

Comprehensive Exploration of Web Application Security Testing with Burp Suite Tools

Rhythm Choudhary¹, Jhanvii Rawat², Garima Singh³

¹Assistant Professor, Department of Computer Applications, Maharaja Surajmal Institute, C-4, Janakpuri, New Delhi

^{2,3}Student, Department of Computer Applications, Maharaja Surajmal Institute, C-4, Janakpuri, New Delhi

Abstract

This research paper investigates the pivotal role of Burp Suite tools in web application security testing, aligned with the foundational principles of secure coding practices outlined by the Open Web Application Security Project (OWASP). The paper provides an in-depth overview of Burp Suite tools, emphasizing their features and the significance of web application vulnerability scanning in enforcing robust security practices. The exploration includes a detailed examination of Burp Repeater and Burp Intruder, with a practical demonstration of a Burp Suite cluster bomb attack. This research contributes to advancing the understanding and application of Burp Suite tools for effective web application security testing within the framework of OWASP secure coding practices.

Keywords: Burp Suite, web application security, OWASP, Kali Linux, vulnerabilities, Burp intruder, Burp repeater.

1. INTRODUCTION

In the ever-evolving digital landscape, web application security is paramount due to the increasing complexity of cyber-attacks. This paper underscores the importance of effective testing and vulnerability assessments for ensuring the resilience of web applications. Burp Suite emerges as a versatile penetration testing tool, offering a rich set of features for evaluating the security posture of web services. The research aims to systematically analyze Burp Suite tools, providing insights into their functionalities and their role in security testing. The study employs controlled experiments on representative web applications, leveraging Burp Suite alongside other leading security testing tools.

Overall, this research paper serves as a comprehensive guide for cybersecurity novice and practitioners seeking to leverage the power of Burp Suite tools in effectively testing the security of web applications which contributes to the existing body of knowledge by offering valuable insights, empirical evidence, and critical analysis to enhance the understanding and application of Burp Suite in the realm of web application security.

2. SECURE CODING PRACTICES OWASP

The OWASP Secure Coding Practices Quick Reference Guide offers essential guidelines for developers to write secure code [1]. The goal of the guide, which was created by the Open Web Application Security Project (OWASP), is to assist developers in identifying and address common security

vulnerabilities in their software applications. Focused on preventing security vulnerabilities at the root level, the guide covers various topics, including session management, error handling, output encoding, input validation, access controls, and cryptography. The guide's practicality and concise recommendations make it a valuable resource for developers, helping reduce common vulnerabilities like SQL injection and cross-site scripting. Each topic provides concise and actionable recommendations that developers can easily implement in their code.

Developers can reduce the possibility of common vulnerabilities like SQL injection, cross-site request forgery (CSRF), cross-site scripting (XSS), and unsecured direct object references by adhering to these best practices. It emphasizes input validation techniques, output encoding strategies, secure authentication, and session management practices. It provides recommendations for implementing proper input validation techniques, such as whitelisting, blacklisting, and regular expressions. Output encoding is another critical aspect covered in the guide, which helps prevent XSS attacks by properly encoding user-generated content before displaying it in web pages. The guide provides guidance on using context-specific encoding techniques to prevent malicious script execution. Furthermore, the guide addresses secure authentication and session management, stressing the importance of using strong password hashing algorithms, implementing multi-factor authentication, and securely managing session tokens to prevent unauthorized access. Error handling and logging practices are also emphasized, as proper error handling can help prevent information leakage and assist in identifying and addressing potential security vulnerabilities.

The Quick Reference Guide for OWASP Secure Coding Practices is designed to be concise, easy to understand, and readily applicable. It serves as a valuable resource for developers of all skill levels, providing practical guidance and examples to enhance the security of software applications. Developers can greatly lower the chance of introducing security flaws and help create more robust and secure systems by implementing these secure coding principles [9].

2.1 OWASP Checklist for Secure Coding Practices

The extensive OWASP Secure Coding Practices Checklist is intended to guide developers in implementing secure coding practices to prevent common security vulnerabilities in their software applications. It serves as a practical reference to ensure that applications are developed with security in mind from the ground up [1]. The checklist covers a wide range of security-related areas, including authentication, session management, access control, input validation, output encoding, cryptography, error handling, and more[7]. By adhering to the recommendations outlined in the checklist, developers can significantly reduce the risk of introducing vulnerabilities into their applications.

A. Input validation

Input validation is a critical aspect of secure coding, and the checklist provides guidance on implementing proper input validation techniques [7]. It emphasizes the importance of sanitising and validating data provided by users to stop injection attacks like cross-site scripting (XSS) and SQL injection. The checklist recommends using whitelisting, blacklisting, or regular expressions to validate input and ensure that only expected and safe data is processed by the application. According to OWASP Secure Coding Practices Checklist [1], for input validation, every data validation procedure must be carried out on a reliable system, such the server. verification of all information obtained from unreliable sources, such as file streams and databases, encoding of data to a common character set.

