Volume: 11 Issue: V Month of publication: May 2023

DOI: https://doi.org/10.22214/ijraset.2023.52962

Harbinger: A Toolkit for Vulnerability Testing of IoT Devices and Web Clients

Sonali Patil[1], Mandaar Rao[2], Lavitra Misal, Darshan Phaldesai[3], Kishor Shivsharan[7]
[1]*Professor, Pimpri Chinchwad College of Engineering,* India
[2, 3, 4]*Student, Pimpri Chinchwad College of Engineering,* India

Abstract: The need for efficient tools to identify and mitigate vulnerabilities has never been more critical due to the complexity and proliferation of online applications and embedded systems. We introduce a Python toolkit in this study that combines a variety of security testing and analysis methodologies, allowing users to locate and fix web application vulnerabilities and examine firmware files. Our toolkit offers aggregated reports summarizing the outcomes of firmware analysis and web application vulnerability testing, in addition to being adaptable and user-friendly so customers may change and add features to meet their unique needs. We provide the findings from testing and analysis carried out on a variety of online apps and firmware files, as well as the design and implementation of our tools. The integration of additional security testing tools, compatibility for other file formats, and the introduction of machine learning-based analysis are just a few of the future directions we could take our toolset. This Python toolkit makes a significant addition to the realm of cybersecurity by offering customers a strong and adaptable solution for web application vulnerability testing and firmware analysis.

I. INTRODUCTION

The need for strong security testing and firmware analysis tools has been emphasized by the rising occurrence of cyber attacks and security breaches. To address this need, we have developed a Python toolkit that offers an all-encompassing solution. This toolkit empowers users to test URLs for vulnerabilities and examine firmware files. It consists of two modules, each containing various functions that perform distinct duties related to their intended purposes.

With the first module of our toolkit, users can test a web service or web application for a range of vulnerabilities. The outcomes of these evaluations are aggregated into a clear and comprehensible final report. The second module, on the other hand, facilitates the analysis of firmware files by extracting the root file system and searching for hardcoded passwords, significant binaries, and SSL certificates. This helps users discover potential vulnerabilities and gives them a more thorough understanding of the firmware files.

The toolkit we have developed offers users a user-friendly interface that allows for extensive security testing and firmware analysis. It streamlines the process of identifying potential vulnerabilities and fortifying systems, making it more accessible for users.

The objectives of the toolkit can be broken down as follows:

1) To develop an intuitive and straightforward interface for security testing and firmware analysis.
2) To facilitate users in conducting vulnerability tests on URLs using well-known security testing tools, such as Arachni and ZAProxy.
3) To empower users to scrutinize firmware files, including extracting the root file system, utilizing the Firmwalker tool.
4) To consolidate the findings of the security testing and firmware analysis into an easy-to-comprehend final report.
5) To enable users to personalize the output configuration of the final report based on their preferences.

II. LITERATURE REVIEW

We present some of the related works where similar approaches were used for vulnerability testing of Iot Devices and Web Clients.

This paper discusses the security concerns raised by the Internet-of-Things paradigm. It provides a complete categorization of extremely reliable surveys that cover different facets of the IoT paradigm. It defines IoT vulnerabilities, attack vectors, security objectives impacted, attacks that exploit such vulnerabilities, related remedial procedures, and currently available operational cyber security tools to infer and monitor such weaknesses. Data from more than 100 IoT-specific works published between 2005 and 2018 were included in the report. It provides us with a variety of vulnerabilities related to IoT and other dimensions. It discusses issues such as inadequate physical security, which occurs when IoT devices operate autonomously in unattended environments, making it possible for an adversary to take control of the device.

It also discusses how an attacker could drain a device's stored energy by sending out a flood of messages or corrupted messages. Weak programming techniques, incorrect encryption, inappropriate open ports, inadequate access control, and insufficient audit measures are all covered.[1]

This paper discusses how the Internet of Things (IoT) aims to provide intelligent services but must be secure. It discusses how the devices have low computing power, which makes ensuring security difficult. As a result, IoT devices are a target for cyber-attacks. The paper discusses the Internet of Thing's security flaws and security challenges. It gives us a detailed implementation and analysis of IoT attacks, as well as some of their solutions.

