Testing and Identifying clickjacking vulnerabilities with AWS and EC2 Instance

Associate professor - DR.K.Suresh[1], Student - Vijayashree Kumar[2], and Student - Gayathri Nambi[3]
Department of Information Technologsy , Sri Venkateswara college of Engineering , Sriperumbadur , 602117, India

Abstract—**This research explores the detection and mitigation of clickjacking vulnerabilities utilizing Amazon Web Services (AWS) and EC2 instances. Through experimentation and analysis, the study aims to identify potential weaknesses in web applications and develop effective countermeasures. By leveraging the scalable infrastructure of AWS and the flexibility of EC2 instances, this work contributes to the advancement of cybersecurity measures in combating clickjacking threats.**

Index Terms—**Clickjacking, Vulnerability Detection, AWS, EC2 Instance, Cybersecurity, Web Applications, Countermeasures, Experimentation, Analysis.**

I. INTRODUCTION

The introduction delves into the exploration of clickjacking vulnerabilities, employing Amazon Web Services (AWS) and EC2 instances.
This study aims to elucidate the significance of these vulnerabilities in contemporary cybersecurity landscapes and proposes methodologies for their detection and identification. By leveraging AWS's scalable infrastructure and EC2's flexibility, this research contributes to enhancing security measures against clickjacking threats in web applications.

II. OVERVIEW

The Clickjacking Vulnerability Checker stands as a pivotal advancement in web security, offering organizations a potent defense against the pervasive threat of clickjacking. Developed using Python and leveraging the Requests library, this command-line utility provides a robust solution for testing and identifying clickjacking vulnerabilities across web domains. It meticulously examines content security policies and HTML content to detect potential vulnerabilities, delivering actionable insights into the security status of web domains. One of its core strengths lies in its versatility and efficiency. Users can effortlessly assess individual URLs or multiple websites concurrently, streamlining the assessment process and optimizing resources. Additionally, the tool seamlessly processes a text file with a list of websites, facilitating batch processing and scalability. Its compatibility with AWS EC2 instances ensures high-performance deployment, enabling swift execution and scalability. Users can retrieve results for multiple domains within seconds, enhancing productivity and response times. Moreover, the Clickjacking Vulnerability Checker integrates continuous monitoring with email alerting, providing real-time updates on detected vulnerabilities. This proactive approach empowers users to stay vigilant and take immediate action, bolstering cybersecurity posture and safeguarding sensitive data . In today's interconnected digital landscape, this tool serves as a critical asset, empowering organizations to proactively identify and mitigate clickjacking threats, thus fortifying their web applications against cyber risks

III. EXISTING SYSTEM

Traditionally, cybersecurity professionals utilize manual techniques or third-party solutions to assess clickjacking vulnerabilities. Manual methods involve tedious processes of inspecting website code and configurations, making it time-consuming and prone to human error. Third-party services, while providing convenience, may lack transparency in their scanning processes, raising concerns about data privacy and reliability. Moreover, reliance on external services may incur additional costs and dependencies.

Disadvantages

• Limited Scalability

• Limited Feedback

• Long processing time

IV. PROPOSED SYSTEM

The developed command-line tool offers a comprehensive solution by automating the detection of clickjacking vulnerabilities. By integrating with the request library, the tool scans websites, enabling identification of vulnerabilities. Its command-line interface enhances usability, allowing users to easily input domains for assessment. The tool's ability to save results to a text file facilitates record-keeping and analysis. By deploying the tool on an AWS EC2 instance, users benefit from scalable computing and reliable, ensuring timely and accurate vulnerability assessments.

