- *Collaborator Everywhere*: This extension integrates in-scope proxy traffic by injecting non-invasive headers specifically designed to expose the backend systems by sending pingbacks to the Burp Collaborator client.
- *Wsdler*: This extension takes a WSDL request, performs parsing of the WSDL request to filter out the operations that are related to the targeted web service, and also performs the generation of SOAP requests that are then sent to the SOAP endpoints.
- *Java Deserialization Scanner*: This extension gives Burp Suite Pro the capability to identify the existing Java deserialization vulnerabilities in the web applications. It can perform both the active and passive scanning and can also be used in an intruder like manual mode or an exploitation mode. This extension also allows the user to perform the discovery and exploitation of Java deserialization vulnerabilities having different types of encodings (Raw, Base64, ASCII Hex, GZIP, Base64 GZIP) by inserting multiple payloads when the following libraries are loaded in the target JVM using an ysoserial jar file.

4. VAPT PROCESS FLOW

As shown in the Figure 1, the VAPT process of the web applications will be carried in two modes: 1) Passive mode, and 2) Active mode. During passive testing, the security analyst attempts to understand the logic of the web application and executes the exploration of the test application like a normal user. Various tools that can be used for the purpose of information gathering for example, an HTTP client-side proxy tool like Burp Suite Pro can be used for the purpose of observing all the incoming and outgoing HTTP requests as well as HTTP responses on the application. After this phase the analyst should know all the access points of the web application (e.g., HTTP headers, parameters, cache and cookies). The passive mode of testing will be held in the sequence of steps 1) information gathering, 2) Fingerprinting web server, 3) Web content scanning, 4) web page content review, and 5) SSL verification. Section 4.1 describes these steps, on the other hand, in the active mode of testing, the security analyst performs active tests on the web application which are categorized as follows: 1) server level testing, 2) client level tests, 3) authorization and identity management testing, 4) authentication testing, 5) input validation testing, 6) web content testing, and 7) business logic testing. Section 4.2 presents the methods of active mode in detail.

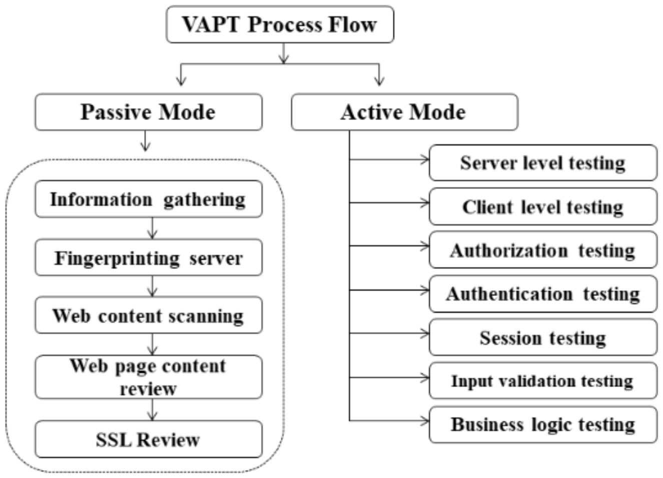

Figure 1. Modes of VAPT process

4.1 Passive mode testing

Passing mode testing consists of the following steps:

1. *Information gathering*: During this step application enumeration is performed. This is the most basic steps in which enumeration is performed such as network configuration, server configurations, open ports, network topology, network devices used, and others.
2. *Webserver fingerprinting:* It is the process of identification of the web server information like type, version, and name of the target system, application or network. It is necessary to discover the accurate web server type of the target application. It can help security analysts for the determination of whether the web applications are prone to any type of cyber-attacks. Web servers running obsolete software versions without any timely regular patching would be vulnerable to various publicly-known exploits related to the older versions. Different methods used for the process of web server fingerprinting consists of banner grabbing, HTTP responses received in response to the requests that are malformed, and with the help of automated testing tools to execute the extensive and diverse scans that make use of different types of techniques.
3. *Web content scanning:* Application servers can be properly configured to perform the enumeration of the file and directory content automatically, especially the ones that do not consist of any index page. This can help an attacker to perform quick identification of the resources of a test application. The next step is to perform the analysis and attack the application resources. Moreover, it

enhances the visibility of the sensitive data/files within the directory that should not be accessible to users, such as temp files as well as stack traces. Web Content Scanning can be performed using many open source tools like Dirb, Nikto, Dirbuster, Gobuster, WPScan (for Wordpress applications), Joomscan (for Joomla based applications) and others.

4. *Web page content review*: It is a very popular and common practice for programmers and developers to include detailed comments or metadata into their source program code. This practice can be a risk when the source code is exposed publicly, and these comments and program metadata inserted in the application's program code may expose the sensitive information existing internally. This sensitive information must not be exposed or accessible to the potential attackers. In order to check any leaked data or information, review of comments and metadata should be performed.
5. *SSL review:* If the web application is making use of HTTPS (HTTP over TLS/SSL) for inter node communication, then the tester should verify the SSL certificate, TLS/SSL version, TLS/SSL encryption enabled services, supported ciphers (weak, medium or strong) and some cryptographic flaws. This information related to SSL.TLS details can be gathered by using testssl tool or sslscan in Kali Linux [44].

4.2 Active mode testing

Figure 2 depicts the elaborated checks for each test under the active mode of VAPT. Each test of the active mode VAPT is explained in the subsections of this section [45-61].

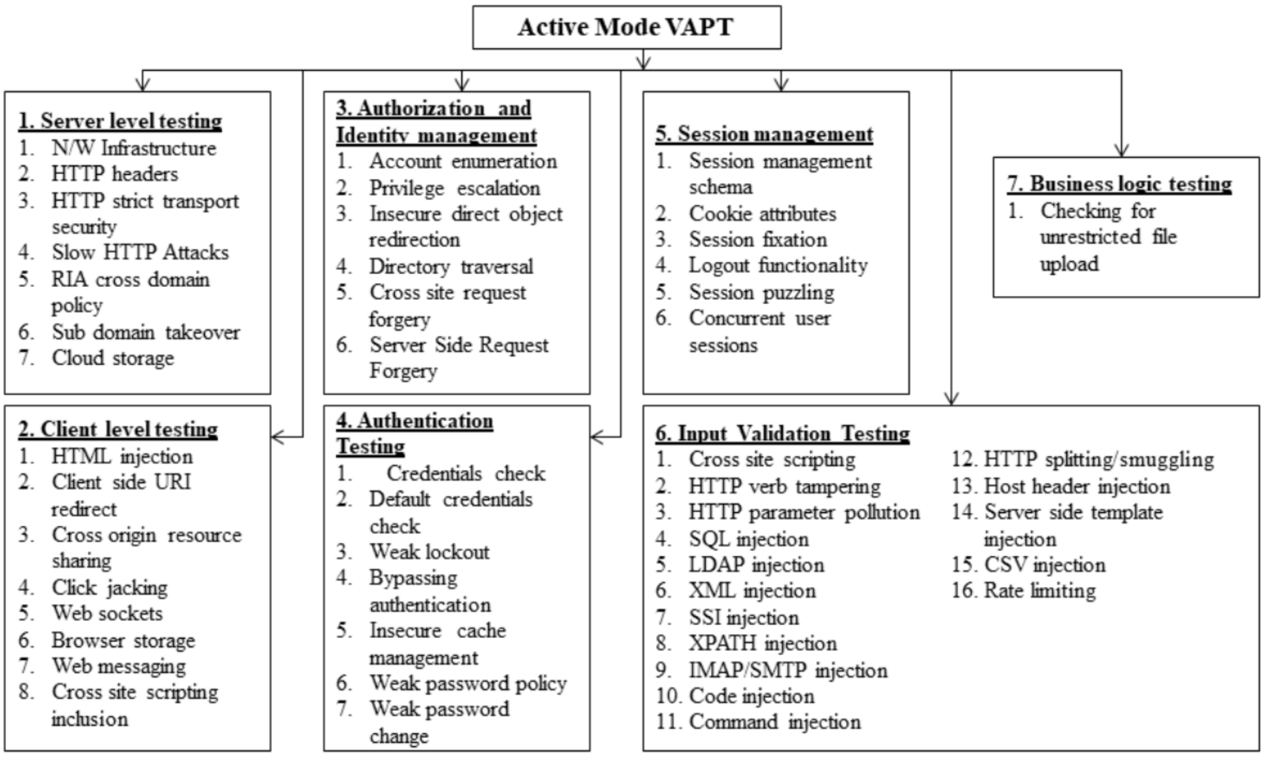

Figure 2. Categories of active mode VAPT process

4.2.1 Server level testing

It is important to perform the proper security configuration of the single nodes and elements that contribute to make up web application architecture in order to prevent any risks or mistakes that might compromise the security of the whole application architecture. The web server or application server configuration plays an essential role in the protection of the site contents, and it must be reviewed carefully in order to detect the common configuration mistakes. Checks to be conducted for server level testing are as follows

1. *Network infrastructure and configuration:* After the process of information fingerprinting of the web application, the tester must look out for known vulnerabilities or use of any components having publicly known vulnerabilities and the exploits for the different versions of software/platform/server being in use by the web application. After the identification of the presence of any mail servers on the web application, the tester should check the SPF and DMARC policies misconfiguration if it exists with the help of dig command or spoof check tool.
2. *HTTP Headers:* By intercepting the application requests in a client-side proxy tool like Burp Suite Pro, the tester will analyse the HTTP request and response headers and modify the request headers in order to see whether there is any change in the behaviour of the HTTP response headers as well as the body.
3. *HTTP Strict transport security:* The HTTP header is a security procedure that websites should have. If this is implemented in web sites, they must perform all communication means through a secure channel, that is, to the web browsers and then all the network traffic is exchanged with a given domain. The secure communication channel means that all the data will be transferred over the HTTPS connection; this will enable the protection of the unencrypted requests from any intruders. It is necessary for the tester to perform the verification regarding the web site whether it is using the HTTP strict security header or not, so that all the data that is travelling from browser to the server is in encrypted form.