The paper lists IOT-specific risks like common Information technology and IoT worms, "Script Copies," racketeering (exposure to intellectual property, thwarting and intelligence gathering), cyberwarfare (traffic monitoring), and Bluetooth connection risks such as the BlueBorne attack vector. The paper then proposes security solutions to these issues, such as Authentication, Authorization, Network Policy, and Analysis, such as visibility and control. The following section focuses on practically implementing IoT attacks such as Wi-Fi-oriented attacks and BlueSnarfer attacks.[2]

The paper describes a hopping routing protocol for secure IoT device transmission. The protocol enables IoT devices to identify themselves before joining an existing network or forming their own. To improve security, it employs multi-layer parameters. This protocol was put through field tests, and the results showed that it was suitable for IoT communication. It addresses IoT device interconnectivity and security. It also includes experiments and tests of the newly introduced SMRP.

The capabilities of their SMRP include the ability to separate devices according to the UCID of the applications, automatically re-creating logical networks within a net of devices, network creation and authentication occurring concurrently, and resilience to hacker attacks by constantly shifting embedded secure information. The SMRP protocol suite is then explained by describing its various layers. It also explained how the protocol works by utilizing standard routing procedures such as the HELLO procedure. The paper concludes by displaying the results, discussions, and field tests.[3]

In November 2022, the authors put forward a study of cross site scripting (XSS). The paper also discusses issues related to standardisation, offers safety measures for input verification and output coding, explains output coding mode while adhering to blacklist verification mode's restrictions, and gives examples of how to enter untrusted data outside of the permitted position, HTML code these data when inserting them between HTML tags, and encode HTML attributes before inserting untrusted data into common HTML attributes. [4]

Websites built with ASP.NET may have security flaws that are hidden from the website owner. This study outlines a technique for identifying security flaws in ASP.NET-built websites. The proposed algorithm does a scan of all files hosted on the IIS. The scanner tool analyses the application's code, which is made using the ASP.NET package and the code that supports the files. A program designed for this intention generates a report which describes the majority of leaks and vulnerability types (by listing the files, the security issue with description, and where it is located). The proposed algorithm will assist organizations in addressing vulnerabilities and improving overall security.[5]

Developing software and web apps are now essential parts of the daily lives of common people. Almost all people use such applications to talk to each other, gather knowledge, and conduct payments. However, these people are at risk because of the coder's lack of experience with protocols relating to security. Despite extensive research into security while on the web and protection against hacking, there exist numerous vulnerable websites. This paper, put forward by the authors from the National University of Colombia, examines three major hacking techniques: Cross Server Scripting, Cross-Site Request Forgery, and SQL Injection on a significant sample of Colombian websites. The goal was to learn how the businesses and organizations in Colombia prioritize (or not) security, and how this affects the end user.[6]

Websites now take up a larger portion of our time. Major exercises are being conducted to ensure the security issues in these applications. Website attacks such as brute force and injection are now a commonplace in recent years. Many attacks operate in real-time. The majority of ways try to focus on preventing and detecting these attacks on websites. The goal of the work created by Himanshi Singh and Mohit Dua from NIT Kurukshetra, Kurukshetra, India, differs from current web application safety research. Website threats like SQL injection and cross-site scripting are the main focus of the effort. The developed tool demonstrates to security analysts and programming students how to identify problems with online applications. The XAMPP server was employed for both the server and client environments in this work. [7]

This paper presents an effective web vulnerability scanner that integrates information collection with vulnerability detection to assess the security of web applications. The majority of current scanners could be more efficient since they only scan a single target.

This problem is addressed by the suggested scanner, which combines data gathering, port scanning, and vulnerability detection. It provides comprehensive and precise detection abilities, including scanning for widespread flaws like SQL injection, File upload vulnerability, and cross-site scripting (XSS). The scanner was created using the Python programming language and runs on a B/S architecture. The test results show that it performs better than OWASP ZAP in finding more vulnerabilities. By using the gathered data to direct the detecting process, the suggested scanner achieves a considerable scanning scope and performance.[8]

In order to discover authentication bypass vulnerabilities in IoT-embedded binary servers, the paper evaluates current methodologies for finding flaws in IoT firmware and proposes a completely original strategy based on fuzzing and static analysis.

The study groups previously published works into various methodologies, including static analysis, symbolic execution, fuzzing on emulators, and thorough testing. Within each category, various tools and frameworks are presented, emphasising their benefits and drawbacks with regard to the identification of IoT firmware vulnerabilities.

Comparing the ratios of various vulnerabilities in various targets helps address the specificity of vulnerability detection in IoT firmware. The content includes a table that displays the prevalence of several vulnerability types (such as XSS, injection, authentication bypass, and overflow) across a range of targets, including WordPress, phpMyAdmin, Apache, routers, webcams, firmware, Excel, Linux, and Android.It focuses on that because there are web interfaces implemented by binary CGIs, IoT firmware frequently has logical bugs and memory problems.