Advantages

•Enhanced Reporting

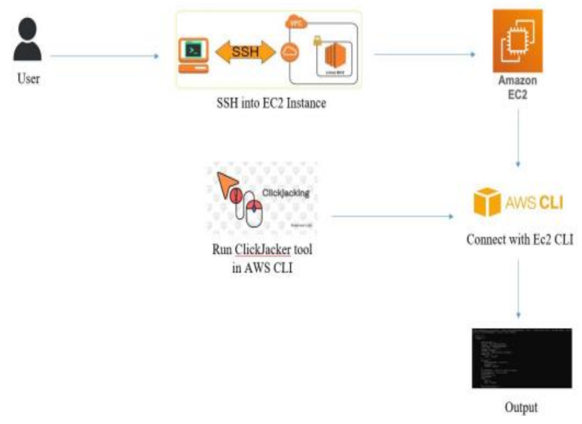
• Can check multiple URLs

• Short processing time

V. HARDWARE AND SOFTWARE CONFIGURATION

HARDWARE CONFIGURATION:
• Processor - i5
• Speed - 3 GHz
• RAM - 8 GB(min)
• Hard Disk - 500 GB
• Key Board - Standard Windows Keyboard
• Mouse - Two or Three Button Mouse
•Monitor-SVGA
SOFTWARE CONFIGURATION:
•Operating System: Linux, Windows/7/10
• Tools: vs code
• Server side Script: Python

VI . SYSTEM ARCHITECTURE



VII. MODULE DESIGN SPECIFICATION

MODULE DESCRIPTION
The command-line tool for identifying clickjacking vulnerabilities leverages several key modules to enhance functionality and usability. The requests module provides seamless HTTP communication, enabling your tool to retrieve webpage content for vulnerability assessment. It simplifies the process of making GET requests to URLs provided by users, ensuring efficient data retrieval. To streamline user interactions, the argparse module offers a robust interface for parsing command-line arguments. This allows users to specify single or multiple website URLs, 20 as well as text files containing lists of websites, making the tool adaptable to various scenarios. With argparse, your tool offers flexibility and ease of use, enhancing user experience. Ensuring system reliability, the os module facilitates file handling operations, verifying the existence of specified files before accessing them. By preemptively checking file paths, it prevents potential errors, contributing to the robustness and stability of your tool. Moreover, the integration of ANSI escape codes enables the presentation of colored text in the terminal, enhancing readability and user comprehension. These codes provide clear visual indicators of clickjacking vulnerability status, empowering users to interpret results swiftly and take informed actions. Collectively, these modules form the foundation of your command-line tool, enabling efficient clickjacking vulnerability assessment and empowering users with actionable insights into website security.

EC2 Instance
It is important to deploy your command-line tool on an AWS EC2 (Elastic Compute Cloud) instance to harness its benefits. EC2 instances provide scalable and resizable compute capacity in the cloud, allowing you to host applications without the need for physical hardware. By choosing an EC2 instance with ample vCPU and RAM resources, you ensure optimal 21 performance for your tool, especially when handling multiple domains simultaneously. Deploying your tool on an EC2 instance offers accessibility, allowing users to interact with it remotely via SSH or other methods. Users can execute commands in the terminal to assess the vulnerability of domains to

clickjacking, receiving prompt responses from the tool running on the EC2 instance. Overall, leveraging AWS EC2 for your command-line tool deployment enhances its performance, reliability, and accessibility, aligning with your goal of efficiently detecting clickjacking vulnerabilities across multiple domains.

CLI (Command Line Interface)
The Command Line Interface (CLI) serves as the primary interface for users to interact with your clickjacking vulnerability detection tool. By leveraging the CLI, users can seamlessly access the tool's functionality directly from their terminal or command prompt environment, without the need for a graphical user interface. This CLI-driven approach offers several advantages, including ease of use, flexibility, and efficiency. With the CLI, users can perform a variety of tasks related to clickjacking vulnerability detection with simplicity and convenience. They can initiate vulnerability scans for single domains by providing the domain's URL as an argument to the CLI command. Additionally, users have the flexibility to analyze multiple domains simultaneously by supplying a list of URLs as arguments, streamlining the assessment process for efficiency. Moreover, the CLI enables users to save the results of vulnerability assessments to text files for further analysis or documentation. By specifying a file path using the appropriate CLI option, users can store the scan results for future reference or sharing with other team members, enhancing collaboration and knowledge sharing within the organization.