B. Output encoding

Output encoding is another important consideration to prevent XSS attacks. The checklist provides recommendations on proper encoding techniques based on the context in which the data is displayed. It emphasizes the need to encode user-generated content to prevent malicious scripts from being executed in users' browsers. According to OWASP Secure Coding Practices Checklist, follow tried-and-true procedures for every kind of outgoing encoding, clean up any output that uses untrusted data in operating system commands. When it comes to output encoding, it is crucial to perform encoding operations on trusted systems and utilize established routines for different types of encoding. Contextual output encoding should be applied to data originating from untrusted sources, and all characters should be encoded unless confirmed safe. Additionally, output containing untrusted data used in queries and operating system commands should be sanitized appropriately.

C. Secure authentication

Secure authentication and session management are addressed in the checklist to prevent unauthorized access. It recommends the use of strong password hashing algorithms, such as bcrypt or Argon2, and encourages the implementation of multi-factor authentication to enhance security. Proper session management techniques, such as using secure session tokens and enforcing session timeouts, are also highlighted.

D. OWASP Secure Coding Practices Checklist

The Checklist emphasizes secure authentication and password management. It recommends requiring authentication for all pages, utilizing trusted authentication controls, and enforcing password complexity and length requirements. The checklist also highlights the importance of securely storing passwords, implementing proper password hashing, and securely managing password reset and changing operations. These practices aim to protect user accounts, prevent unauthorized access, and ensure effective password management in applications. According to OWASP Secure Coding Practices Checklist, authentication should be required for all pages and resources, except those explicitly designated as public, utilize established and tested authentication services whenever possible. If the application manages a credential store, store only cryptographically strong one-way salted hashes of passwords, with the table/file being writable only by the application (avoid using MD5 algorithm if possible). Password hashing should be implemented on a trusted system, such as the server, set short expiration times for temporary passwords and links. These practices ensure secure authentication and effective password management to protect user accounts and prevent unauthorized access.

Access control is essential to ensure that users have appropriate privileges within an application. The checklist emphasizes the importance of implementing role-based access control (RBAC) and least privilege principles to limit users' access to only what is necessary for their intended tasks such as: maintain consistency between server-side implementation and presentation layer rules, encrypt and perform integrity checks on client-stored state data, Limit the number of transactions per user/device to deter automated attacks, periodically re-validate user authorization for long authenticated sessions, assign minimum privileges to service accounts or accounts interacting with external systems. Error handling and logging practices are covered to assist developers in handling errors securely. The checklist emphasizes the importance of providing informative error messages to developers while avoiding

disclosing sensitive information to users. It also encourages the implementation of proper logging mechanisms to facilitate incident response and security analysis. Cryptography is a fundamental aspect of secure coding, and the checklist provides guidance on using cryptographic algorithms and libraries correctly. It emphasizes the importance of using well-established and properly implemented cryptographic functions and storing cryptographic keys securely.

By following the recommendations in the OWASP Secure Coding Practices Checklist, application developers may greatly improve the security posture of their products. It provides actionable guidelines and best practices, allowing developers to incorporate security measures into the software development lifecycle effectively ^[7]. Integrating these secure coding practices from the early stages of development helps to build applications that are resilient against common security threats and vulnerabilities.

3. VULNERABILITY SCANNING TOOL : BURP SUITE

This section explores the utilization of Burp Suite as a web application vulnerability scanning tool, running on the Kali Linux operating system. Kali Linux provides a robust environment for security assessments, aligning with industry best practices. Burp Suite comprises various tools that synergistically contribute to web application security testing. By leveraging the powerful features of Burp Suite within the Kali Linux ecosystem, we were able to effectively identify and analyze vulnerabilities in web applications, aligning with industry best practices for secure coding and testing.

3.1 Burp Suite and its tools

A popular and effective collection of tools for evaluating the security of web applications is called Burp Suite. It is created by PortSwigger, a leading provider of web security solutions. Burp Suite [3] offers a comprehensive range of functionalities that aid in identifying vulnerabilities, analyzing web traffic, and assessing the overall security posture of web applications. The suite includes several tools that work together synergistically to provide a holistic approach to web application security testing. Burp Proxy facilitates traffic interception and manipulation, Burp Spider automates the discovery of application components, Burp Scanner identifies vulnerabilities, and Burp Intruder enables customized attack scenarios. The section provides an overview of Burp Proxy and focuses on Burp Repeater and Burp Intruder, demonstrating their capabilities through practical examples.

Another important tool within Burp Suite is Burp Spider. It performs automated crawling of web applications, discovering and mapping out the various pages and functionality within the target application. By traversing through the application, it helps in identifying potential security vulnerabilities, such as missing input validation or unprotected sensitive information.

