An Approach to Automate Vulnerability Assessment and Penetration Testing for Start-ups

Abstract:

Vulnerability Assessment and Penetration Testing (VAPT) describes a broad range of security assessment services designed to identify and address cyber security exposures across an organisation's IT estate. Cyber-attacks and threats are real-world problems today with thousands of networks and websites and being compromised every day. Some of the normal reasons we see for carrying out a VAPT are: a) Customer needs, specifically for security of their information, b) Compliance of Industry standards, c) Validation of security controls, d) Proactive security audits to protect their data and systems from new threats.

Even for start-ups, to understand and tackle each vulnerability manually, is not easy. It involves lot of manual efforts and time. Also, since each start-up might have different requirements, (i.e features from various stake-holders on its application), therefore, customizing the VAPT process exclusively for a start-up makes the task of the process developer complex and at times difficult to achieve in real time. An attempt is made to automate the VAPT process for start-ups. The process consists of identification of vulnerability and penetration, and automate the process using various tools or by bash scripting. In this work, we present the automated VAPT process, developed specifically for the start-ups.

Key words: Vulnerability Assessment and Penetration Testing, Cyber security, start-ups

An Approach to Automate Vulnerability Assessment and Penetration Testing for Start-ups

Extended Abstract

1. Introduction:

Vulnerability Assessment and Penetration Testing (VAPT) is a security testing to identify security vulnerabilities in an application, network, endpoint and cloud. VAPT has unique strengths and are often collectively done to achieve security (pro-active and reactive), compliance, and validation goals of an organization. To be specific, vulnerability assessment (Akgul, 2006, Priyawati *et al.* 2022) process begins with discovering assets in a computing environment. The process involves, a) identification of flaws in networks and applications, b) evaluation of risk levels of each vulnerability, and c) prioritizing high-risk issues. Further, it highlights the problem areas and suggest improvements. Vulnerability remediation consists of a) reconfiguring the system, b) managing patches, and c) security infrastructure strengthening.

The penetration testing process (also called as 'Pentesters') consists of a) determining the scope of testing and the level of exploitation, b) identification of vulnerabilities to assess the severity of related risks. They simulate real-world attacks and exploit the identified vulnerabilities, by injecting agents to enable access to the system for a specified period. Pentesting involves complex steps which form an attack (Valea and C. Oprişa, 2020). The Pentesters also carry out risk analysis to understand the level of access to the system the attack achieved. Pentesting is used for proactive defense and information systems protection. (Stefinko *et al.*, 2016). Knowing the risks, the organization implements the suggested fixes to mitigate the vulnerabilities in their system. Pentests has been an approach to prevent security breaches by mimicking 'black hat' hackers to expose possible exploits and vulnerabilities. (Chaudhary *et al.*, 2020) Thus, VAPT helps to protect organization by providing visibility of security weaknesses and guidance to address them (Dolan-Gavitt, *et al,* 206; Finder 2004; Gautam and Tiwari, 2019). Also, VAPT is increasingly important for organizations wanting to achieve compliance with standards including the GDPR, ISO 27001 and PCI DSS. The next section describes the automation process developed, a brief description of the software used to develop the automated processes followed by conclusions.

2. The Overall Automation Process

In this section, we discuss the two-step automation process developed, keeping in view its application to a start-up organization.

Step 1: End Point Detection and Classification

End point is a specific part of URL which indicates a particular vulnerability which might be present in the following website. For e.g. ‘www.abc.com?id=1’, might signify that it is vulnerable as it has “?id=1” in it. Endpoint detection works by monitoring endpoint and network events and recording the information in a central database where further analysis, detection, investigation, reporting, and alerting take place. Not all endpoint detection and response tools work the same way offer the same spectrum of capabilities. Some endpoint detection and response tools perform more analysis on the agent, while others focus on the backend via a management console. Some vary in collection timing and scope or in their ability to integrate with threat intelligence providers. However, all endpoint detection and response tools perform the same essential functions with the same purpose: to provide a means for continuous monitoring and analysis to more readily identify, detect, and prevent advanced threats.

Various end points were detected and classified into different vulnerabilities. For example: Endpoints (.php, .aspx, .jsp) can be classified into vulnerabilities such as SQL Injection, XSS injection, SSTI etc.

a. SQL Injection: SQL injection usually occurs when a user is asked for an input (such as their username/userid), and instead, the user gives an SQL statement that runs on the database.
b. XSS: Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.
c. SSTI: A server-side template injection occurs when an attacker is able to use native template syntax to inject a malicious payload into a template, which is then executed.