Argument Parsing Module (argparse)
By incorporating argparse into your project, you enable users to interact with your tool effortlessly, issuing commands and providing input parameters directly from the command line. The module allows you to define a set of command-line options and arguments, along with their respective descriptions, making it intuitive for users to understand the available functionalities and how to utilize them effectively. With ‘argparse’, users can leverage various command-line options to customize the behavior of your tool according to their specific requirements. For example, users can specify a single domain to check for vulnerabilities using the ‘-s’ or ‘—single’ option, or they can provide multiple domains separated by spaces to scan multiple domains simultaneously using the ‘- m’ or ‘—multiple’ option. Additionally, users have the flexibility to supply a text file containing a list of domains using the ‘-f ’ or ‘—file’ option, enabling them to perform vulnerability scans on large sets of domains efficiently. Overall, by leveraging the capabilities of the ‘argparse’ module, you enhance the usability, flexibility, and robustness of your clickjacking vulnerability detection tool, empowering users to efficiently assess and mitigate clickjacking vulnerabilities in their web applications with ease.

Requests Module
The requests module allows your tool to perform HTTP GET requests to the specified URLs, enabling it to fetch the web pages' content programmatically. This capability is essential for scanning multiple domains for clickjacking vulnerabilities, as it automates the process of accessing and analyzing web page data without manual intervention. Moreover, the requests module provides robust error handling 23 mechanisms, allowing your tool to gracefully manage various networkrelated exceptions, such as connection errors, timeouts, and invalid URLs. By handling these exceptions effectively, your tool can provide informative error messages to users, guiding them on how to resolve connectivity issues or input errors. Overall, the requests module serves as a foundational component of your clickjacking vulnerability detection tool, enabling it to retrieve, analyze, and interpret web page data efficiently. By leveraging the capabilities of the requests module, your tool can provide users with actionable insights into the security posture of web applications, empowering them to mitigate clickjacking risks effectively

VIII . PERFORMNCE ANALYSIS

RESULT AND DISCUSSION

Data Collection
In order to check website URLs for clickjacking vulnerabilities, users must provide information about them throughout the data collection process. There are various ways to go about this: A single website URL is entered by users using the '-s' or '--single'

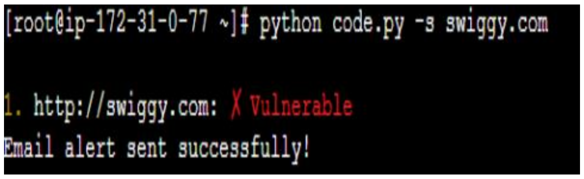
option as a command-line input. This URL is retrieved by your script, which then looks for any clickjacking weaknesses. Many URL Input: Inputters can use the '-m' or '--many' option to specify several website URLs as command-line arguments, separated by spaces. Each URL is checked individually by your script as it loops through them all. In each instance, your script uses the requests library to send HTTP requests to the specified URL or URLs after reading the supplied URL.

Data processing
If any vulnerabilities are found, the program produces results that are easy to understand and tell whether or not each URL is susceptible. Color coded text is used to improve reading and comprehension when presenting these results to the user in an easy-to-use style. Furthermore, the tool allows for the flexibility to process numerous URLs at once, allowing users to evaluate the security status of different domains. In order to promote proactive security, this functionality helps customers to monitor and manage the security posture of their web assets over time. The tool supports continuous monitoring of web assets, facilitating proactive security practices. In case of detected vulnerabilities, the tool triggers email alerts to notify users promptly, enabling timely response and mitigation.

IMPLEMENTATION

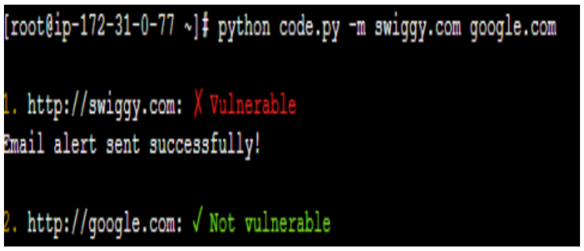
The output of the script is intended to be understandable and clear. In order to improve readability, it uses ANSI escape codes for colored text, highlighting sensitive areas in red and non-vulnerable areas in green. To aid in the rapid identification of results, every website entry has a yellow bullet point added to it. Moreover, the script provides an extra functionality that allows users to store the URLs of susceptible websites in a designated text file for later analysis. By giving a filename with the -v or —save flag, users can activate this feature and make it simple to retrieve sensitive site data for later use.