4. *Slow HTTP attacks:* Slow HTTP attacks are a type of web application attack resulting in a DOS attack in which tester will send multiple requests to the application server within the specific time limit. If there is an incomplete HTTP request, or if there is low transfer rate then the server will keep its resources occupied waiting for the response from the past requests instead of responding back to the requests. When the concurrent connection limit of the web server reaches its maximum threshold, this results in a denial-of-service situation from application server. The tester can verify for Slow HTTP DoS with the help of an Application Layer DoS attack simulation tool called slowhttpattack or slowloris.
5. *RIA cross domain policy:* Rich Internet Applications (RIA) had performed the adoption of Adobe's policy files like crossdomain.xml to grant the controlled cross-domain access to the data and repair consumption with the help of technologies like Silverlight, Flashlight, Oracle Java, and Adobe Flash. Hence, a site can give access to its services from a different domain remotely. In most cases, Adobe's policy files are configured poorly for the access restrictions. For testing the RIA policy file vulnerability, the analyst should be able to perform the retrieval of the policy files. The various types of permitted permissions would be verified under the smallest level of privileges after retrieving all the RIA policy files. Necessary requests should only be received from the permissible domains and ports and requests from the overly permissive or restricted policies should be ignored.
6. *Subdomain takeover:* It is a type of web application vulnerability that appears when a corporation has configured a DNS CNAME entry for one in all its subdomains referring to an external service (e.g. Amazon cloud, Heroku, Github, Bitbucket, Desk, Squarespace and Shopify) but the service isn't any longer utilized by that organization. An attacker can purchase and register to the external service provider and consequently claim for the affected subdomain. For testing the Subdomain Takeover vulnerability, the primary step is to perform enumeration of DNS servers of the target and other resources. The tester checks if the subdomain is running/active or not after it is found that it exists. If that subdomain can be purchased, then it is considered vulnerable.
7. *Cloud storage:* Cloud storage services provide the facilitation to the web applications and services for storage and access to the objects of system existing in cloud storage services like Google cloud, AWS. Improper configuration in the access control, although may conclude in exposure of sensitive information being modified as well as data access by an unauthorized party. A common vulnerability example is a misconfigured Amazon S3 bucket; however, other services providing the cloud storage may also exposure to various types of cyber threats.

4.2.2 Client level testing

Client-Side testing is a type of testing concerned with the code execution on the client side that typically exists within the web browser or browser plug-in. The execution of code on the client-side of the web application is different from the one executing on the server side of the web application and it returns the subsequent content. Client-side security needs penetration testing to be performed because client-side attacks can quickly risk and compromise the critical data assets and information. It is essential to test the susceptibility and application network's capability to identify and respond back to the client-side attacks before the damage is irreversible. Client level testing involves checking for following:

1. *HTML Injection*: HTML injection is injection vulnerability in the web application that takes place when a user can control an input parameter and is able to perform the injection of any malicious HTML program into a web page having this vulnerability. This web application flaws/weakness takes place due to an improper input validation and sanitization and improper encoding of the output. The attacker performs an HTML injection attack to send an arbitrary HTML page with malicious intentions to the victim.
2. *Client side URI redirection*: Client-side URL redirection is additionally referred to as open redirection. This type of weakness or vulnerability exists in an application when it allows all types of untrusted input consisting of an external URL and doesn't perform any proper sanitization. The inserted URL could result in redirection of the user to a different page from the application, like a malicious web page created and then controlled by an attacker.
3. *Cross origin resource sharing*: CORS could also be a mechanism that enables an internet browser to perform cross-domain requests of the XMLHttpRequest L2 API in a very controlled manner. The XMLHttpRequest L1 API solely allows requests to be sent inside the identical origin as a result of it being restricted by the identical origin policy. Cross-origin requests have an associated origin header that identifies the domain initiating the request and is usually sent to the server. CORS defines the protocol to use between a web browser and a server to figure out whether or not a cross-origin request is allowed. Access-Control-Allow-Origin could also be a response header utilized by a server to purpose that domain square measure allowed to browse the response. The analyst ought to rummage around for insecure configurations as for example using a wildcard (* symbol) as worth of the Access-Control-Allow-Origin header which implies all domains square measure allowed. Another insecure example is once the server returns the origin header with none further checks, which could cause access of sensitive information. Access-Control-Allow-Credential may well be a vicinity of a pre-flight request indicating that the last word request will embody user credentials.
4. *Click jacking*: It comes under the UI redressing method, it is a malicious and complex attacking method whereby an internet user is mislead to click on a particular link which redirects to a webpage which looks similar what the user is expecting with but this is eventually an attacker page having the actual website as a part of its frame. The tester tests this vulnerability by investigating whether it is possible to load the target website in an inline frame of a sample test html page.
5. *Webs sockets*: Web sockets works by enabling full-duplex communication channel between the web client or web server allowing the client and server to ensure asynchronous communication. It is the server's function to perform verification of the Origin header in the Web socket handshake of HTTP communication. Under this
    - The tester should first perform the identification of whether the application is using Web Sockets by inspecting the code on the client-side for ws:// or wss:// URI scheme.

- In the Burp suite pro, with the help of a Web socket client, try to connect to the remote Web Socket server. If we observe the establishment of a successful connection then it means that the web server would not be able to perform the validation of the origin header.
- Verify that Web Socket connection is making use of secure socket layer (SSL) to transmit sensitive information with wss://

6. *Browser storage*: Web browsers allow the functionality for storage functionality in the client-side for developers to perform the storage and the retrieval of data such as local storage, session storage, IndexedDB, cookies, etc. Testing should be conducted to work out whether the web site is performing the storage of the data that is considered too sensitive to be stored at the client side. The assessment of storage object code handling should be done in order to determine any injection attacks in future.
7. *Web messaging*: Web Messaging (also noted as Cross Document Messaging) allows the web applications that are running on multiple domains to intercommunicate with each other in a secure manner. Before this was introduced, the intercommunication of different origins (between iframes, tabs and windows) was enforced restrictions under the same origin policy and eventually was checked by the web browser. After this, Cross Document Messaging was originated and was implemented altogether with multiple browsers. It ensures the secure communications between origins across iframes, windows and tabs are ensured with the assistance of Cross Document Messaging. The messaging API introduced the postMessage() method, with which plain-text messages are going to be sent cross-origin. It contains two input parameters: 1) message, and 2) domain. The checks to be made in this are
   - The tester should verify whether the application code is performing the sanitization and processing of the messages from only the acceptable and trusted domains. Within the sending domain, also ensure that the receiving domain is explicitly stated, which isn't used because of the second argument of postMessage().
   - JavaScript code is recommended to be interpreted by tester to perform the determination of the implementation of web messaging and specifically, testers should have an interest about how the target website is forming a restriction on the messages that are coming from unknown domains, and thus the method in which the information is processed when they are received from trusted domains.
8. *Cross site script inclusion*: Cross-Site Script Inclusion (also popularly called XSSI) is a type of vulnerability that results in the leakage of sensitive information across origin or cross-domain boundaries. XSSI could be a client-side attack almost like Cross-Site Request Forgery (CSRF) but includes a different purpose. CSRF uses the authenticated user to perform the execution of state-changing functions inside a victim's page (e.g. transfer money to the attacker's account, modify privileges, reset the password, etc.), XSSI instead uses JavaScript on the client-side to leak sensitive data from authenticated sessions. to check for XSSI:
   - Identify of the endpoints accountable for sending sensitive data, what parameters are required, and identification of all relevant dynamically and statically generated JavaScript responses using authenticated user sessions.
   - Determine whether the sensitive data are often leaked using JavaScript via Global Variables, Global function parameters, JavaScript Runtime errors, or Prototype chaining using this.

4.2.3 Authorization testing

Authorization is the concept or process of allowing access to the restricted resources to only those users that are permitted to use them. For testing authorization settings or configuration in the web application, tester first needs to understand how the authorization process works, and then use that information to plan for the attack on the application using the weak configuration of the authorization mechanism. Authorization is a concept that comes after a successful authentication process, so the analyst will perform the verification of this point after he has the valid user credentials, related to a well-defined set of roles and privilege rights. During the testing of this security control, it should be checked if it is possible to perform the authorization schema bypass, find vulnerability in the path traversal, or tester can also find different ways to conduct the escalation of the privilege rights assigned to the analyst. The tests for checking the Authorization and Identity management are given below.

1. *Account enumeration:* Often web applications reveal when a username exists on a system, either as a consequence of misconfiguration or as a design decision. For a case, sometimes, after we submit wrong credentials, we receive a message that states that either the username is present on the system or the provided password as wrong. The data obtained is employed by an attacker to realize an inventory of users on the system. This information is often used to attack the web application, for instance, through a brute force or default credentials attack. The tester should interact with the authentication mechanism of the application to grasp if sending a particular request causes the application to answer in several manners. This issue exists because the data released from an online application or web server when the user provides a legitimate username is different than after they use an invalid one.
2. *Privilege escalation:* It takes place when a user can access many other restricted resources or restricted functionalities than they're normally allowed to access, and such elevation or changes should be prevented by the web application. In every portion of the applying where a user can create data/information within the database (e.g., making a payment, adding or removing a contact, or sending a message), can receive information (account statement, order details, etc.), or delete information (delete or drop users, delete messages), it is necessary to record the functionality. The tester should attempt to access such functions as another user so as to verify if it's possible to access a function that ought to not be permitted by the user's role/privilege (but can be permitted as another user). If the tester is able to access such functionalities with a user with the identical user role then it becomes horizontal privilege escalation. If the functions accessed by the analyst as a traditional user belong to a better user role, then it becomes vertical privilege escalation.
3. *Insecure direct object reference:* Insecure direct object

references vulnerability takes place when an application provides direct access to things supported user-supplied input. If this vulnerability exists in the web application, then attackers can bypass authorization and access resources within the system directly, for e.g. database records or files. To test for this type of vulnerability the tester must first plan all locations within the application where user input is employed to reference objects directly. For instance, locations where user input is employed to access a database row, a file, application pages and more. Next, the analyst should modify or change the parameter value tied to reference objects and assess whether it is possible to retrieve objects belonging to other users or otherwise bypass authorization. As shown in Figure 3, consider the sample request at a link http://foo.bar/somepage?invoice=12345, in this case, the worth of the invoice parameter is employed as an index in an invoices table within the database. The analyst should change the worth of invoice and check if the corresponding invoice number details are displayed within the application or not.