Step 2: Analysis and design of automation process

Typically, our approach can be helpful to automate the following vulnerabilities: a) Endpoint Detection, b) Misconfigured Mail Server, c) LDAP (Lightweight Directory Access Protocol), d)

SSTI (Server Side Template Injection), e) Host Header Injection, f) DNS Misconfiguration, g) Command Injection, h) SQL Injection, i) Cross-Site Scripting(XSS Injection), j) Click jacking, k) Web Application Firewall Bypass, l) HTML Injection. In this section we describe the automation process for specific few such as a) SQL injection b) Cross Site Scripting and c) SSTI Automation. We explain the details of these as follows:

a. SQL injection: SQL injection allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

   A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information. Many high-profile data breaches in recent years have been the result of SQL injection attacks, leading to reputational damage and regulatory fines. In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period. The following figure explains the details of SQL injection.

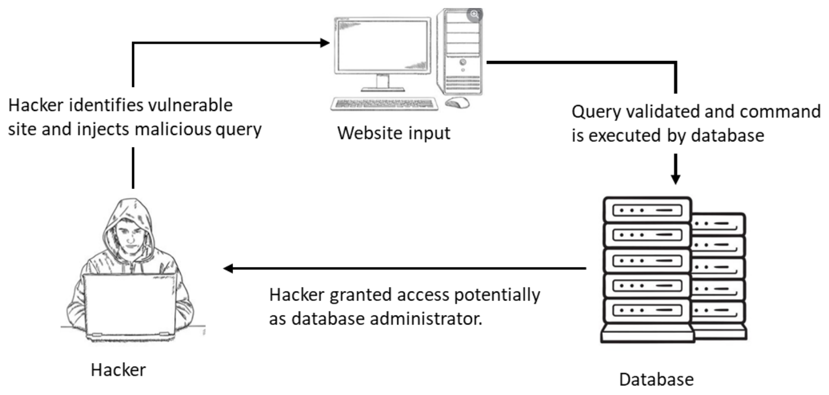
Hacker identifies vulnerable site and injects malicious query
Website input
Query validated and command is executed by database
Hacker granted access potentially as database administrator.
Hacker
Database

Figure 1: SQL Injection

Few examples of SQL injection are as follows:

- Retrieving hidden data: here an SQL query is modified
- Subverting application logic: here a query is changed to interfere with the application's logic.
- UNION attacks: here data from different database tables is retrieved.

b) Cross-site Scripting: Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the user's interactions. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, and to carry out any actions that the user is able to perform. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data.

There are three main types of XSS attacks. These are:

- Reflected XSS, where the malicious script comes from the current HTTP request.
- Stored XSS, where the malicious script comes from the website's database.
- DOM-based XSS, where the vulnerability exists in client-side code rather than server-side code.

Typically, XSS works by manipulating a vulnerable web site so that it returns malicious JavaScript to users. When the malicious code executes inside a victim's browser, the attacker can fully compromise the interaction with the application.

An attacker who exploits a cross-site scripting vulnerability is typically able to: a) Impersonate or masquerade as the victim user, b) Carry out any action that the user is able to perform, c) Read any data that the user is able to access, d) Capture the user's login credentials, e) Perform virtual defacement of the web site, f) Inject trojan functionality into the web site.

The actual impact of an XSS attack generally depends on the nature of the application, its functionality and data, and the status of the compromised user. For example, in a brochureware application (where all users are anonymous and all information is public), the impact will often be minimal. In an application holding sensitive data, such as banking transactions, emails, or healthcare records, the impact will usually be serious.

c) Server-side template injection: A Server-Side Template (SST) injection occurs when an attacker is able to use native template syntax to inject a malicious payload into a template, which is then executed server-side. Template engines are designed to generate web pages by combining fixed templates with volatile data. Server-side template injection attacks can occur when user input is concatenated directly into a template, rather than passed in as data. This allows attackers to inject arbitrary template directives in order to manipulate the template engine, often enabling them to take complete control of the server. Figure 2 shows an overview of SST injection