Burp Scanner is a highly regarded tool that performs automated security scans on web applications. It applies a wide range of security checks to identify common vulnerabilities, including insecure direct object references, cross-site scripting (XSS), and SQL injection. Burp Scanner combines both active and passive scanning techniques, providing comprehensive coverage in vulnerability detection.

Burp Intruder is a versatile tool that facilitates automated and manual security testing by generating and manipulating custom HTTP requests. By using brute-force attacks, fuzzing, and parameter tampering, users can enable the discovery of vulnerabilities related to input validation, session handling, and more. This tool empowers users to test the application's resilience against various attack vectors.

Burp Suite also includes other useful tools like Burp Repeater, which allows for manual request modification and replay for testing specific scenarios. Burp Decoder aids in decoding and encoding data

to identify vulnerabilities related to input handling and data transformation. Additionally, Burp Comparer assists in comparing responses and identifying differences that may indicate security weaknesses or anomalies.

In order to evaluate the security of web applications, security experts, penetration testers, and developers frequently use Burp Suite, which offers an extensive and integrated set of tools. Its diverse range of features and customizable options enable thorough testing, vulnerability identification, and validation of security controls. By leveraging Burp Suite, organizations can enhance the security posture of their web applications and mitigate potential risks.

3.2 Burp Proxy

One of the primary components is Burp Proxy, which acts as a web proxy server. The Proxy tab in Burp Suite serves as a central hub for monitoring and analyzing web traffic. It captures and analyzes data exchanged between web browsers and servers, providing detailed insights into requests and responses [8]. In Burp Suite's Proxy tab, users can access the primary hub for monitoring and analyzing web traffic in the Community Edition. It provides the ability to observe all the data exchanged between web browsers and the servers of visited sites.

By default, Burp Suite captures traffic from all sites. However, when utilizing the built-in browser provided by Burp Suite, it focuses solely on reporting traffic to that specific browser. This simplifies information management within Burp Suite, as activities in users' regular browsers won't interfere with the generated reports.

The Proxy tab is divided into four sub-tabs: Intercept, HTTP History, WebSocket History, and Options. These sections allow users to capture and analyze traffic for in-depth examination within Burp Suite.

3.3 Burp Repeater

Burp Repeater allows users to intercept and edit specific requests between the client and server, facilitating thorough vulnerability testing. It supports session handling, enabling the testing of authenticated functionality. The server's response is then displayed within the Repeater interface, enabling users to assess the application's behaviour and identify potential security weaknesses. The section provides a practical demonstration of Burp Repeater, guiding users through intercepting requests, modifying user interactions, and forwarding requests.

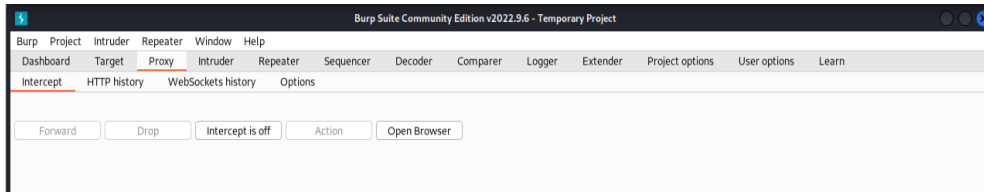
.In addition, Burp Repeater supports session handling, allowing users to maintain the session state while conducting multiple requests. This feature is particularly valuable for testing authenticated functionality and identifying vulnerabilities related to session management.

Burp Repeater offers various features to enhance testing efficiency. Users can save and reload requests, compare different versions of requests and responses, and bookmark important requests for future reference. The tool also supports macro recording and replay, streamlining the process of automating repetitive testing tasks. Furthermore, Burp Repeater supports session handling, allowing users to maintain the session state while performing multiple requests. Burp Repeater also provides a range of features for efficient testing. Users can save and reload requests, compare different versions of requests and responses, and bookmark important requests for later reference. It also supports macro recording and replay, making it easier to automate repetitive testing tasks.

Demo : Burp Repeater

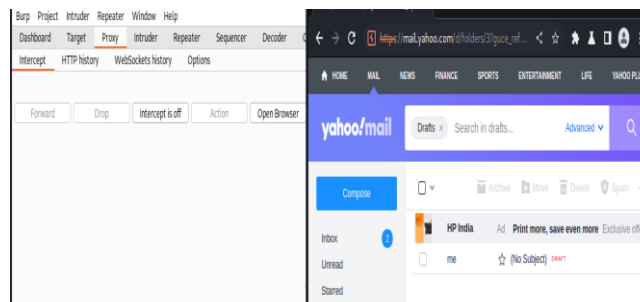
To begin using Burp Repeater, launch the Burp Suite application in Kali Linux and navigate to the Repeater tab. This tab provides an intuitive interface where users can view and manipulate HTTP traffic.