SINGLE URL



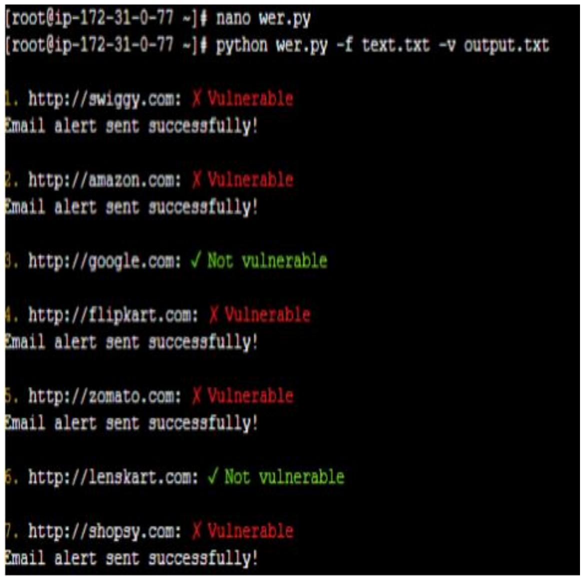
```
[root@ip-172-31-0-77 ~]# python code.py -s swiggy.com

1. http://swiggy.com: ✗ Vulnerable
Email alert sent successfully!
```

DOUBLE URL



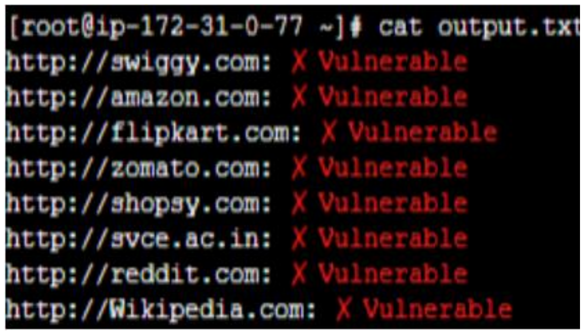
```
[root@ip-172-31-0-77 ~]# python code.py -m swiggy.com google.com

1. http://swiggy.com: ✗ Vulnerable
Email alert sent successfully!

2. http://google.com: ✓ Not vulnerable
```

MULTIPLE URL



```
[root@ip-172-31-0-77 ~]# nano wer.py
[root@ip-172-31-0-77 ~]# python wer.py -f text.txt -v output.txt

1. http://swiggy.com: ✗ Vulnerable
Email alert sent successfully!

2. http://amazon.com: ✗ Vulnerable
Email alert sent successfully!

3. http://google.com: ✓ Not vulnerable

4. http://flipkart.com: ✗ Vulnerable
Email alert sent successfully!

5. http://zomato.com: ✗ Vulnerable
Email alert sent successfully!

6. http://lenskart.com: ✓ Not vulnerable

7. http://shopsy.com: ✗ Vulnerable
Email alert sent successfully!
```

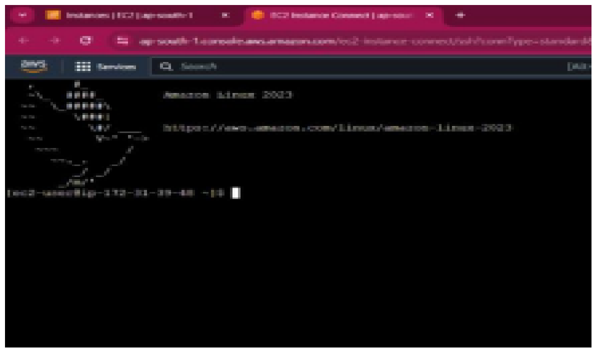
DISPLAYS ONLY VULNERABLE URLS



```
[root@ip-172-31-0-77 ~]# cat output.txt
http://swiggy.com: ✗ Vulnerable
http://amazon.com: ✗ Vulnerable
http://flipkart.com: ✗ Vulnerable
http://zomato.com: ✗ Vulnerable
http://shopsy.com: ✗ Vulnerable
http://svce.ac.in: ✗ Vulnerable
http://reddit.com: ✗ Vulnerable
http://Wikipedia.com: ✗ Vulnerable
```