The suggested approach for locating authentication bypass vulnerabilities in binary servers embedded in Internet of Things devices makes use of static analysis and fuzzing. In order to create test cases for the dynamic fuzzer, the technique involves employing a static analyzer to extract keywords from the target binary web server. The fuzzer then reviews the responses to determine whether an authentication bypass flaw has been activated. By checking for known authentication bypass issues, the usefulness of the suggested method is confirmed.[9]

The paper discusses how embedded systems are becoming more prevalent in our society and how their security is becoming into a significant issue as a result. It discusses the findings of a number of recent assessments that revealed embedded systems have developed a reputation for being insecure. Even though individuals are aware of the problems with embedded systems, they are ignorant of their security. The analysis, which is the first large-scale examination of firmware pictures to be made public, is presented in the publication. The paper decompressed 32,000 firmware images into 17% million distinct files and statically examined each one. According to the article, they found 38 previously unidentified vulnerabilities in more than 600 firmware images. The paper further correlated these files with unrelated firmware images and extended the vulnerabilities to over 123 other products. According to the study, at least 1.4 lakh internet-accessible devices are impacted by these vulnerabilities.[10]

This paper talks about the programming language Lua. Lua is an embeddable, interpreted and cross-platform language that is used to make production web applications. The paper states that Lua is rising in popularity because of its simple design and low usage of resources. The Lua codebase, according to the report, has enormous potential and may one day power both production-level servers and a sizable number of devices. The lack of static analysis techniques for locating susceptible code in Lua is then discussed in the article. According to the study, Lua is not supported by the majority of widely used vulnerability detection tools. The Static Analysis for Security Testing tool for Lua code, which focuses on online vulnerabilities, is ultimately presented in the paper. By using straightforward and occasionally purposefully exposed Lua code, the article demonstrates the possibilities of its code.[11]

The paper talks about how companies are in the process of finding solutions for the detection of the increasing amount of malware that is being sent around the internet. The paper states that anti-malware developers usually use the unique signature of the payload as a signal to find malware. Other developers look at the communication channels. The paper states that research has been done on the information that HTTP headers can provide. Different properties of the headers have been studied to check if there is malware such as the size of the headers, type errors or absence of some headers. The goal of the paper was to check whether HTTP headers can be used to determine the presence of malware. The paper talks about how different websites have their own header order that request special services or offer specific content. The paper finally concludes that it is unfeasible to use the header order to reliably detect malware.[12]

III. PROPOSED SYSTEM

A. Methodology

The proposed system is a Python toolkit that offers users a comprehensive solution for conducting security testing and firmware analysis. The toolkit is composed of two modules, each containing multiple functions that perform specific tasks related to their respective purposes.

Navigating to the directory and launching the python file in the terminal is how the toolkit is launched. The toolkit starts and asks whether the user wants to proceed with the Web Application Vulnerability Scanning or Firmware Analysis.

If the user chooses the prior option, the toolkit asks the user for information such as the URL to be scanned, the types of vulnerabilities it wants to test, and the strength of the attacks. After the user is done providing these inputs to the toolkit, it first gathers intel about the web server which is hosting the URL. Then, it starts the testing of the user-given url which usually takes some time.

The toolkit also creates an extensive audit report after the scan is finished, which includes information about the web server as well as a detailed list of vulnerabilities it discovered there, along with the affected URLs.

The latter option prompts the user to enter the location to a.bin file, which is typically the firmware file, if that option is chosen. The root file system is extracted from the firmware file after the toolkit has scanned it. Further investigation of this Root File System is possible to look for sensitive information such as SSL certificates, hardcoded passwords, shadow files, and crucial binaries. Additionally, a text file containing all of the crucial details about the scanned firmware is generated.

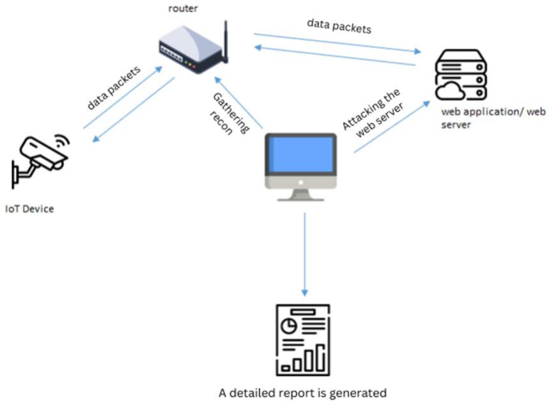

Figure 1 Working Methodology

Our Python toolkit has been specifically designed to cater to a diverse group of users, ranging from security professionals to firmware developers, and anyone else who requires security testing or firmware analysis. Our system is highly customizable and can be tailored to meet the needs of users that have different levels of technical expertise.