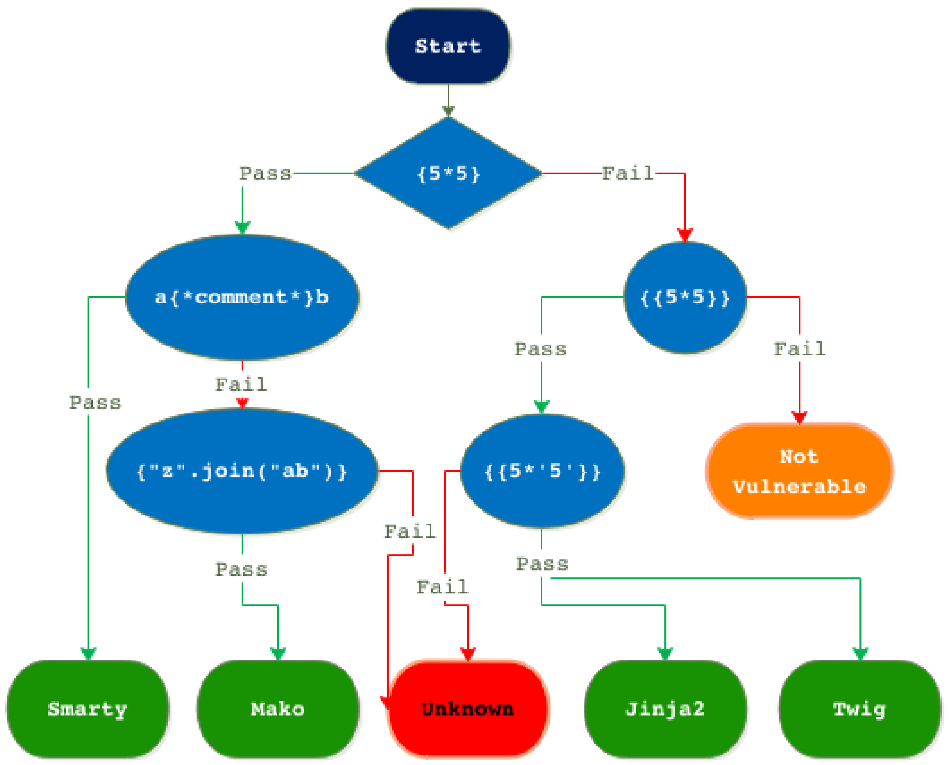

Figure 2 : SST Injection

Specifically for SST injection, we follow a unique procedure called as Detect, Identify, Exploit, Explore and Attack. These are explained briefly as follows:

Detect: As with any vulnerability, the first step towards exploitation is being able to find it. Perhaps the simplest initial approach is to try fuzzing the template by injecting a sequence of special characters commonly used in template expressions, such as the polyglot ${{<% [%'"}}%\. In order to check if the server is vulnerable you should spot the differences between the response with regular data on the parameter and the given payload. If an error is thrown it will be quite easy to figure out that the server is vulnerable and even which engine is running.

Identify: Once you have detected the template injection potential, the next step is to identify the template engine. Although there are a large number of templating languages, many of them use very similar syntax that is specifically chosen not to clash with HTML characters.

Exploit: The first step after finding template injection and identifying the template engine is to read the documentation.

Explore: Assuming no exploits have presented themselves, the next step is to explore the environment to find out exactly what you have access to.

Attack: At this point you should have a firm idea of the attack surface available to you and be able to proceed with traditional security audit techniques, reviewing each function for exploitable vulnerabilities. It's important to approach this in the context of the wider application - some functions can be used to exploit application-specific features.

The theme of automation is shown in the block diagram presented in Figure 3.

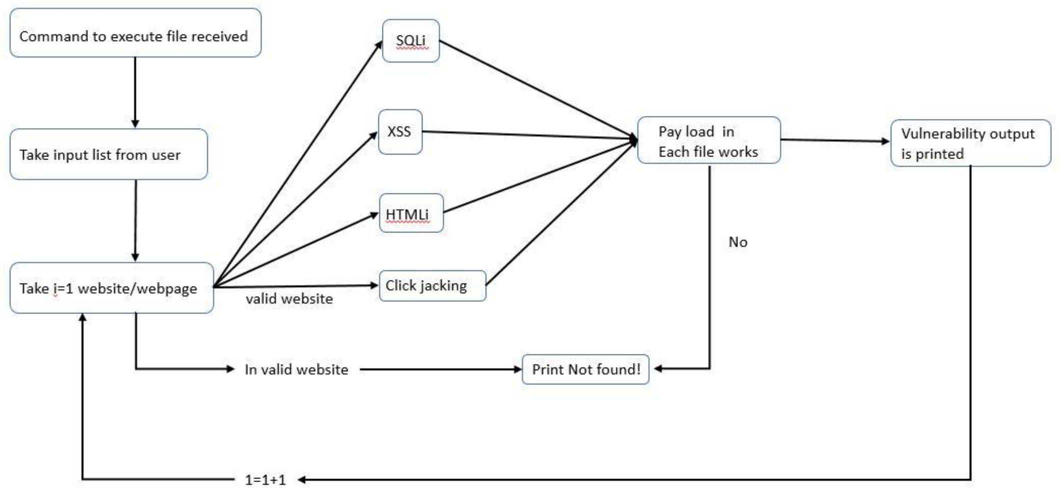
Command to execute file received
Take input list from user
Take i=1 website/webpage
SQLi
XSS
HTMLi
Click jacking
valid website
Pay load in Each file works
Vulnerability output is printed
No
In valid website
Print Not found!
1=1+1