4. *Directory traversal and file inclusion:* Web servers and web applications implement authentication mechanisms to regulate access to files and resources. Web applications use server-side scripts to incorporate different types of files and manage images, templates, load static texts, etc. But there also are security vulnerabilities if these input parameters aren't correctly validated. In order to see which part of the web application is liable to input validation bypassing, the analyst must enumerate all parts of the application that accept content from the user. Few checks which will be performed here are:
   - If there is any request parameters used for file related operations.
   - If there are any interesting variables

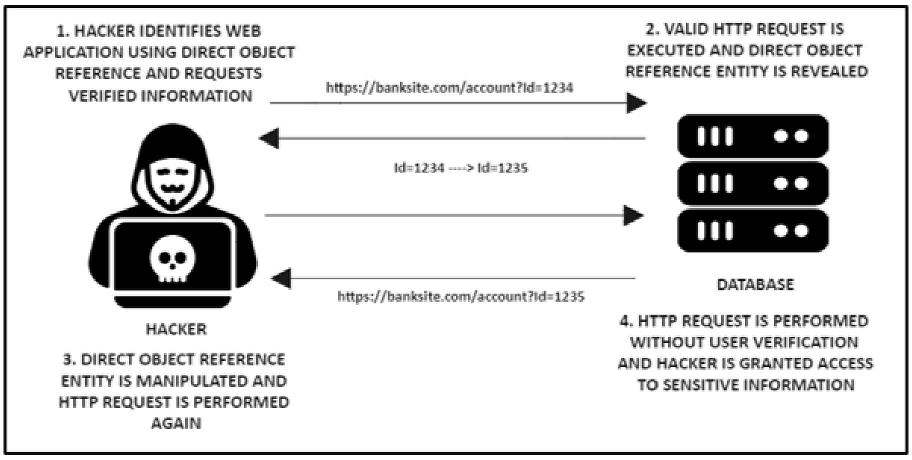
1. HACKER IDENTIFIES WEB APPLICATION USING DIRECT OBJECT REFERENCE AND REQUESTS VERIFIED INFORMATION
2. VALID HTTP REQUEST IS EXECUTED AND DIRECT OBJECT REFERENCE ENTITY IS REVEALED
https://banksite.com/account?Id=1234
Id=1234 ----> Id=1235
https://banksite.com/account?Id=1235
DATABASE
HACKER
3. DIRECT OBJECT REFERENCE ENTITY IS MANIPULATED AND HTTP REQUEST IS PERFORMED AGAIN
4. HTTP REQUEST IS PERFORMED WITHOUT USER VERIFICATION AND HACKER IS GRANTED ACCESS TO SENSITIVE INFORMATION

Figure 3. Insecure direct object reference (IDOR) attack

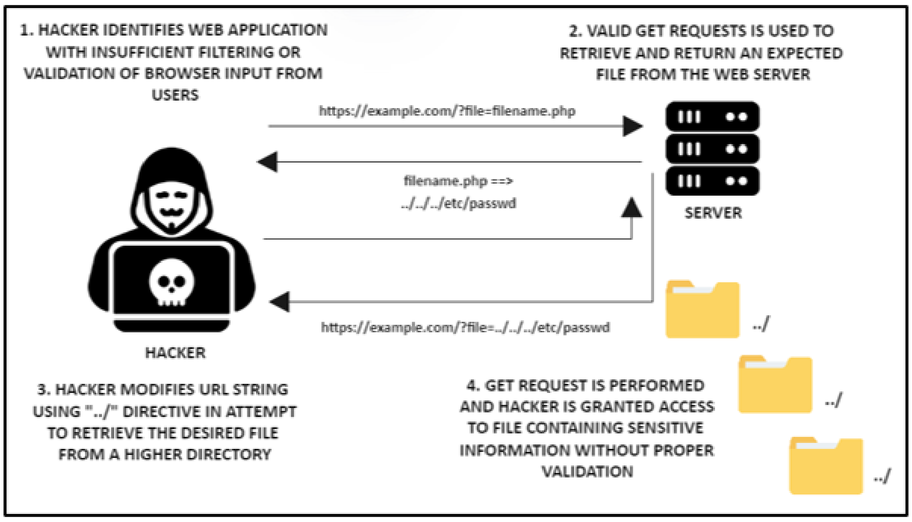
1. HACKER IDENTIFIES WEB APPLICATION WITH INSUFFICIENT FILTERING OR VALIDATION OF BROWSER INPUT FROM USERS
2. VALID GET REQUESTS IS USED TO RETRIEVE AND RETURN AN EXPECTED FILE FROM THE WEB SERVER
https://example.com/?file=filename.php
filename.php ==> ../../../etc/passwd
SERVER
https://example.com/?file=../../../etc/passwd
../
../
../
HACKER
3. HACKER MODIFIES URL STRING USING "../" DIRECTIVE IN ATTEMPT TO RETRIEVE THE DESIRED FILE FROM A HIGHER DIRECTORY
4. GET REQUEST IS PERFORMED AND HACKER IS GRANTED ACCESS TO FILE CONTAINING SENSITIVE INFORMATION WITHOUT PROPER VALIDATION

Figure 4. Directory traversal attack

As shown in Figure 4, the next stage of testing is analyzing the input validation functions present within the web application. To successfully test for this flaw, the analyst has to have knowledge of the system being tested and also the location of the files being requested. A couple of examples for the following scenario are as follows:

http://example.com/getUserProfile.jsp?item=../../../../etc/passwd

http://example.com/index.php?file=http://localhost:8080

5. *Cross-site request forgery:* As shown in the Figure 5, Cross-Site Request Forgery (also popularly known as CSRF) is a type of web security attack that forces a user to execute unintended actions on an internet application within which they are currently authenticated. In order to test for CSRF vulnerability within the web application, audit the web application to establish if its session

management is vulnerable. If session management relies only on client side values (information available to the browser), then the web application is vulnerable. Client-side values refer to cookies and HTTP authentication credentials (Basic Authentication and other sorts of HTTP authentication; application-level authentication). To perform the exploitation of the CSRF vulnerability, the tester then searches for HTML forms within the application and makes an HTML web page with some hidden fields and hosts it on an area server. Submit such WebPages after logging web application. If the page submits the malicious request successfully, CSRF is exploited.

6. *Server side request forgery:* As shown in the Figure 6, Server-side request forgery (SSRF) is a type of web security vulnerability that enables an attacker to mislead the server-side application or web server to form a call back connection to itself and in other cases also to the other web-based services within the organization's infrastructure or to an external third-party system. Manual detection of the Server-Side Request Forgery vulnerability consists of making a careful analysis of the HTTP Requests, taking the inputs parameters and headers whose input values are whole or partial URL references to other internal web resources within or external to the web application.

4.2.4 Authentication testing

Authentication is the mechanism of attempting to perform the verification of the digital identity of the sender in the communication channel. A commonly used example of the authentication process is the login mechanism. For testing the authentication schema, tester need to first understand how the authentication process works and use that information to attack the authentication process considering its flaws and weaknesses. However, most of the web applications have a requirement to perform the authentication process to gain access rights to any sensitive or private information or to execute the tasks therefore not every authentication method is able to provide suitable security to the application. For performing the authentication testing, checks to be conducted are as follows:

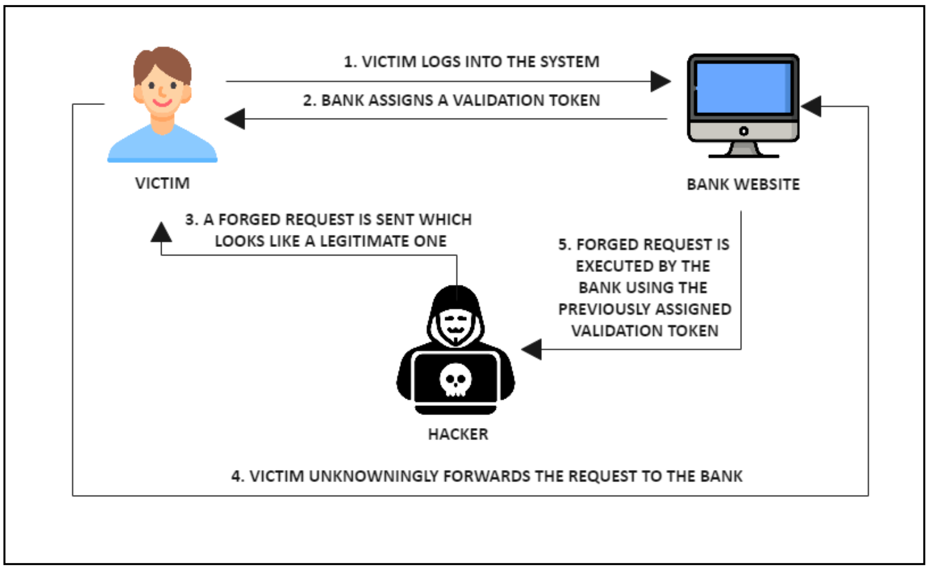
1. VICTIM LOGS INTO THE SYSTEM
2. BANK ASSIGNS A VALIDATION TOKEN
VICTIM
BANK WEBSITE
3. A FORGED REQUEST IS SENT WHICH LOOKS LIKE A LEGITIMATE ONE
5. FORGED REQUEST IS EXECUTED BY THE BANK USING THE PREVIOUSLY ASSIGNED VALIDATION TOKEN
HACKER
4. VICTIM UNKNOWNINGLY FORWARDS THE REQUEST TO THE BANK

Figure 5. Cross site request forgery

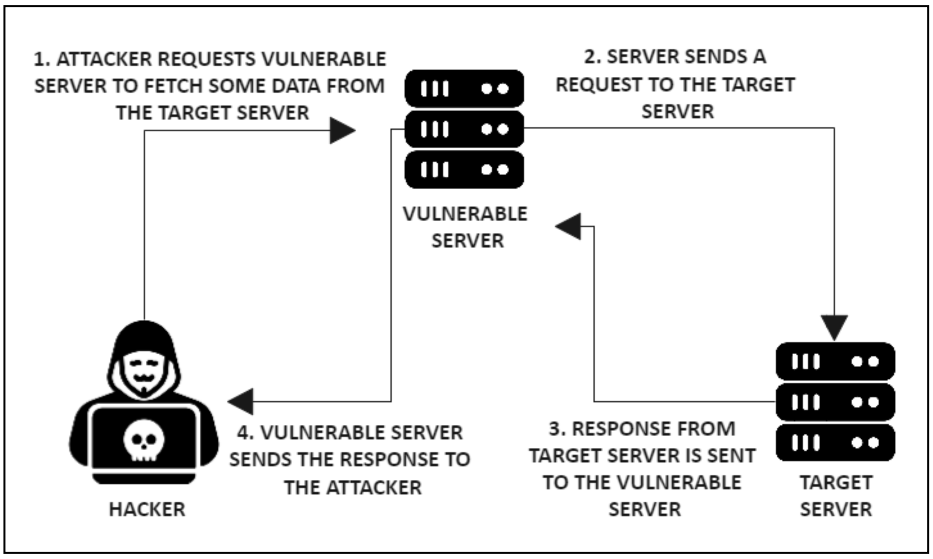
1. ATTACKER REQUESTS VULNERABLE SERVER TO FETCH SOME DATA FROM THE TARGET SERVER
2. SERVER SENDS A REQUEST TO THE TARGET SERVER
VULNERABLE SERVER
HACKER
4. VULNERABLE SERVER SENDS THE RESPONSE TO THE ATTACKER
3. RESPONSE FROM TARGET SERVER IS SENT TO THE VULNERABLE SERVER
TARGET SERVER