The target audience for our Python toolkit includes a diverse range of users that have different levels of technical expertise. The toolkit is designed to be flexible and customizable, making it suitable for:

1) Penetration testers who need to perform security testing and identify potential vulnerabilities in URLs to evaluate potential security risks.
2) Security researchers who need to perform firmware analysis and identify potential security flaws or vulnerabilities in order to improve the security of smart devices.
3) Firmware developers who need to ensure that their products are secure and free of vulnerabilities.
4) IT professionals who are responsible for ensuring the security of their organization's infrastructure and need a comprehensive tool for testing and analysis.
5) Students or individuals who are interested in learning more about security testing and firmware analysis and want a tool that is user-friendly and easy to use.

IV. IMPLEMENTATION

A. Libraries Used

1) Web Scraper using Beautiful Soup

Data extraction, copying, and filtering are all just parts of the scraping process. Web-scraping is the process of obtaining data or feeds from the internet (such as from web pages or websites).

2) Web Crawler using UrlParse

In many cases, URLs contain crucial information that can be used to assess a website, a participant's search, or the distribution of the content in each area. Python includes a number of useful libraries that enable you to parse URLs and get their component elements and UrlParse is one such library.

3) Extracting Text using Regular Expressions or Regex

A regex, which is based on a predetermined pattern, is a powerful tool for text matching. It may determine whether a pattern is present or absent by comparing a text to a particular pattern. Additionally, it can split a pattern into one or more smaller patterns. Its primary function is to provide a search, which calls for a string and a regular expression. In this instance, it either returns the first match if any at all.

4) Making HTTP Requests using requests Library:

The Requests library is one of the crucial parts of Python for delivering HTTP requests to a specified URL. Using the HTTP library Requests, which is licensed under the Apache2 standard, Python users can send HTTP/1.1 requests.

The first module initiates by requesting the user to input a URL. Once the URL is provided, a function is executed to gather intelligence on the web server. The server's critical details such as its physical location, IP address, used technology, and port number are extracted and saved in a JSON file called 'ww.json.'

Subsequently, another function checks the provided URL for possible vulnerabilities. After completing the security test, the results are saved in a JSON file named 'test.json.'

To summarize the outcomes of the security test, a module called 'finaljson' is activated. This module extracts data from both 'ww.json' and 'test.json' and combines them into a single JSON file called 'final.json.' The 'final.json' file provides a comprehensive summary of the security test outcomes, including information about any identified vulnerabilities.

The Python toolkit proposed in this project includes a second module that facilitates the analysis of firmware files and the extraction of the root file system. Upon initiation, the module prompts the user to provide the path to a .bin file. Once the path is specified, the module automatically extracts the root file system and scans it to identify significant details about the firmware. Subsequently, the toolkit generates a comprehensive report summarizing the firmware analysis results. The report provides crucial information about the firmware, such as its architecture, version, and components included in the root file system. Additionally, the user can choose the preferred format of the final report.

V. RESULT AND DISCUSSION

For the analysis of this toolkit we used Extreme Vulnerable Web Application (XVWA) which is a web application intentionally designed with multiple security vulnerabilities for the purpose of training and practicing web application security testing. It is an open application and provides a sage and legal environment for security researchers, pentesters and developers to learn about web application vulnerabilities.

Attacks	Harbinger	Skipfish	Arachni
SQLi	Yes	Yes	Yes
BSQLi	Yes	Yes	Yes
RXSS	Yes	Yes	Yes
LFI	Yes	Yes	No
LOG4SHELL	Yes	No	No
CSRF	Yes	Yes	Yes
NOSQLi	Yes	No	Yes

Table 1. Number of Vulnerabilities of the evaluated scanners

False negative rates and false positive rates are two crucial ideas in the world of information security and are frequently applied to gauge the effectiveness of security products like vulnerability scanners, intrusion detection systems, and antivirus programmes.

False Negative Rate: The percentage of real positive occurrences that a security instrument fails to detect is measured by the false negative rate. In other words, it is the frequency with which genuine security incidents are missed by the security instrument. Data breaches or other malicious behaviour may occur as a result of major security incidents that go unreported due to a high false negative rate.