1. Launch Burp's integrated browser



Navigate to the Proxy then Intercept tab within Burp Suite. Toggle the Intercept is off button, changing it to Intercept is on. Enable interception, click the Open Browser option (Launching Burp's integrated browser). This will open Burp's preconfigured browser, designed to seamlessly work with Burp Suite.

2. Intercepting a request



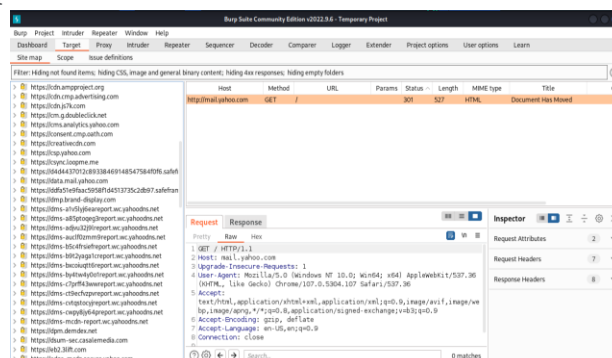
Using Burp's browser, attempt to visit mail.yahoo.com and observe that the site fails to load. Burp Proxy has intercepted the HTTP request sent by the browser before it reaches the server. Users can find this intercepted request in the Proxy then Intercept tab.

Examining an intercepted request in Burp Proxy, the request is held here, allowing users to analyze and modify it before forwarding it to the target server.

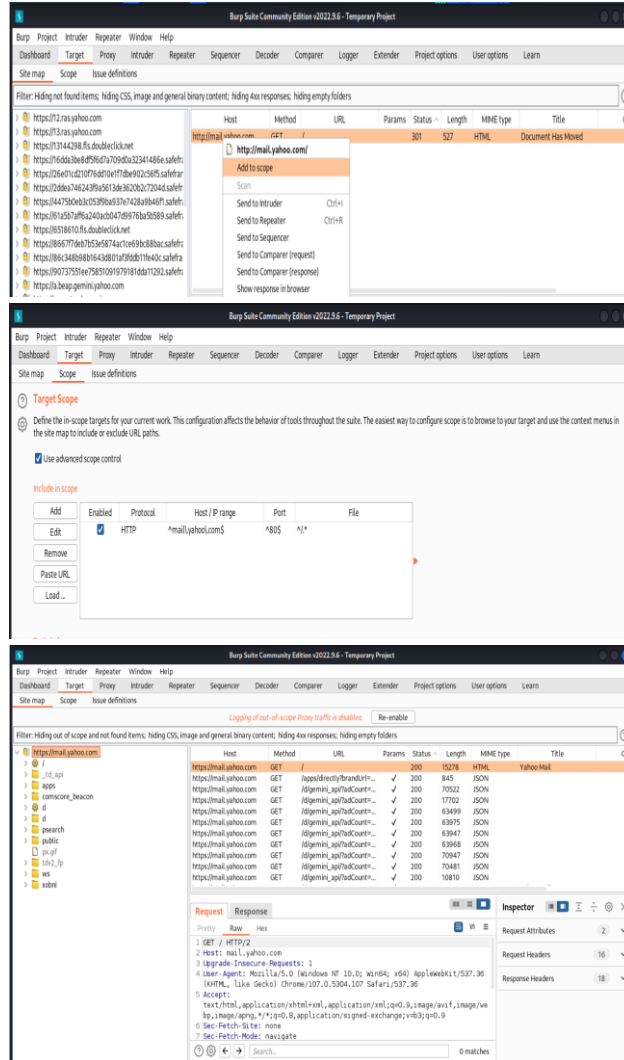
3. Disabling interception

Since browsers often send numerous requests, users may not want to intercept each one. Click the Intercept is on button to switch it to Intercept is off. Proxy Intercept is off. Re-load the page and you will be able to sign into the mail. Within your test email browser try interacting with the user input functions like writing a mail, deleting a mail, making folders etc. All these interaction will be recorded into the target tab's site map.

4. Setting target and scope

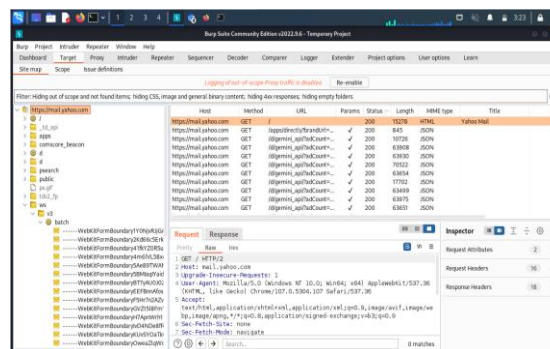


Navigate to target and then site map, there are the list of all the activity and navigation performed over the intercepted browser. Now we can set the scope of targeted requests. Click the entry of the choice, for this demo we'll choose the mail.yahoo.com, right click the request and choose “add to scope”



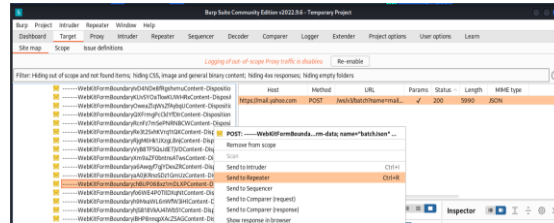
Now you can filter the request which only involves yahoo mail.