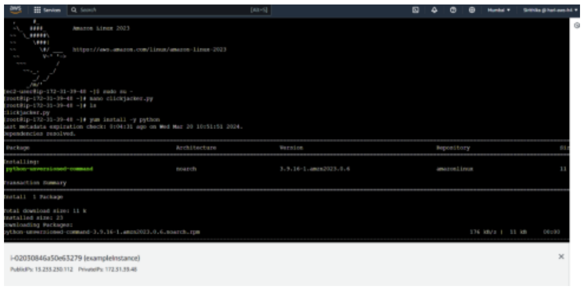
IX . APPENDICES

1.COMMAND LINE INTERFACE



```
Amazon Linux 2023

https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-39-48 ~]$
```

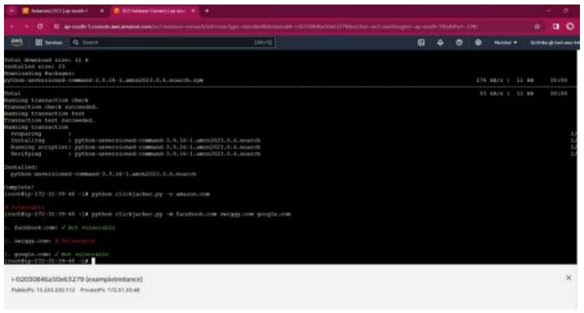
2.INSTALLED CLICKJACKER TOOL



```
       Amazon Linux 2023

       https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-39-48 ~]$ sudo su -
[root@ip-172-31-39-48 ~]# nano clickjacker.py
[root@ip-172-31-39-48 ~]# ls
clickjacker.py
[root@ip-172-31-39-48 ~]# yum install -y python
Last metadata expiration check: 0:04:31 ago on Wed Mar 20 10:51:51 2024.
Dependencies resolved.
================================================================================
Package                      Architecture   Version                 Repository
================================================================================
Installing:
python-unversioned-command   noarch         3.9.16-1.amzn2023.0.6   amazonlinux     11

Transaction Summary
================================================================================
Install  1 Package

Total download size: 11 k
Installed size: 23
Downloading Packages:
python-unversioned-command-3.9.16-1.amzn2023.0.6.noarch.rpm    176 kB/s |  11 kB     00:00
```

i-02030846a50e63279 (exampleInstance)
PublicIPs: 15.233.230.112 PrivateIPs: 172.31.39.48

3.OUTPUT



```
Total download size: 11 k
Installed size: 23
Downloading Packages:
python-unversioned-command-3.9.16-1.amzn2023.0.6.noarch.rpm    176 kB/s |  11 kB     00:00
--------------------------------------------------------------------------------
Total                                                           93 kB/s |  11 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Installing       : python-unversioned-command-3.9.16-1.amzn2023.0.6.noarch
  Running scriptlet: python-unversioned-command-3.9.16-1.amzn2023.0.6.noarch
  Verifying        : python-unversioned-command-3.9.16-1.amzn2023.0.6.noarch

Installed:
  python-unversioned-command-3.9.16-1.amzn2023.0.6.noarch

Complete!
[root@ip-172-31-39-48 ~]# python clickjacker.py -u amazon.com

✗ Vulnerable
[root@ip-172-31-39-48 ~]# python clickjacker.py -m facebook.com swiggy.com google.com

1. facebook.com: ✓ Not Vulnerable

2. swiggy.com: ✗ Vulnerable

3. google.com: ✓ Not Vulnerable
[root@ip-172-31-39-48 ~]#
```

i-02030846a50e63279 (exampleInstance)
PublicIPs: 15.233.230.112 PrivateIPs: 172.31.39.48