Figure 6. Server-side request forgery (SSRF) attack

1. *Credentials over an encrypted channel:* Testing for credentials in transit refers to the verification of the user's authentication data being transmitted via an encrypted channel (SSL/TLS) to avoid the data from being intercepted by any intruders or malicious users. The analysis focuses simply on trying to know if the information travels unencrypted from the web browser to the web server, or if the web application takes the acceptable and required security measures by employing a protocol like HTTPS. The fact that the network traffic is in encrypted form does not necessarily mean that it is completely safe.
2. *Default Credentials:* Web applications often make use of popular and free or commercial software which has the capability to be installed on web servers with minimal configuration or any customization by the web server administrator. This software even has default user credentials for initial authentication and configuration which are not changed by the application admins or owners. After gathering enough information about it, the tester can find and explore for administrative/login portals and try brute forcing them with default user credentials considering the exact software along with its version.
3. *Weak logout mechanism:* Account lockout mechanisms are accustomed to preventing and mitigate the brute force password guessing attacks. Accounts are typically locked after 4 to 5 unsuccessful attempts for login and might only be unlocked after a determined time slot with the help of a self-service unlocks mechanism, or intervention by an administrator. To assess the capability of the account lockout mechanism to prevent brute force password guessing, try to enter invalid login credentials by using the incorrect password innumerable times, before using the valid password to verify that the account was locked out.
4. *Bypassing authentication schema*: For testing the authentication schema, you might require understanding or prior knowledge of how the authentication mechanism works and using that procedure information to bypass the authentication process. There are several methods of bypassing the authentication schema that is employed by a web application that should be tested:
   - *Direct Page Request (Forced Browsing):* If an online application implements access control only on the login page, the authentication schema is also bypassed. As an example, if a user directly requests a singular page via forced browsing, that page may not check the credentials of the user before granting access.
   - *Parameter modification:* Another problem related to authentication design is when the web application verifies a successful login on the premise of a group value parameter. A user could modify these parameters to comprehend access to protected areas without providing valid credentials.
   - *Session ID Prediction:* Most commonly, web applications make use of session identifiers or session IDs to manage the authentication process. Therefore, if session ID generation is predictable, a malicious user can be ready to find a legitimate session ID and gain unauthorized access to the application, impersonating a previously authenticated user.
5. *Insecure cache management*: Testers need to make sure the testing web application doesn't leak any sensitive information into the browser cache. To do this, they'll certify for each page that contains sensitive information the server instructs the browser to not cache any data. Such a directive should be issued within the HTTP response headers with the subsequent directives:

Cache-Control: no-cache, no-store
Expires: 0
Pragma: no-cache

These directives are generally robust, although additional flags for Cache-Control are also necessary such as:

Cache-Control: must-revalidate, max-age=0, s-maxage=0

6. *Weak password policy:* Testers must check for the following to evaluate the health of the password policy such as:
   - Password complexity
   - Minimum length of password
   - Whether users can set the new password as the old one
   - Whether users are forced to change password after password recovery when password is sent in plaintext over Email/SMS
7. *Weak password change*: The forgot password and alter password functionalities should be tested for the following:
   - Check if the change password functionality asks for current password or not. If it doesn't, there's a prospect of CSRF attack.
   - Check if the users can manipulate or subvert the password change or reset process to vary or reset the password of another user or administrator.
   - Check if the password reset tool shows you the password; this offers the attacker the flexibility to log into the account, and unless the application provides information about the last log within the victim wouldn't know that their account has been compromised.

4.2.5 Session testing

Web applications consist of implementations of different processes and mechanisms for the storage and validation of the user credentials for a predetermined time. This type of mechanism is called Session Management. For testing the session management mechanism, the tester needs to check whether the session cookies and session tokens are being created in a secure and unpredictable manner. During the testing of session management functionality, the checks to be conducted are as follows:

1. *Session management schema*: The analysis of the session ID variables should be performed to determine presence of easy to guess or predictable data patterns. Session ID analysis could also be manually executed or by using Burp Suite to conclude the similar data patterns within the Session ID content. Manual checks should include comparisons of Session IDs issued for the identical login conditions – e.g., the identical username, password, and IP address.
2. Cookie attributes: Cookie is a small text that resides on clients' desk. The attributes of a cookie to be tested are:
   - *Secure attribute*: It is a cookie attribute that instructs online browser to forward the browser cookie considering that the HTTP request is being sent over HTTPS (HTTPS over SSL). This will prevent the browser cookies from being passed as requests in an unencrypted form. Secure attribute of a cookie should be set to true.

- *HTTP Only attribute*: This attribute is employed for preventing attacks like session details exposure since it does not allow the access to a cookie via JavaScript which is a client side script. The HTTP Only attribute of a cookie should be set to True.
- *Domain attribute*: This attribute is employed for verifying the domain of the cookie against the server's domain to ensure that the HTTP request is created.
- *Path attribute:* It plays a significant role in the scope setting of the cookies in relation to the specific domain. In addition to this, the path of the URL that the cookie is valid for is to be mentioned explicitly. The cookies will be sent inside the request if the path and the domain are matched.
- *Expires attribute:* The Expires tag attribute is used for:
  - setting persistent cookies
  - restrict the time span if the session exists for a long span
  - setting the browser cookie to a past date in order that it gets deleted fast

3. *Session fixation*: Session Fixation vulnerability occurs when a web application performs the user authentication without performing the validation of the prevailing session identifier, henceforth continuing with the use of the same session identifier already assigned to the user in the previous session. The tester should make sure that a replacement session ID is issued upon a successful authentication.
4. *Logout functionality*: Session termination may be a crucial part of the session lifecycle. A secure session termination requires a minimum of these next components:
   - *Logout UI (User Interface)*: Verification of the functionality of the sign off option within program. For the testing purpose, the tester should view every web page from the perspective of a user who wants to sign off from web application.
   - *Session Timeout*: Analyst should verify whether the application logs out a user due to lack of activity in the web application for a specific period of time, verifying that it should be considered unacceptable to reuse same session and also to ensure that no data gets stored in the web browser cache. The proper value of the session timeout is dependent on the purpose of web application and that a balance of security and purposefulness is maintained.
   - *Server Side Session Termination*: Primarily, the cookie values should be stored in the browser that is accustomed for session identification. Trigger the exit function and analyse the application's response behaviour, specifically in the case of session cookies. Observe and then navigate through the pages that are only visible in an authenticated user's session. It should be ensured that no data should be accessible by the unauthenticated users which are only permissible to be viewed and accessed by an authenticated user.
5. *Session puzzling*: This type of vulnerability takes place in an application which makes use of the same session cookie for multiple sessions. The attacker can easily access the web pages in an order not predictable by the application developers so that the session variable is ready in one scenario then utilized in another scenario. This type of vulnerability is identified and then taken advantage of by the enumeration of all session variables employed by the web application and in which situation they are considered valid.
6. *Concurrent user sessions*: It is considered suitable to recommend the applications for having user functionalities that enable the real-time verification of active sessions, monitoring and alerting the original users regarding the concurrent login sessions and also provide a facility for terminating the sessions remotely and manually, also there should be tracking functionality of the account activity record by keeping a record of many client details such as IP address, user-agent, login date and time, idle time, etc. The analyst can check the presence of concurrent user sessions by logging in to the identical user account from different browsers.

4.2.6 Input validation testing

Input validation, also popularly known as data validation, is the testing technique done on any user-supplied input parameter on the web application. Input validation should mitigate any improper or inaccurately formed data from being inserted or stored into the application's information system. Because it is typical to identify any malicious user who is trying to attack software, web applications should verify and perform the validation of all the input that is entered into the web application. Input validation mechanism should take place when input data is received from an external third party and especially when the data is coming from an untrusted data source. Injection attacks, memory leakage, and compromised systems can be caused due to improper input validation. The following are the checks to be conducted during input validation testing.

1. Cross site scripting: There are mainly 3 types of Cross Site Scripting which are as follows:
   - *Reflected XSS*: As shown in Figure 7, reflected XSS is a cross-site scripting attack which is very commonly found in web applications. In the case of website application that has the reflected xss vulnerability, it will allow the passing of the input data without any validation, and hence it will directly be sent to the client via the requests. Attacker's script or exploit code is most commonly found in Javascript, VBscript, and ActionScript. Tester first verifies each insertion point or input vector to detect any potential XSS vulnerabilities. For detecting this type of vulnerability, the tester will intend to make use of a specially generated user input into every insertion point (input field) that is present in the application.
   - *Stored XSS*: As shown in Figure 8, Stored Cross-site Scripting (XSS) is the most dangerous of all XSS attacks. It is necessary for a web application to be vulnerable to stored cross site scripting attack to be able to allow users to store data. For testing the stored XSS, the tester needs to perform the identification of all insertion points for the user-supplied input data that are being stored into the back-end server of the web application and then they are being displayed by the web application. Followed by this, the attacker then inserts a specially crafted malicious javascript input vector into all the identified insertion data points.

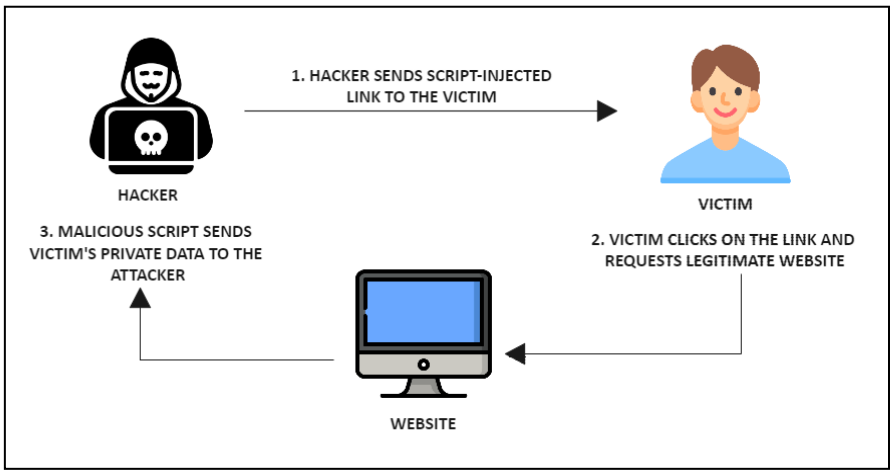
1. HACKER SENDS SCRIPT-INJECTED LINK TO THE VICTIM
HACKER
VICTIM
3. MALICIOUS SCRIPT SENDS VICTIM'S PRIVATE DATA TO THE ATTACKER
2. VICTIM CLICKS ON THE LINK AND REQUESTS LEGITIMATE WEBSITE
WEBSITE