5. Navigating the user interaction

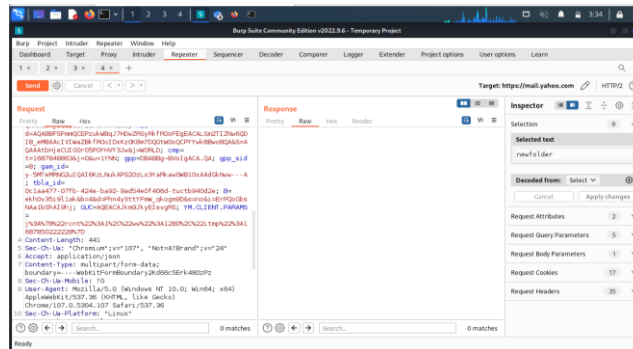


After choosing any of the request navigate to “ws” which has the another folder “v3” within that the “batch” is the folder that has all the user interaction records.

6. Forwarding the request



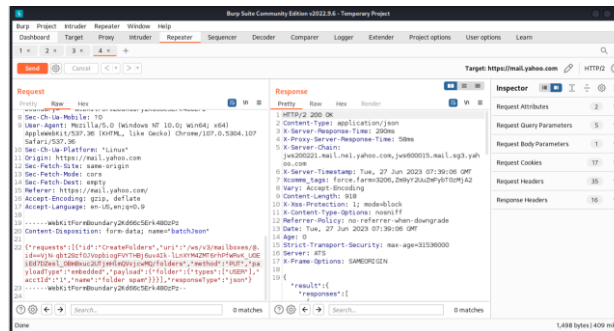
Right click on the request and choose “send to repeater”



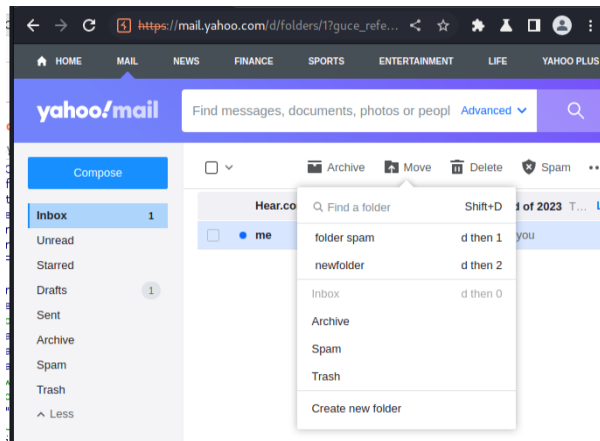
7. Modifying the user request



Within this request we can find that the user created a new folder and named it “new folder” we can change the name of this folder to “folder spam” which will create a new folder with the name mentioned. Click send and we can see that the response of our modified request is shown on the response section.



Go to the browser to see the actual folder that we created by modifying intercepted request from Burp Suite.



3.4 Burp Intruder

Burp Intruder [5] is a powerful tool designed for automated and manual security testing, supporting various attack types such as brute-force attacks and fuzzing. The section introduces the capabilities of Burp Intruder, emphasizing its versatility in generating and manipulating custom HTTP requests for vulnerability discovery. It is designed to automate and streamline the process of performing various types of attacks on web applications, with the goal of identifying vulnerabilities and weaknesses. Intruder allows security testers to customize and automate attacks by defining payloads, specifying attack positions, and configuring various attack options. It supports multiple attack types, including Sniper, Battering Ram, Pitchfork, and Cluster Bomb, each catering to different attack scenarios. With Intruder, testers can manipulate parameters, headers, and other request components to launch attacks like brute forcing, fuzzing, and parameter manipulation. It enables testers to iterate through a list of payloads, sending multiple requests to the target application and capturing the responses for analysis.