X . CONCLUSION

In conclusion, the development of the clickjacking vulnerability checker command-line tool represents a significant step towards enhancing web security. By leveraging the capabilities of the requests module and the argparse library, the tool provides users with a convenient and efficient means of assessing the susceptibility of web domains to clickjacking attacks. Moreover, the integration of continuous monitoring and email alerting functionalities adds proactive security measures to the tool. Users can set up continuous monitoring to regularly check for clickjacking vulnerabilities in specified URLs, and the tool automatically sends email alerts upon detecting any vulnerabilities. This real-time alerting mechanism ensures prompt notification and enables users to take swift action to mitigate potential risks. The tool's ability to analyze individual URLs, multiple domains, or entire lists of websites demonstrates its versatility and scalability. The integration of ANSI escape codes for colored text enhances the user experience by providing clear and visually appealing vulnerability status indicators. Deploying the tool on an Amazon EC2 instance further enhances its performance and scalability, allowing it to handle large volumes of requests efficiently. The choice of AWS EC2 ensures reliability and 30 availability, crucial factors for a security-focused application. Overall, the clickjacking vulnerability checker tool with continuous monitoring and email alerting capabilities offers comprehensive security assessment and proactive risk management for web applications.

REFERENCE

[1] Rehman, Ubaid Ur & Khan, Waqas & Saqib, Nazar Abbas & Kaleem, Mohammad. (2013). On Detection and Prevention of Clickjacking Attack for OSNs. 160-165. 10.1109/FIT.2013.37.

[2] Simic, I. (2022) Host header - what is an HTTP host header injection?, Crashtest Security. Crashtest Security.Availableat:https://crashtestsecurity.com/invalid-host-header/ (Accessed: December 21, 2022)

[3] Panjwani, S., Tan, S., and Jarrin, K. M. (2005) An experimental evaluation to determine if port scans are precursors to an attack. Proceedings of DSN'05, Washington, DC, USA, June 28–July 1, pp. 602–611. IEEE Computer Society

[4] K. D. Tandale and S. N. Pawar, "Different Types of Phishing Attacks and Detection Techniques: A Review," 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), 2020, pp. 295-299, doi:10.1109/ICSIDEMPC49020.2020. 9299 624.

[5] V. Zlomislić, K. Fertalj and V. Sruk, "Denial of service attacks: An overview," 2014 9th Iberian Conference on Information Systems and Technologies (CISTI), 10.1109/CISTI.2014.6876979. 2014, pp. 1-6, doi

[6] Dipti Pawade; Divya Reja; Abhilasha Lahigude Era Johr

[7] E. Woollacott, "Facebook account takeover: Researcher scoops 40k bug bounty for chained exploit,"2022.[Online].Available: https://portswigger.net/daily-swig/facebook-account-takeover researcher-scoops-40k-bug-bounty-for-chained-exploit.

[8] J. Walker, "Teen hacker scoops 4,500 bug bounty for Facebook flaw that allowed attackers to unmask page admins," 2021. 34 [Online]. Available: https://portswigger.net/ daily-swig/teen- hacker-scoops-4--500-bug bounty-for-facebook-flaw-that-allowed-attackers-to-unmask-page admins

[9] P. X. Mai, F. Pastore, A. Goknil, and L. C. Briand, "MCP: A security testing tool driven by

requirements," in Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., 2019, pp. 55–58.
[10] D. Deahl, Another Facebook vulnerability, 2018. [Online]. Available: https://bit.ly/3gtPo80 Available: https://github.com/portswigger/burp-csj
[11]OWASP,WSTGATHZ01:Testingdirectory traversalfileinclude,2020. [Online]. Available: https://bit.ly/3l1sRTl
[12]V.J.Manèsetal.,"Theart,science,andengineeringof fuzzing: Asurvey," IEEE Trans.Softw. Eng., vol. 47, no. 11, pp. 2312– 2331, Nov. 2021.
[13]OWASP,WSTG-SESS 03:Testing for session fixation, 2020.[Online]. Available: https: //bit.ly /34myBkN
[14] Common attack pattern enumeration and classification(CA PEC), 2021.[Online]Available: https://capec.mitre.org
[15] Java Platform, Enterprise Edition, 2021. [Online]. Available:https://www.oracle.com/java/technologies/ java-ee-glance.html
[16] S. Elder et al., "Do I really need all this work to findvul nerabilities?,"Empirical Softw. Eng., vol. 27, no. 6, 2022, Art. no. 154. [Online].Available: https://doi.org/10.1007/s10664--022- 10179-6
[17] OWASP, OWASP zed attack proxy, 2021. [Online]. Available: https:// www.zaproxy.org
[18] SonarSource, Sonarqube: Code quality and code security toolset 2021. [Online]. Available: https:// www.sonarqube.or