Figure 7. Reflected XSS attack

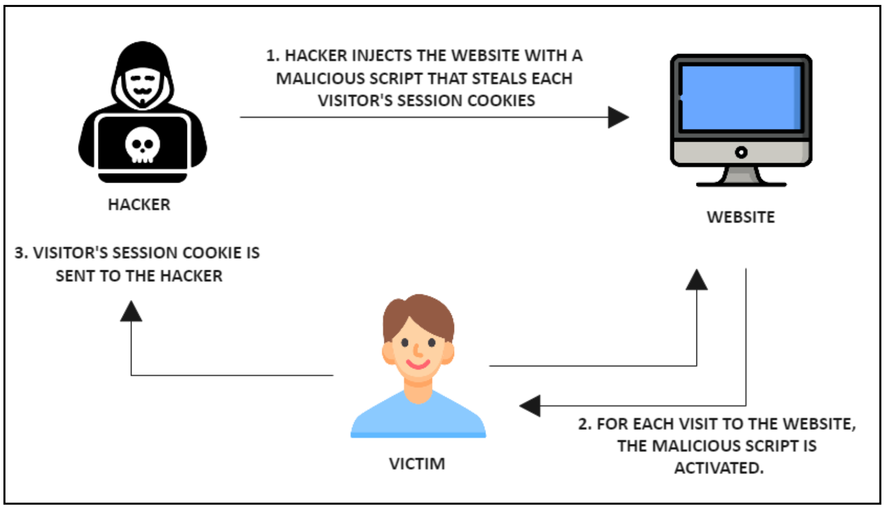
1. HACKER INJECTS THE WEBSITE WITH A MALICIOUS SCRIPT THAT STEALS EACH VISITOR'S SESSION COOKIES
HACKER
WEBSITE
3. VISITOR'S SESSION COOKIE IS SENT TO THE HACKER
2. FOR EACH VISIT TO THE WEBSITE, THE MALICIOUS SCRIPT IS ACTIVATED.
VICTIM

Figure 8. Stored cross-site scripting attack

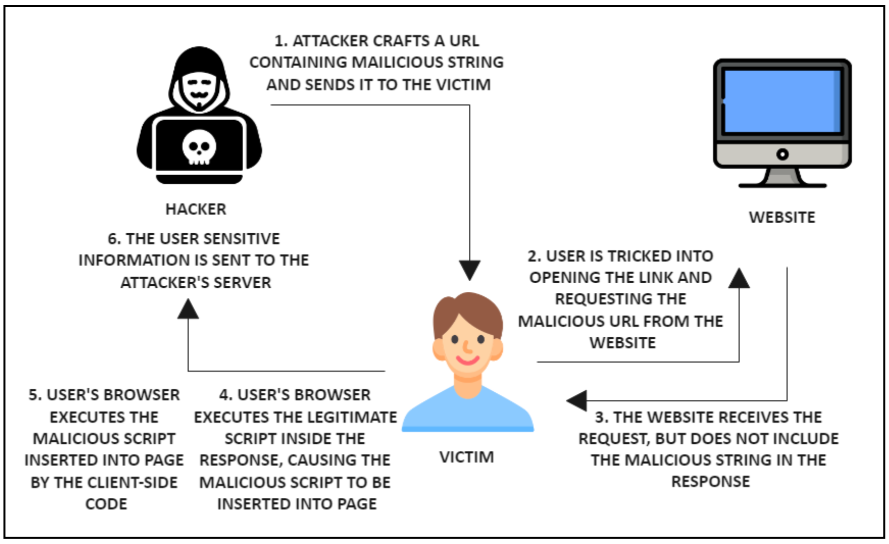
1. ATTACKER CRAFTS A URL CONTAINING MAILICIOUS STRING AND SENDS IT TO THE VICTIM
HACKER
WEBSITE
6. THE USER SENSITIVE INFORMATION IS SENT TO THE ATTACKER'S SERVER
2. USER IS TRICKED INTO OPENING THE LINK AND REQUESTING THE MALICIOUS URL FROM THE WEBSITE
5. USER'S BROWSER EXECUTES THE MALICIOUS SCRIPT INSERTED INTO PAGE BY THE CLIENT-SIDE CODE
4. USER'S BROWSER EXECUTES THE LEGITIMATE SCRIPT INSIDE THE RESPONSE, CAUSING THE MALICIOUS SCRIPT TO BE INSERTED INTO PAGE
VICTIM
3. THE WEBSITE RECEIVES THE REQUEST, BUT DOES NOT INCLUDE THE MALICIOUS STRING IN THE RESPONSE

Figure 9. DOM based cross site scripting attack

- *DOM-based XSS*: As shown in Figure 9, DOM-based cross-site scripting is a type of cross-site scripting attack related to the DOM (Document Object Model). This is a type of XSS vulnerability that is a result of the content on the dynamic browser-side of the javascript page taking the user data as input from the browser application and then doing malicious activities with it which might harm or affect the users data due to the execution of JavaScript code that has been injected into the application. The DOM based XSS vulnerability exists in a web application when the flow control of the program code is done by using the DOM elements along with an exploit script by the attacker to perform the modification of the functionality of the application or in other words, the flow control of the application.

2. *HTTP verb tampering*: For testing the HTTP Verb tampering vulnerability, the tester performs the analysis of the HTTP response from the web application to a variety of request methods for accessing the system objects. The tester should test this vulnerability by accessing all of these system objects with all the possible HTTP request methods for every single object that were identified during the web application's spidering process. If the server at the web application allows the HTTP request method other than POST or GET request method, the test comes out to be a fail or the application is considered safe from HTTP verb tampering, considering that the test web application does not allow the other HTTP request methods. The only remediation is to disable the HTTP request methods other than the GET and POST request methods to the web application servers.
3. *HTTP parameter pollution*: To test this vulnerability, the tester tests by checking the web application's HTTP response after receiving many HTTP parameters under the same request method or same name; let us consider an example in which the parameter userid is inserted in the request parameters twice. Appending many HTTP parameters with the same name may result in the wrong interpretation of values by the application. With the help of these vulnerabilities to conduct the exploitation, the tester will be able to cause functional modifications or errors in the internal parameters to bypass the mechanism of input validation in the web application. The tester first has to perform the identification of any form action that allows the input of any invalidated user input data.
4. *SQL injection*: As shown in Figure 10, to perform the testing of SQL injection vulnerability, the tester first verifies whether they can inject any input into the web application in order to perform the execution of a user-controlled SQL query in the web application's database. The tester first checks if the application takes user input without proper input validation and inserts it into the SQL queries directly. For an SQL Injection attack to be successful, it is necessary for an attacker to create an SQL Query which is syntactically correct to fulfil his malicious intentions. If an error message is returned by the web application after the insertion of into the user supplied input, generating a message stating an incorrect query, then it might help the attacker to craft and modify the logic of the existing application query. It helps the attackers to perform the injection attack successfully. There are a variety of methods that exist to perform the exploitation of SQL injection vulnerability in the web applications such as Union Operator, Blind SQL injection, Error based injection, Boolean conditions based injection, Out-of-band injection, and Time delay based injection.

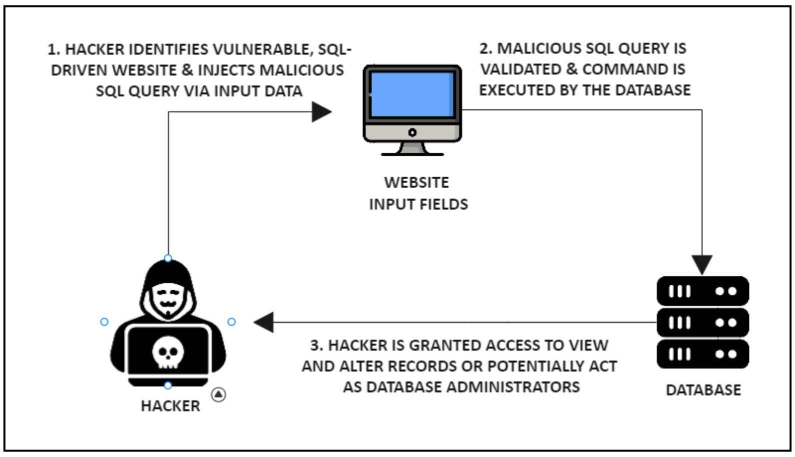
1. HACKER IDENTIFIES VULNERABLE, SQL-DRIVEN WEBSITE & INJECTS MALICIOUS SQL QUERY VIA INPUT DATA
2. MALICIOUS SQL QUERY IS VALIDATED & COMMAND IS EXECUTED BY THE DATABASE
WEBSITE INPUT FIELDS
3. HACKER IS GRANTED ACCESS TO VIEW AND ALTER RECORDS OR POTENTIALLY ACT AS DATABASE ADMINISTRATORS
HACKER
DATABASE

Figure 10. SQL injection

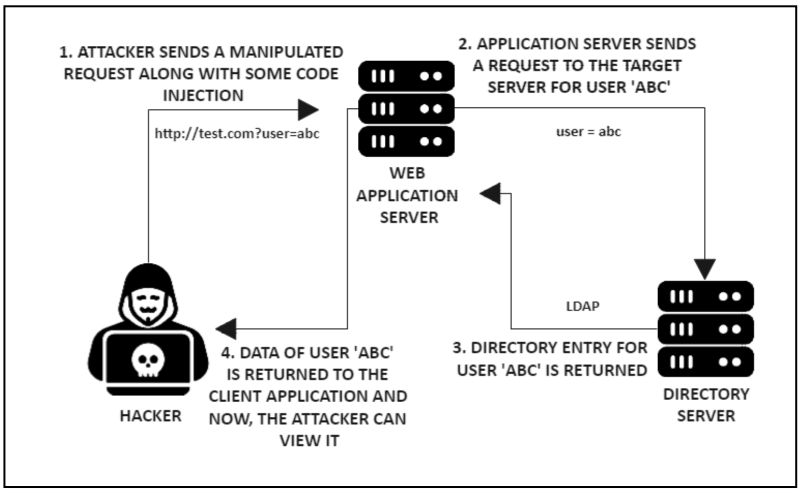
1. ATTACKER SENDS A MANIPULATED REQUEST ALONG WITH SOME CODE INJECTION
http://test.com?user=abc
2. APPLICATION SERVER SENDS A REQUEST TO THE TARGET SERVER FOR USER 'ABC'
user = abc
WEB APPLICATION SERVER
LDAP
3. DIRECTORY ENTRY FOR USER 'ABC' IS RETURNED
DIRECTORY SERVER
4. DATA OF USER 'ABC' IS RETURNED TO THE CLIENT APPLICATION AND NOW, THE ATTACKER CAN VIEW IT
HACKER