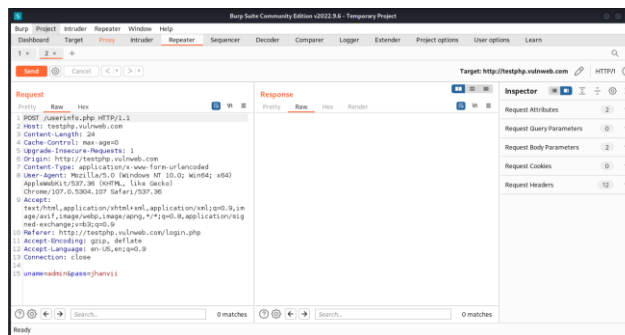
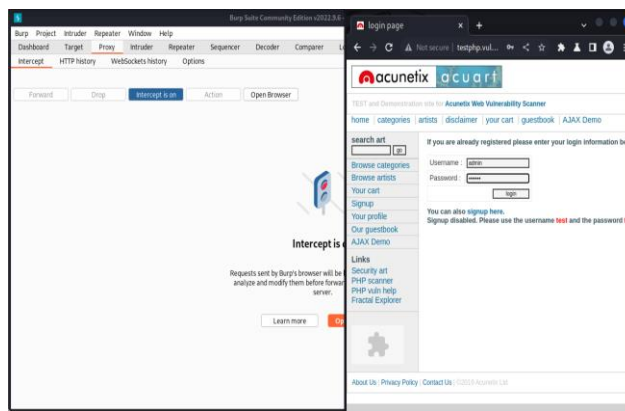
The tool provides extensive options for payload customization, including payload positions, payload lists, payload processing rules, and payload encoding. It also offers response handling options for filtering, marking, and extracting interesting data from the responses.

Intruder offers a flexible and intuitive user interface that allows testers to easily configure and launch attacks, monitor the progress, and analyze the results. It provides detailed insights into the success or failure of each attack attempt, enabling testers to identify potential vulnerabilities or areas of concern within the target application. Using Burp Suite Intruder, security testers can efficiently test the security

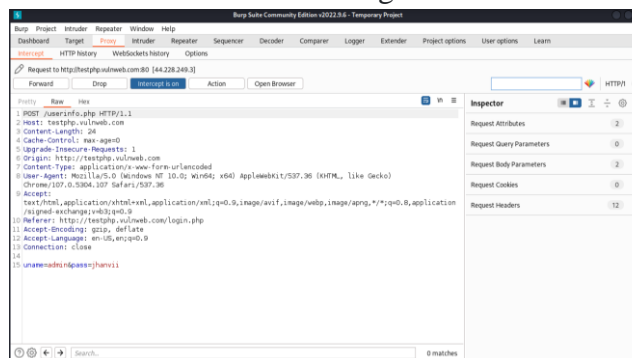
of web applications, uncover potential vulnerabilities, and help organizations improve their overall security posture.

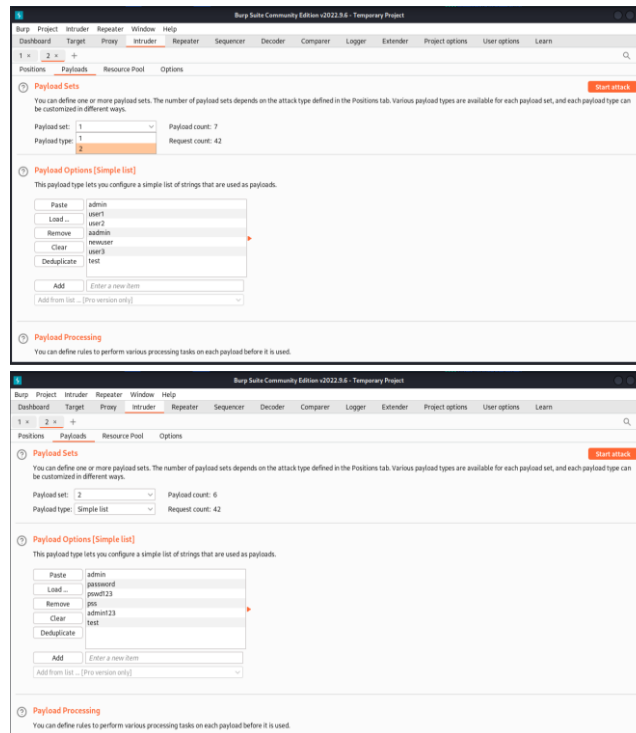
1. Demo : cluster bomb attack

The Burp Suite Intruder tool offers a powerful capability called Cluster Bomb attack [10], which allows for performing a targeted and extensive attack on a specific parameter or input field of a web application. A Cluster Bomb attack is a form of brute force attack that involves generating multiple payloads and iterating through them systematically. The demo guides users through the steps of preparing the request, defining payloads, configuring Intruder settings, launching the attack, and analyzing results.

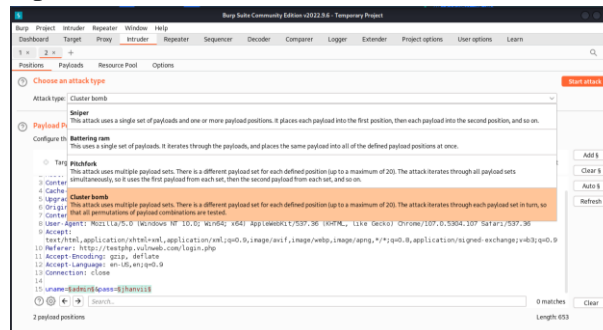


1. Prepare the Request: Start by capturing the target request in Burp Suite Proxy or import it manually. Identify the parameter users want to attack and configure it to be the Intruder target.