False Positive Rate: The false positive rate calculates the proportion of erroneous alarms that a security instrument generates. In other words, it refers to the frequency with which occurrences that aren't true security concerns result in warnings or reports being generated by the security instrument.

A high false positive rate can lead to alert fatigue, which can make it difficult for security teams to identify and respond to genuine security incidents.

The formulas for false negative rate (FNR) and false positive rate (FPR) are as follows:

- *False Positive Rate (FPR) = False Positives/(False Positives+True Negatives)*
- *False Negative Rate (FNR) = False Negatives/(False Negatives+True Positives)*

Where:

- False Negatives: how many incidents pose real security risks but go unnoticed by the security tool
- True Positives: the number of incidents that the security tool correctly identifies as security threats
- False Positives: the count of events that are not actual security threats, but are incorrectly detected by the security tool
- True Negatives: the count of events that are not actual security threats and are correctly identified as such by the security tool.

Name	False Negative	FPR	Detail Description
Harbinger	6/15	10%	Yes
Arachni	5/13	50%	Yes
Skipfish	5/10	29%	No

Table 2. Summary of the result

Tool Name	Capabilities
Harbinger	File Extraction, can search for sensitive information and network keys, supports signature scanning, identify known file types and signatures
Bytesweep	Identify and extract common file types and structures, uses ML.
FREED	Includes prebuild analysis plugin, offer web interface

Figure 3 Firmware Analysis Tool Comparison

VI. FUTURE SCOPE

The future scope of this toolkit can be:

1) Improved Reporting and Visualization
2) Integration with Other Security Tools and Platforms
3) Support for Different File Formats
4) Integration with Cloud-Based Services
5) Support for Different Operating Systems

VII. CONCLUSION

In this paper, we successfully implement a toolkit for Web Clients and IOT devices vulnerability detection. This toolkit enables us to conduct attacks like MITM and XSS scripting. To investigate the devices' weaknesses, we can potentially execute a variety of different assaults. Once these flaws are identified, we may produce a report and offer fixes for these security issues.

Our proposed model (Harbinger) has lower false positives than the current commercially solutions like Arachni, Skipfish models.

REFERENCES

[1] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum and Nasir Ghani, “Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-scale IoT Exploitations,” in 1553-877X (c) 2018 IEEE

[2] Octavia Georgiana Dorobantu, Simona Halunga, “Security threats in IoT,” in 2020 IEEE

[3] Paul Loh Ruen Chze, Kan Siew Leong, “A Secure Multi-Hop Routing for IoT Communication,” in 2014 IEEE World Forum on Internet of Things (WF-IoT)

[4] Tianma Wang, Jianjun Qiand and Dongmei Zhao, “Research on Cross-site Scripting Vulnerability of XSS Based on International Student Website” in 2022 International Conference on Computer Network, Electronic and Automation (ICCNEA)

[5] Huyam AL-Amro and Eyas El-Qawasmeh, “Discovering security vulnerabilities and leaks in ASP.NET websites”, in Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)

[6] Danny Alvarez E., Daniel Correa B. and Fernando Arango I., “An Analysis of XSS, CSRF and SQL Injection In Colombian Software And Website Development”, in 2016 8th Euro American Conference on Telematics and Information Systems (EATIS)

[7] Himanshi Singh and Mohit Dua, “Detection & prevention of website vulnerabilities: Current Scenario and Future Trends”, in 2017 2nd International Conference on Communication and Electronics Systems (ICCES)

[8] Wei Xie, Yikun Jiang, Yong Tang, Yuanming Gao, Ning Ding, "Vulnerability Detection in IoT Firmware: A Survey", in 2017 IEEE 23rd International Conference on Parallel and Distributed Systems

[9] Bing Wang, Lu Liu, Feng Li, Janye Zhang, Tao Chen, Zhenwan Zou, "Research on Web Application Security Vulnerability Scanning Technology", in 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC 2019

[10] Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti, Eurecom, "A Large-Scale Analysis of the Security of Embedded Firmwares", in 23rd USENIX Security Symposium. August 20–22, 2014 • San Diego, CA. ISBN 978-1-931971-15-7

[11] Andrei Costin, "Lua code: security overview and practical approaches to static analysis", in 2017 IEEE Symposium on Security and Privacy Workshops

[12] Roland Zegers, "HTTP Header Analysis", in Master System and Network Engineering University of Amsterdam August 2015