Figure 11. LDAP injection

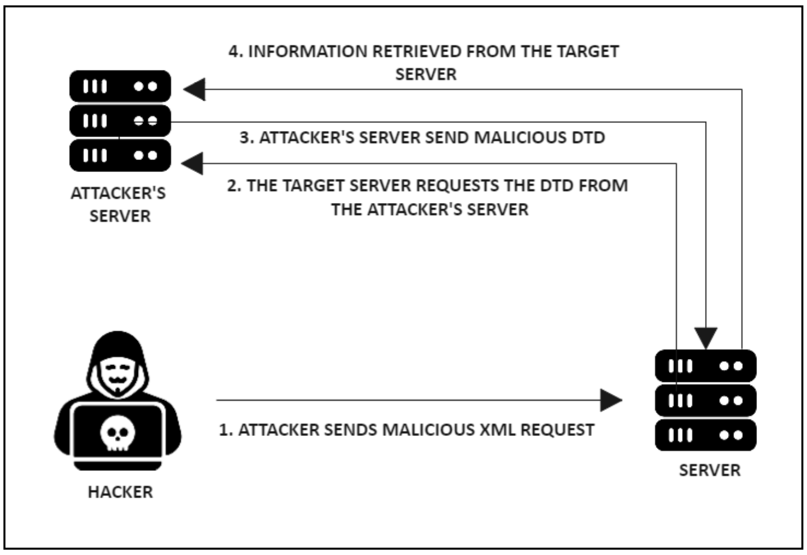
4. INFORMATION RETRIEVED FROM THE TARGET SERVER
3. ATTACKER'S SERVER SEND MALICIOUS DTD
2. THE TARGET SERVER REQUESTS THE DTD FROM THE ATTACKER'S SERVER
ATTACKER'S SERVER
1. ATTACKER SENDS MALICIOUS XML REQUEST
HACKER
SERVER

Figure 12. XXE injection

5. *LDAP injection*: As shown in the Figure 11, LDAP injection is a type of vulnerability which exists in the server side, and it is also popularly known as the server side attack, which allows the disclosure, modification and insertion of the sensitive information and data about the users depicted in an LDAP tree structure. This is performed by the manipulation of the input data parameters which is then forwarded to the addition, internal search and data modification functions. If a web application makes use of an LDAP server to perform the verification of the user credentials in the authentication mechanism and if it is having LDAP injection vulnerability, then the tester, we can conduct the bypass of the authentication check mechanism by the injection of an always true LDAP query into the existing query similarly like SQL injection, XML injection, X-PATH injection.
6. *XML injection*: XML Injection is a type of injection attack which is done by an attacker by injecting any XML (Extensive Mark-up Language) document into the application and checks for process by application. If the XML parser fails to perform the semantic verification of the input data then the application will send back a positive response and results in to successful attack. The tester first checks for the presence of a XML Injection vulnerability in the web application by inserting the XML meta characters such as ‘, , <, >, <!--/→ in the data insertion points. For example, consider that there exists an attribute like the following in the web application:

<node attrib='$inputValue'/>

Considering the condition where if the inputValue = foo’ is instantiated and then inserted as the attrib value shown as below:

<node attrib='foo'/>

Then this XML document is not considered valid.

By defining the new entities, tester can extend the set of valid entities. If URI is the entity definition, then the entity is known as an external entity. External entities usually intend to force the XML parsers for the purpose of accessing the URI specified resources, e.g., a file on a remote system or the local machine. As shown in Figure 12, this configuration makes the application vulnerable to XXE attacks (External XML Entity), which can in turn, be used to perform DOS attack on the local system to gain unauthorized access to perform unauthorized activities like accessing the files on the local machine or a remote system, scanning the remote machines, and perform DOS on the remote systems.

7. *SSI injection*: SSI Injection, also known as Server-Side Includes injection, is special directives that are parsed by the web server before forwarding this page to the user. Appending an SSI directive into a static HTML document can be done by the following: -

<!--#echo var=DATE_LOCAL →

to print the current date and time.

The first thing that is done by the tester while testing for an SSI Injection vulnerability is to check whether the web server supports SSI directives or not. Followed by the tester tries to find the page in the web application where the insertion point is present in order to submit any input data, then tester performs a verification of whether the application has implemented a proper input validation mechanism.

8. *XPATH injection*: XPath is a type of programming language that is specifically designed for the purpose of addressing various sections of a document in an XML format. To perform the XPath injection testing, it is first checked whether injection can be done with the XPath query into the HTTP request that is going to be read or interpreted by the application, hence this allows the tester to execute the user-controlled XPath queries. After the successful exploitation of this vulnerability, it becomes possible for the tester to perform the authentication mechanism bypass and then gain the access to the restricted information without any authorization.
9. *IMAP/SMTP injection*: The purpose of performing this test is to check whether anyone can perform the injection of any arbitrary IMAP/SMTP exploit payload commands and send it to the mail servers due to improper sanitization of the input data. This type of injection attack allows the unauthorized access to the mail server which should not be accessed from the Internet directly. The tester’s role is to perform the analysis of the application’s capability in the input data handling in order to detect the vulnerable input parameters. It is necessary for the tester to send a malicious request to the web application server during the input validation testing phase and then perform an

analysis of the response. Finally, after the identification of all the vulnerable parameters, the tester needs to perform the determination of the level of injection that is possible in the application and further exploit the application by designing the test plan.

10. *Code injection*: To perform the testing of code injection, the tester need to perform the submission of the input data to be dynamically processed by the application's web server. These types of tests can target different types of scripting engines present in the server-side, e.g.., ASP, JSP and PHP. In order to protect against these types of attacks, secure coding practices and proper input validation is required.
11. *Command injection*: Command injection, also popularly known as an OS command injection, is a type of attack which takes place through a web application interface by executing the OS commands on the web application server. The web interface is known to be vulnerable to this type of exploit if it is not properly sanitized. The tester will test if this vulnerability exists in the application or not by injecting an OS exploit payload or command to the web application with the help of an HTTP request.
12. *HTTP splitting/smuggling*: HTTP splitting and smuggling details are as follows:

HTTP Splitting: HTTP Splitting is a type of vulnerability that performs the exploitation of the lack of sanitization of input by allowing an intruder to do the insertion of CR and LF (Carriage Return and Line Feed) characters into the response headers of the web application and to perform the 'splitting' of the HTTP response into two different parts of the HTTP response body. The headers that are most likely to be used by an attacker for conducting this attack are Set-cookie and Location header. The analyst must first detect all the identify all the insertion points present in the application that will directly affect the HTTP response headers, and then perform the verification of whether the insertion of a CR+LF sequence was successful or not.

HTTP Smuggling: HTTP Smuggling is a type of vulnerability that involves different techniques in which a modified HTTP message body can be successfully parsed and then understood by various types of web browser agents or WAF (Web Application Firewall). This attack technique allows the attacker to send one type of request to one device while the other device receives a different type of request. HTTP Request smuggling further provides the facilitation of several possible exploitations like XSS, partial cache poisoning and also bypassing the firewall protection.

13. *Host header injection*: The main reason why the host header exists is there are situations when there are multiple web applications hosted on the same IP address by the single web server. The HTTP header (or host header) mentions explicitly the website or web application to perform the incoming HTTP/HTTPS request processing. The host header value is used by the web application server for forwarding the HTTP request to the specified web application. The tester checks for the proper validation of the value of this header field by providing another domain in the host header request field. This type of injection attack turns out to be successful when the web application server performs the input processing to send the request to a host controlled by an attacker. Web cache poisoning, Web cache deception and password reset poisoning can be performed by host header injection.
14. *Server side template injection*: SSTI vulnerabilities occur when the user supplied input data is integrated into the server-side template of the application in an insecure way this improper function setting on the server can cause a remote code execution. Server side template injection vulnerabilities are present in both text and programming language format. In normal text format, users can insert any type of text within the HTML program. Considering the programming language format, the user supplied input data must be inserted inside a template code statement. In both the cases the methodology for testing this vulnerability consist of the following steps:
    - Detection of the insertion points in the application vulnerable to template injection
    - Identification of the application's template engine
    - Creating an exploit code for the vulnerability exploitation
15. *CSV injection*: It is a type of injection attack that takes place when the web applications embed the untrusted input inside CSV files. CSV injections are also popularly known as the formula injection. Microsoft Excel or LibreOffice Calc which is a spreadsheet program can be used to open a CSV file, so any cells starting with '=' will be understood by the software as a formula. The tester can check for CSV injection in web applications which have the functionality of generating and exporting the files of CSV format of user supplied data or user input.
16. *Rate limiting*: Rate Limiting, also popularly known as request limiting, is primarily used for controlling the amount of incoming and outgoing traffic to or from the server or network. Implementation of the limit on upcoming requests to the server is to allow for a better flow of data and to increase the security by preventing attacks such as Distributed DoS attack. Rate Limiting can be checked in the following components:
    - HTML Forms: Any application having login forms, contact forms, registration forms, feedback forms, submission forms, etc., need to be checked.
    - Emails: Any application having send email functionality can be verified for request limiting.
    - OTPs: Any application having OTP functionality can also be tested for request limiting.

4.2.7 Business logic testing

For testing the flaws present in the business logic of a multi-functional dynamic web application, it requires unconventional method. If an application's authentication mechanism is developed with the intention of performing a standard procedure following the same steps again and again in a specific order to authenticate a user, the pattern become predictable by any potential attacker and then they might be able to mess up with the actual web application logic and framework. For conducting these tests, the testers to think differently, develop abused and misuse cases or in other words, attack scenarios and use multiple testing techniques followed by the software functional testers. The application must be able to have a functionality to check whether logically valid data is being directly sent to the front end and the server side of an application. Some of the web controls or testing indicators for the business logic testing is as follows

1. *Unrestricted file upload*: The file upload functionality can pose a considerable risk or threat to the web applications if the file uploaded by any anonymous user turns out to be malicious after uploading it to the web servers or the

application servers. The primary step in the invalidated file upload functionality attacks is to send the malicious code to the target system. In order for this harmful attack to be successful, the attacker needs to find a way to get the code executed at the user's end. One of the most important factors while testing this vulnerability is to check what the application does with the file which is being uploaded, where it is stored and how it will be executed.

The security analyst should ensure if the application has the file upload functionality, then it should not allow the uploads of files having malicious extensions that are not relevant to the application's intended functionality. The analyst should also find ways to test if it is possible to bypass any filtering that the developer might have employed in the application's file upload functionality in order to mitigate and avoid any unwanted file uploads. The analyst performs the front-end (graphical user interface) functional valid testing to check that only the valid values are accepted in the application. Next, the analyst looks for variables where the insertion points take the cost or quality values. After the insertion points are found, the tester starts with the interrogation of the input fields with logically invalid data like unique identifiers. Tester has to check if they are working properly or not and that it does not accept any logically invalid data.

5. GUIDELINES IN CONDUCTING VPAT PROCESS

This section presents the do's and don'ts during the general assessment and technical assessment of web applications. General assessment is the code inspection and walkthroughs held during the development of code. Technical assessment (or specific) inspection focuses on specific possible vulnerability of the web application and will be tested while running the program.

5.1 Do's & don'ts during the general assessment

Do's

- Test all parameters for different kinds of attacks
- Tamper the Request headers and test.
- On a UAT (User Acceptance Test), dev instances must use tools like nikto, dirb etc. Observe if any critical details are being reflected anywhere or are using encoding such as base64.

Don'ts

- On a production environment don't test for rate limiters with large threads as it might cause a DOS attack.
- Don't reveal the vulnerability to any other person or take advantage of it for prank or own benefit
- Running automated scans on production environments, even Burp scans, should be avoided.
- Avoid Spidering/Crawling on POST requests in critical applications.
- Changing another user's data (even on UAT it should be avoided)
- If gained remote access by any means, just run basic commands for POC like whoami, hostname etc. Do not enumerate all files/folders on the server. Do not execute any harmful commands like deleting files, opening a remote port that may open a backdoor for others.