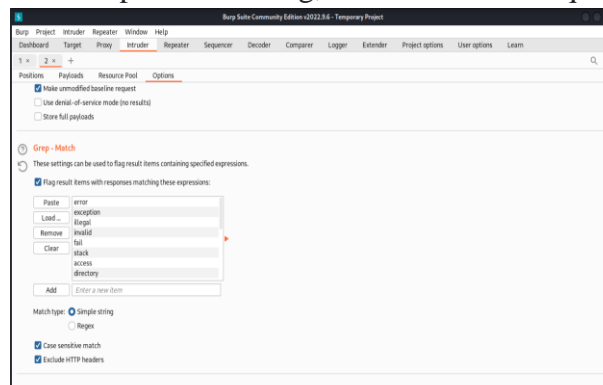




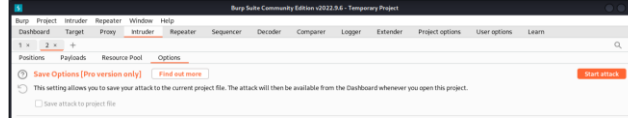
- Define the Payloads: Create a list of payloads that will be used for the attack. These can include common usernames, passwords, or any other relevant data that users want to test. Ensure the payload list covers a wide range of possibilities.



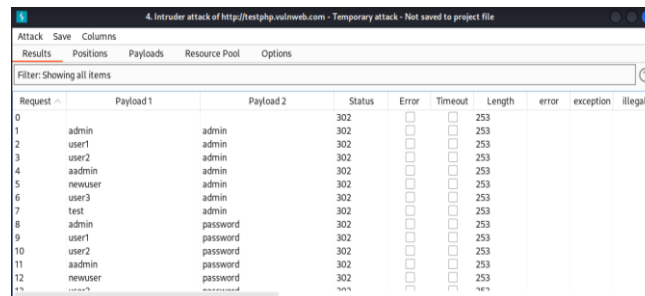
- Configure Intruder Settings: In the Intruder tab, select the Cluster Bomb attack type. Set the payload position to the desired parameter and load the payload list. Customize any other attack options, such as payload processing rules or response handling, based on users requirements.



4. Fine-tune Attack Options: Adjust the attack options as needed. Users can specify the number of threads or iterations to control the attack speed and intensity. It's important to strike a balance between thoroughness and performance to avoid overwhelming the target server.

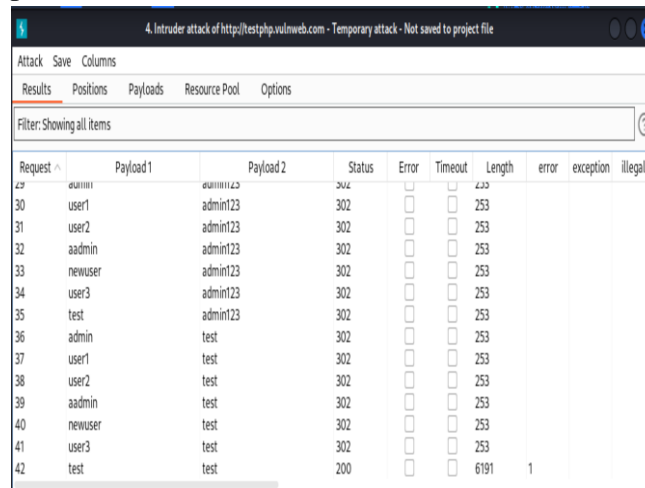


5. Launch the Attack: Start the Cluster Bomb attack and monitor the progress in real-time. Burp Suite Intruder will iterate through each payload combination and send requests to the target, capturing the responses.



Request	Payload 1	Payload 2	Status	Error	Timeout	Length	error	exception	illegal
0			302	<input type="checkbox"/>	<input type="checkbox"/>	253			
1	admin	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
2	user1	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
3	user2	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
4	aadmin	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
5	newuser	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
6	user3	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
7	test	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
8	admin	password	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
9	user1	password	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
10	user2	password	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
11	aadmin	password	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
12	newuser	password	302	<input type="checkbox"/>	<input type="checkbox"/>	253			

6. Analyze the Results: Once the attack is complete, review the responses to identify any successful hits or potential vulnerabilities. Pay attention to any unexpected behaviors or error messages that could indicate a successful exploitation.