Figure 3: The Block Diagram Showing the Theme of Automation

The theme of the approach used for automation can be quickly explained as follows: When the Bash Script is executed, it takes the input as a text file which contains a list of websites that need to be checked for vulnerabilities. Execution starts with taking the first website URL from the text file and it is sent as an input to each of sub-bash files (SQLi, XSS, HTMLi, Clickjacking) (this manuscript is limited only to the SQLi and XSS) which starts its scanning and testing of each and every payload and tries injecting it on the website and looks for its corresponding effect. If the code spots any changes, it sends a message saying vulnerability is found at such website URL and gives further detailed report output on all other check web URLs. Such process is repeated for each bash scripted vulnerability for each website from the input text file and the output is thus extracted in one go where we can see what all vulnerabilities are present in the website under one single output window.

3. Software used for Automation

It is difficult to automate a process without any software, in order to develop the automation process for VAPT, we used the software such as: i) VMware Virtual Box and ii) Burp Suite

i) VMware Virtual Box is a type-2 hypervisor for x86 virtualization developed by Oracle Corporation, VirtualBox is free and open-source software. Currently version 7 is available. This software comes with various features such as a) Mouse pointer integration, meaning automatic coupling and uncoupling of mouse cursor when moved inside and outside the virtual screen, if supported by guest operating system, b) Special drivers and utilities to facilitate switching between systems, etc..

ii) Burp Suite: Burp or Burp Suite is a set of tools used for penetration testing of web applications. It is developed by the company named Portswigger, Burp Suite is an integrated platform and graphical tool for performing security testing of web applications, it supports the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.

4. Conclusions

This study will help in automating the VAPT exclusively for the start-ups. While the initial testing of beta automation process is under progress, the initial findings, with respect to its applicability are very encouraging. The automated VAPT was tested on few start-ups, some of them were in the domain of video messaging platform for iOS, Android, and Windows using mobile devices. These organizations, in particular, enabled their user/s to live stream brief video clips. Few other organizations were helping the clients to smoothly transition from the existing platform to a newer platform.

For a few comparative tests conducted, it was observed that the time in order to execute the automated process is reduced considerably with respect to the manual exercise. In one of the instance, the time saving was about 80% as compared to manual testing. Most importantly, being start-ups, this approach helps them evaluate most of the highly vulnerable elements. Although, on the accuracy front, it appears that the manual exercise provides marginally better results. This has given thought provoking insights and has motivated us to develop a hybrid version or a two stage version of VAPT, wherein, the automated process would quickly conduct VAPT and broadly assess major vulnerabilities and in the next stage on can evaluate manually, to fine tune the processes. Our work is limited to identification of flaws that are subject to vulnerability. Scope exists to develop complete package on evaluation of risks and vulnerability remediation.

With a few rounds of modification, we feel sure that this approach will help the start-ups immensely.

References:

Akgul, Y, 2016. Web site accessibility, quality and vulnerability assessment: a survey of government web sites in the Turkish republic. *Journal of Information Systems Engineering & Management*, *1*(4), p.50.

Dolan-Gavitt, B, Hulin, P, Kirda, E, Leek, T, Mambretti, A, Robertson, W, Ulrich, F and Whelan, R., 2016. Lava: Large-scale automated vulnerability addition. In *IEEE Symposium on Security and Privacy (SP)* pp. 110-121.

Finder, S.G, 2004. Vulnerability in human subject research: existential state, not category designation. *The American Journal of Bioethics*, *4*(3), pp.68-70.

Goutam, A and Tiwari, V, 2019. Vulnerability assessment and penetration testing to enhance the security of web application. In *4th International Conference on Information Systems and Computer Networks (ISCON)* pp. 601-605.

Nathani, B.C and Adi, E, 2012. Website vulnerability to session fixation attacks. *J. Information Eng. App. II*, *7*, pp.32-36.

Priyawati, D, Rokhmah, S and Utomo, I.C, 2022. Website Vulnerability Testing and Analysis of Website Application Using OWASP. *International Journal of Computer and Information System (IJCIS)*, *3*(3), pp.142-147.

Valea and C. Oprişa, 2020, Towards Pentesting Automation Using the Metasploit Framework, In, *IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)* pp. 171-178

Y. Stefinko, A. Piskozub and R. Banakh, 2016. Manual and automated penetration testing. Benefits and drawbacks. Modern tendency, In 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), pp. 488-491.

S. Chaudhary, A. O'Brien and S. Xu, 2020. Automated Post-Breach Penetration Testing through Reinforcement Learning, In IEEE Conference on Communications and Network Security (CNS), pp. 1-2.

Dalalana Bertoglio, D., Zorzo, A. 2017. Overview and open issues on penetration test. Journal of Brazillian Computer Soceity 23(2), pp. 1-16