5.2 Do's & don'ts during the vulnerability specific assessment

Table 1 presents the list of Do's and Don'ts during the vulnerability specific assessment.

Table 1. Do's and Don'ts during the vulnerability specific assessment

S.No.	Vulnerability	Do's	Don'ts
1.	SQL Injection	In a black box test on a production environment just check SQL injection using time delay to confirm time-based SQL injection, error to confirm error based SQL Injection	Don't use any payload which deletes / updates / inserts any new record in the existing tables of database.
		In a grey box test on a UAT environment try to extract database name or version number as a POC	Don't drop any table
		In a grey box test on a UAT environment try to attain Remote Code Execution. After RCE, don't enumerate everything; instead run whoami and hostname to gain information.	On a production environment don't try to extract data from existing database tables or try to get Remote Code Execution
		If above test fail try to test Double query and Second Order SQL Injection	Don't execute harmful commands after gaining shell like shut down, or opening a port and creating a backdoor.
2.	Cross Site Scripting	Check for any user input that is being reflected in the response/ at some different page. If anything is being reflected in the response try to add a JavaScript code	In a production environment don't fuzz the parameters with a list of XSS payloads as it can lead to unwanted payloads getting stored in the website. Also it would generate huge traffic.
		(example: <script>alert(1)</script>)	
		If there exits places where images can be uploaded, try uploading malicious svg files with JavaScript code embedded.	Be careful with Stored XSS on production. After taking POC modify the entity to clear off the script. If not possible then intimate client for the same so that other user's do not get unnecessary pop ups
3.	Cross Site Request Forgery	Intercept the requests of some actionable items on the website and verify the presence of an anti CSRF token. If the token is not present generate a CSRF POC using Burp Suite	During assessment on a production environment do not delete/add/modify any file or perform any action which harms the website.
		If the CSRF token is present, try to bypass it by removing the token, adding any random value of the same characters or look if a sequence is being followed.	On Production, don't change the sensitive field. Do it for non-sensitive fields for POC.

4.	Missing SPF / DMARC Records	For testing purpose use tools like spoofcheck.py If either of the two - SPF or DMARC records is / are found to be missing then try to send a mail from a spoofed fictitious email of a website that is being tested (example: admin@website.com) to your email id.	Do not send mail to any other person even for testing purposes by exploiting the misconfigured SPF/DMARC records.
5.	XML External Entity Injection	Test for XXE wherever the request body contains XML. Try to craft payloads of XML by which either some information is leaked through response in the error or	Do not try the Billion laugh attack or any such xxe attack which can harm / compromise the server.
6.	Click Jacking	If there are any pages on the website that has important input fields, load that page in an iframe	
7.	Invalidated File Upload	Try to upload file other than the expected file type (example - uploading PHP, html, JS files in place where image is expected) and navigate to the URL of the upload file to see if it is executed Try to bypass the file upload check using double extension on a file to be uploaded Try to use Magic bytes of a valid file type (expected file type) and contents of a file we want to be uploaded (example - contents of a PHP file like : *<? php echo hello?>)* In a UAT environment if an invalidated file is uploaded successfully, try to upload a reverse/bind shell and see if it gets executed.	On a production environment, do not upload a reverse/bind shell or any other type of file that can harm the infrastructure of the website On a production environment do not upload very large files as it might caught DoS on the website If any harmful file gets uploaded make sure to intimate the team to delete those files later.
8.	Lack of Request Limiters	It should be tested wherever there is a function of sending an OTP or Email. For a black box test try only with 10 -15 requests in intruder with a single thread set in Burp intruder options. For a grey box test on a UAT environment try with 100 requests with 5 threads set in Burp intruder options	Do not provide email or mobile number of some other person while testing In a production environment do not send multiple requests using multiple threads, as it might cause a DoS attack For testing rate limiters on POST fields, do not over populate the database. Send minimal requests necessary for POC.
9.	LFI / RFI	In a grey box test on a UAT environment, in every parameter try to traverse directories and see if any file's content can be used. Use payloads like (../../../etc/passwd if on a linux system) In parameters try to access files from some other domain. If you are able to access then it is vulnerable to RFI	If config files are obtained, make sure to obfuscate passwords in report.
10.	Slow HTTP DoS	Use tools like slowloris/slowhttptest to check for the following vulnerability.	Don't test this on a production website. Do not put the script in an infinite loop. Causing many open connections, Stop the script once POC is taken.
11.	Host Header Injection	On intercepting the request using Burp Suite tamper with the Host header value and instead provide some other domain. If on forwarding the request, the Location header in the Response shows the tampered website, it shows that site is vulnerable to this attack. Try changing HTTP/1.1 to HTTP/1.0 and completely remove the Host Header. After forwarding the request, check if the Location parameter in the Response header shows some private IP Try adding X-Forwarded-Host Header in the captured request and add a domain that belongs to you. After forwarding the request, check whether the Location header in the Response has the domain that was entered in X-Forwarded-Host header in the request.	While testing don't redirect to any website, redirect to a domain that belongs to you. Do not try to reset any legitimate user's password (by email) by host header injected request.

6. CONCLUSION AND FUTURE WORK

In this paper, the main emphasis made on the most common vulnerabilities found in the web applications.

The paper also discusses in detail about the tools that can be used for automated testing such as Nmap, Acunetix, Nessus, OWASP ZAP, Dirbuster and many other such tools. The most common and popular tool used for penetration testing is tested by the white hat hackers/security analysts using automated and manual testing procedures. The paper provides the security testing techniques that can be done on any web application in an ethical way. The testing techniques are broadly categorized into two types, Black box and Gray box testing. This paper concludes that vulnerability assessment is the identification of the security vulnerabilities and penetration testing is the real time simulation of how an actual exploitation or attack will take place when the weakness, misconfiguration or security flaw existing in the web application. Burp Suite Professional. It can be used for manual exploitation as well as automated crawl and auditing of the web applications. This paper also suggested many burp extensions that can be installed in Burp Suite and can be helpful during the VAPT. Tools which can be helpful for manual exploitation are Metasploit, Wireshark, Burp Suite and many such tools. This paper concludes that VAPT is highly recommended to be performed in every organization as nowadays the data is stored on the internet, and this can pose a threat to the organizations reputation or money if any attack takes advantage of the security flaws existing in the organizations network. As a future work, we attempt to work on the VAPT process of mobile applications.

ACKNOWLEDGMENT

Authors would like to thank Team Lead at Safe Security Pvt. Ltd, Okhla, Delhi, India. (Formerly Lucideus Technologies Pvt Ltd) for providing supporting environment for conduction of this study.

REFERENCES

[1] Goutam, A., Tiwari, V. (2019). Vulnerability assessment and penetration testing to enhance the security of web application. 2019 4th International Conference on Information Systems and Computer Networks (ISCON), pp. 601-605. https://doi.org/10.1109/iscon47742.2019.9036175

[2] Kuruwitaarachchi, N., Abeygunawardena, P.K.W., Rupasingha, L., Udara, S.W.I. (2019). A systematic review of security in electronic commerce- threats and frameworks. Global Journal of Computer Science and Technology, 33-39. https://doi.org/10.34257/gjcstevol19is1pg33

[3] Kashif, M., Javed, M.K., Pandey, D. (2020). A surge in cyber-crime during COVID-19. Indonesian Journal of Social and Environmental Issues (IJSEI), 1(2): 48-52. https://doi.org/10.47540/ijsei.v1i2.22

[4] Foregenix Survey: https://www.foregenix.com/blog /over-75-of-global-magento-websites-at-high-risk-from-hackers-due-to-a-simple-security-oversight, accessed on 9 Aug. 2021.

[5] Humayun, M., Niazi, M., Jhanjhi, N., Alshayeb, M., Mahmood, S. (2020). Cyber security threats and vulnerabilities: A systematic mapping study. Arabian Journal for Science and Engineering, 45(4): 3171-3189. https://doi.org/10.1007/s13369-019-04319-2

[6] Asaduzzaman, M. (2020). Security Aspects of e-Payment System and Improper Access Control in Microtransactions (No. 3717). EasyChair. Available at: https://yahootechpulse.easychair.org/publications/preprint_download/ZFhp.

[7] Pentest monkey. http://pentestmonkey.net/, accessed on 15 Aug. 2021.

[8] Seng, L.K., Ithnin, N., Said, S.Z.M. (2018). The approaches to quantify web application security scanners quality: A review. International Journal of Advanced Computer Research, 8(38): 285-312. https://doi.org/10.19101/ijacr.2018.838012

[9] Toch, E., Bettini, C., Shmueli, E., Radaelli, L., Lanzi, A., Riboni, D., Lepri, B. (2018). The privacy implications of cyber security systems. ACM Computing Surveys, 51(2): 1-27. https://doi.org/10.1145/3172869

[10] Thomas, T.W., Tabassum, M., Chu, B., Lipford, H. (2018). Security during application development. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3173574.3173836

[11] Vamsi, P.R., Jain, A. (2021). Practical security testing of electronic commerce web applications. International Journal of Advanced Networking and Applications, 13(1): 4861-4873. https://doi.org/10.35444/IJANA.2021.13109

[12] P Raghu Vamsi, A.J. (2021). Getting started with android mobile applications security testing. Scientific and Practical Cyber Security Journal. (Available at: https://journal.scsa.ge/papers/getting-started-with-android-mobile-applications-security-testing/).

[13] Devi, R.S., Kumar, M.M. (2020). Testing for security weakness of web applications using ethical hacking. 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI) (48184), pp. 354-361. https://doi.org/10.1109/icoei48184.2020.9143018

[14] Priyanka, A.K., Sai Smruthi, S. (2020). Web application vulnerabilities: Exploitation and prevention. 2020 International Conference on Electrotechnical Complexes and Systems (ICOECS), pp. 729-734. https://doi.org/10.1109/icoecs50468.2020.9278437

[15] Amin, K., Sharma, P. (2020). Red team analysis of information security measures and response. Available https://www.academia.edu/download/64372201/IRJET-V7I4823.pdf.

[16] Vats, P., Mandot, M., Gosain, A. (2020). A comprehensive literature review of penetration testing & its applications. 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 674-680. https://doi.org/10.1109/icrito48877.2020.9197961

[17] Umrao, S., Kaur, M., Gupta, G.K. (2016). Vulnerability assessment and penetration testing. International Journal of Computer and Communication Technology, 200-203. https://doi.org/10.47893/ijcct.2016.1367

[18] Khera, Y., Kumar, D., Sujay, Garg, N. (2019). Analysis and impact of vulnerability assessment and penetration testing. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). https://doi.org/10.1109/comitcon.2019.8862224

[19] Hasan, A., Meva, D. (2018). Web application safety by penetration testing. International Journal of Advanced Studies of Scientific Research, 3(9). Available: https://www.academia.edu/download/58290599/SSRN-id3315587.pdf.