Request	Payload 1	Payload 2	Status	Error	Timeout	Length	error	exception	illegal
29			302	<input type="checkbox"/>	<input type="checkbox"/>	253			
30	user1	admin123	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
31	user2	admin123	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
32	aadmin	admin123	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
33	newuser	admin123	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
34	user3	admin123	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
35	test	admin123	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
36	admin	test	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
37	user1	test	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
38	user2	test	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
39	aadmin	test	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
40	newuser	test	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
41	user3	test	302	<input type="checkbox"/>	<input type="checkbox"/>	253			
42	test	test	200	<input type="checkbox"/>	<input type="checkbox"/>	6191	1		

7. Refine and Repeat: Based on the results, refine users' payload list, attack options, or parameters to further optimize the Cluster Bomb attack. Iteratively repeating the process can help uncover additional weaknesses or strengthen the overall security of the web application.

4. CONCLUSION

In conclusion, this research paper has thoroughly examined the significance of Burp Suite tools, with a specific focus on Burp Repeater and Burp Intruder, in the realm of web application security testing. The study has elucidated the practical application of Burp Suite tools through controlled experiments, emphasizing their utility in uncovering vulnerabilities while adhering to OWASP secure coding practices. Burp Repeater emerges as a tool that enhances testing efficiency by providing essential

functions such as request saving, reloading, request and answer comparison, and bookmarking. The paper underscores the value of Burp Repeater in facilitating a streamlined and effective testing process.

Furthermore, the research delves into the capabilities of Burp Intruder, showcasing its role in automating and streamlining various attack types on web applications. This tool proves instrumental in identifying vulnerabilities and weaknesses, offering extensive options for payload customization, response handling, and attack configuration. The multiple attack types supported by Burp Intruder contribute to its versatility and effectiveness in assessing web application security.

The overarching objective of this study is to enhance the understanding and utilization of Burp Suite tools, aligning them with the principles of OWASP secure coding practices. By employing Burp Suite tools, organizations can not only discover vulnerabilities but also enhance their overall security posture and ensure the reliability of their web applications. This research serves as a valuable guide for security practitioners and organizations seeking to leverage Burp Suite tools effectively in the dynamic landscape of web application security testing.

5. REFERENCES

1. Open Web Application Security Project (OWASP). "OWASP Secure Coding Practices Quick Reference Guide." [Online] Available: https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf
2. Open Web Application Security Project (OWASP). "Vulnerability Scanning Tools." [Online] Available: https://owasp.org/www-community/Vulnerability_Scanning_Tools
3. Kali Linux. "Tool Documentation." [Online] Available: www.kali.org/tools/burpsuite/
4. PortSwigger. "Burp Suite documentation." [Online] Available: <https://portswigger.net/burp/documentation>
5. PortSwigger. "Burp Suite Intruder Attacks." [Online] Available: <https://portswigger.net/burp/documentation/desktop/tools/intruder/configure-attack/attack-types>
6. O. B. Fredj, O. Cheikhrouhou, M. Krichen, H. Hamam, A. Derhab. "An OWASP Top Ten Driven Survey on Web Application Protection Methods." Institute of Electrical and Electronics Engineers (IEEE), 2020. [Online] Available: https://www.researchgate.net/publication/346035890_An_OWASP_Top_Ten_Driven_Survey_on_Web_Application_Protection_Methods
7. J. Warsinkse. "CISSP: Certified Information Systems Security Professional." Wiley, 2019. [Online] Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119423300>
8. I. Babincev, D. Vuletic. "Web application security analysis using the Kali Linux operating system." Vojnotehnicki glasnik, 2016. [Online] Available: <https://www.redalyc.org/pdf/6617/661770082010.pdf>
9. Secure Coding. [Online] Available: <https://www.securecoding.org/> [10] "How to Use Burp Suite's Intruder Tool to Pentest Web Apps." MakeUseOf. [Online] Available: <https://www.makeuseof.com/how-to-use-burp-intruder/>
10. "Burp for Beginners: How to Use Repeater." YouTube. [Online] Available: <https://www.youtube.com/watch?v=LfuU5u6C2Sk>

11. D. Assaf. "An Approach Toward Implementing Continuous Security in Agile Environment." Instituto Politecnico do Porto (Portugal), 2022. [Online] Available: <http://hdl.handle.net/10400.22/16233>
12. G. Khawaja. "Web Penetration Testing and Secure Software Development Lifecycle." In Kali Linux Penetration Testing Bible, Wiley, 2021, pp.231-255. [Online] Available: <https://ieeexplore.ieee.org/document/9936716/>
13. A. Al Anhar, Y. Suryanto. "Evaluation of Web Application Vulnerability Scanner for Modern Web Application." 2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST), Yogyakarta, Indonesia, 2021. doi: 10.1109/ICAICST53116.2021.9497831. [Online] Available: <https://ieeexplore.ieee.org/document/9497831/>
14. A. Mohan, G. A. Swaminathan, N. J. Shafana. "Automated Tools and Techniques in Vulnerability Assessment." 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2022. doi: 10.1109/ICSSIT53264.2022.9716474. [Online] Available: <https://ieeexplore.ieee.org/document/9716474/>