[20] Yaqoob, I., Hussain, S.A., Mamoon, S., Naseer, N., Akram, J., UR Rehman, A. (2017). Penetration testing and vulnerability assessment. Journal of Network Communications and Emerging Technologies (JNCET), 7(8): 10-18. Available at: https://www.jncet.org/Manuscripts/Volume-7/Issue-8/Vol-7-issue-8-M-03.pdf.

[21] Hasan, A.M., Meva, D.T., Roy, A.K., Doshi, J. (2017). Perusal of web application security approach. 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT), pp. 90-95. https://doi.org/10.1109/intelcct.2017.8324026

[22] Hussain, M.Z., Hasan, M.Z., Taimoor, M., Chughtai, A., Taimoor, M., Chughtai, A. (2017). Penetration testing in system administration. International Journal of Scientific & Technology Research, 6(6): 275-278. Available: https://www.ijstr.org/final-print/june2017/Penetration-Testing-In-System-Administration.pdf.

[23] Haque, M.F., Miah, M.B.A., Masud, F.A. (2017). Enhancement of web security against external attack. European Scientific Journal, 13(15): 228. https://doi.org/10.19044/esj.2017.v13n15p228

[24] Nagpure, S., Kurkure, S. (2017). Vulnerability assessment and penetration testing of web application. 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA). https://doi.org/10.1109/iccubea.2017.8463920

[25] Shinde, P.S., Ardhapurkar, S.B. (2016). Cyber security analysis using vulnerability assessment and penetration testing. 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave). https://doi.org/10.1109/startup.2016.7583912

[26] Nyambo, D., Yonah, Z., Tarimo, C. (2016). On the identification of required security controls suitable for converged web and mobile applications. International Journal of Computing and Digital Systems, 5(1). https://doi.org/10.12785/ijcds/050105

[27] Singh, H., Surender, J., Pankaj, K.V. (2016). Penetration testing: Analyzing the security of the network by Hacker's mind. Volume V IJLTEMAS, 56-60. https://www.academia.edu/download/46153650/56-60.pdf.

[28] Goel, J.N., Mehtre, B.M. (2015). Vulnerability assessment & penetration testing as a cyber defence technology. Procedia Computer Science, 57: 710-715. https://doi.org/ 10.1016/j.procs.2015.07.458

[29] Lamba, A. (2014). Cyber Attack prevention using VAPT tools (vulnerability assessment & penetration testing). Cikitusi Journal for Multidisciplinary Research, 1(2). http://www.cikitusi.com/gallery/9-996.pdf.

[30] Shah, S., Mehtre, B.M. (2014). An overview of vulnerability assessment and penetration testing techniques. Journal of Computer Virology and Hacking Techniques, 11(1): 27-49. https://doi.org/10.1007/s11416-014-0231-x

[31] Shah, S., Mehtre, B.M. (2013). A reliable strategy for proactive self-defence in cyber space using VAPT tools and techniques. 2013 IEEE International Conference on Computational Intelligence and Computing Research. https://doi.org/10.1109/iccic.2013.6724216

[32] Lallie, H.S., Shepherd, L.A., Nurse, J.R.C., Erola, A., Epiphaniou, G., Maple, C., Bellekens, X. (2021). Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. Computers & Security, 105: 102248. https://doi.org/10.1016/j.cose.2021.102248

[33] Kumar, B., Roy, S. (2021). An empirical study on usability and security of E-commerce websites. Advances in Intelligent Systems and Computing, 735-746. https://doi.org/10.1007/978-981-15-7527-3_69

[34] Toapanta Toapanta, S.M., Mera Caicedo, H.A., Naranjo Sanchez, B.A., Mafla Gallegos, L.E. (2020). Analysis of security mechanisms to mitigate hacker attacks to improve e-commerce management in ecuador. 2020 3rd International Conference on Information and Computer Technologies (ICICT). https://doi.org/10.1109/icict50521.2020.00044

[35] Khera, Y., Kumar, D., Sujay, Garg, N. (2019). Analysis and impact of vulnerability assessment and penetration testing. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). https://doi.org/10.1109/comitcon.2019.8862224

[36] Rahman, M.A., Amjad, M., Ahmed, B., Siddik, M.S. (2020). Analyzing web application vulnerabilities. Proceedings of the International Conference on Computing Advancements. https://doi.org/10.1145/3377049.3377107

[37] Lis, A. (2019). Comparison and analysis of web vulnerability scanners (Bachelor's thesis). https://elib.uni-stuttgart.de/bitstream/11682/10634/1/bachelorthesis_alexander_lis.pdf.

[38] Abdullah, H.S. (2020). Evaluation of open source web application vulnerability scanners. Academic Journal of Nawroz University, 9(1): 47. https://doi.org/10.25007/ajnu.v9n1a532

[39] Amankwah, R., Chen, J., Kudjo, P.K., Towey, D. (2020). An empirical comparison of commercial and open-source web vulnerability scanners. Software: Practice and Experience, 50(9): 1842-1857. https://doi.org/10.1002/spe.2870

[40] Pan, Y. (2019). Interactive application security testing. In 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA), pp. 558-561. https://doi.org/10.1109/icsgea.2019.00131

[41] Vega, E.A.A., Orozco, A.L.S., Villalba, L.J.G. (2017). Benchmarking of pentesting tools. International Journal of Computer and Information Engineering, 11(5): 602-605. https://doi.org/doi.org/10.5281/zenodo.1130587

[42] Building a Pentesting Lab. (2020). The Pentester Blueprint, 65-81. https://doi.org/10.1002/9781119684367.ch5

[43] Felderer, M., Büchler, M., Johns, M., Brucker, A.D., Breu, R., Pretschner, A. (2016). Security testing: A survey. In Advances in Computers, 101: 1-51. https://doi.org/10.1016/bs.adcom.2015.11.003

[44] Sołtysik-Piorunkiewicz, A., Krysiak, M. (2020). The cyber threats analysis for web applications security in Industry 4.0. Studies in Computational Intelligence, 127-141. https://doi.org/10.1007/978-3-030-40417-8_8

[45] Dang, Q.H. (2015). Secure Hash Standard. https://doi.org/10.6028/nist.fips.180-4.

[46] Common Vulnerabilities and Exposures. Available online: https://cve.mitre.org/, accessed on 15 Aug. 2021.

[47] Rahalkar, S. (2020). Extending burp suite. A Complete Guide to Burp Suite, 131-145. https://doi.org/10.1007/978-1-4842-6402-7_9

[48] GitHub, I. (2016). GitHub. URl: https://github.com/, accessed on 15 Aug. 2021.

[49] Kali Linux. URl: https://kali.org/, accessed on 15 Aug. 2021.

[50] Parrot Security. URl: https://parrotlinux.org/, accessed on 15 Aug. 2021.

[51] Security Onion. URl: https://securityonion.net/, accessed on 15 Aug. 2021.

[52] Sy, E., Mueller, T., Burkert, C., Federrath, H., Fischer, M. (2020). Enhanced performance and privacy for TLS over TCP fast open. Proceedings on Privacy Enhancing Technologies, 2020(2): 271-287. https://doi.org/10.2478/popets-2020-0027

[53] Mendez, X. Wfuzz—The Web Fuzzer. Available online: https://github.com/xmendez/wfuzz, accessed on 15 Aug. 2021.

[54] Exploit database. https://www.exploit-db.com/, accessed on 15 Aug. 2021.

[55] Klein, A. (2008). Attacks on the RC4 stream cipher. Designs, Codes and Cryptography, 48(3): 269-286. https://doi.org/10.1007/s10623-008-9206-6

[56] Alenezi, M., Nadeem, M., Asif, R. (2021). SQL injection attacks countermeasures assessments. Indonesian Journal of Electrical Engineering and Computer Science, 21(2): 1121-1131. https://doi.org/10.11591/ijeecs.v21.i2.pp1121-1131

[57] Fossati, T., Tschofenig, H. (2016). Transport layer security (TLS)/datagram transport layer security (DTLS) profiles for the internet of things. Transport. https://doi.org/10.17487/rfc7925

[58] Viega, J., Messier, M., Chandra, P. (2002). Network security with openSSL: cryptography for secure communications. " O'Reilly Media, Inc.".

[59] Pentester Land. https://pentester.land/, accessed on 15 Aug. 2021.

[60] Harris, J.K. (2018). Heartbleed: A case STUDY. Issues in Information Systems, 19(2): https://doi.org/10.48009/2_iis_2018_99-108

[61] Calzavara, S., Roth, S., Rabitti, A., Backes, M., Stock, B. (2020). A tale of two headers: A formal analysis of inconsistent click-jacking protection on the web. In 29th {USENIX} Security Symposium ({USENIX} Security 20), 683-697. Available at: https://www.usenix.org/system/files/sec20fall_calzavara_prepub.pdf.

LIST OF ABBREVIATIONS

API	Application Programming Interface
ARP	Address Resolution Protocol
ASVS	Application Security Verification Standard
CIA	Confidentiality Integrity and Authentication
CLI	Command Line Interface
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
CRLF	Carriage Return and Line Feed
CVE	Common Vulnerabilities and Exposure
CVSS	Common Vulnerability Scoring System
DHCP	Dynamic Host Configuration Protocol
DMARC	Domain Based Message Authentication Reporting
DNS	Domain Name Server
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
ID	Identifier
IDS	Intrusion Detection System
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
ISS	Internet Security Scanner
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LFI	Local File inclusion
MASVS	Mobile Application Security Verification Standard
MITRE	MITRE Adversarial Tactics, Techniques, and Common Knowledge
ORM	Object Relational Model
OS	Operating System
OSI	Open System Interconnection
OSINT	Open-source intelligence
OTP	One Time Password
OWASP	Open Web Application Security Project
POC	Proof of Concept
PT	Penetration Testing
RARP	Reverse Address Resolution Protocol
RATS	Remote Access Trojan
SANS	SysAdmin, Audit, Network and Security
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Mail Protocol
SOAP	Simple Object Access Protocol
SPF	Sender Policy Framework
SQL	Structured Query Language
SSI	Server Sides Include
SSL	Secure Sockets Layer
SSTI	Server-Side Template Injection
SWAAT	Securing Web Application Technologies
TCP	Transfer Control Protocol
TLS	Transport Layer Security
UAT	User Acceptance Test
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VA	Vulnerability Assessment
VAPT	Vulnerability Assessment and Penetration Testing
WAF	Web Application Firewall
WASC	Web Application Security Consortium
WSDL	Web Services Description Language
XML	Extensive Markup Language
XSS	Cross Site Scripting
XXE	External XML Entity
ZAP	Zed Attack Proxy