STM JOURNALS

Journal of
Open Source Developments

ISSN: 2395-6704
Volume 11, Issue 3, 2024
September–December
DOI (Journal): 10.37591/JoOSD

https://journals.stmjournals.com/joosd

Research JoOSD

Securix: An Open-Source Security-based Linux Distribution for Novice Cyber Security Enthusiasts

Riona Maria Varghese[1], Neeraja Anil[2,*], Nidhin Anil[3], Nikhil John Jose[4], Arun Madhu[5], Justin Mathew[6]

Abstract

In the ever-evolving field of cybersecurity, the need for accessible and user-friendly tools in favor of beginner users has never been greater. The existing ethical hacking-related operating systems, while powerful, often overwhelm beginners with their complexity. The main objective of this project is to bridge the gap by providing an operating system for novice cyber enthusiasts looking to explore the world of ethical hacking and security. The proposed operating system will provide informative and user-friendly applications that simulate different attacks such as XSS attacks, SQL injection, Wi-Fi attacks, and Distributed Denial-of-Service (DDoS) attacks, thereby allowing the users to gain hands-on experience of the same. The app will be developed using the PyQt6 framework for the frontend, with Firebase serving as the backend. The operating system is created using Linux From Scratch to provide a minimal system with only the needed tools. The applications provide a brief tutorial on what each attack is, what type of weakness it depends on, and what exploits are possible. It also introduces the user to a few common sets of tools that are used for exploiting these attacks. As a unique learning platform, our operating system focuses on simplicity and aims to be a stepping stone for absolute beginners to be able to solidify their foundations and navigate through the field of cybersecurity with ease and confidence.

Keywords: Cybersecurity, education, ethical hacking, operating system, Linux

INTRODUCTION

Cybersecurity education faces ongoing challenges in providing accessible and comprehensive learning tools for beginners, particularly for navigating the complexities of various hacking Linux distributions. Although these distributions are indispensable for experienced users, they often lack user-friendly interfaces and tailored guidance for novices. This gap has spurred the development of a meticulously crafted Linux operating system expressly designed for novice cyber enthusiasts seeking hands-on learning experience. The existing array of hacking distributions frequently falls short of addressing the distinct requirements of beginners in cybersecurity. These distributions, which lack an intuitive graphical user interface (GUI), inadvertently impose a steep learning curve, thereby rendering navigation and interaction daunting for users who are unacquainted with the intricacies of cybersecurity. Furthermore, the absence of clearly defined pathways for inclusive learning often overwhelms novices and impedes their progression in the field. At its core, our operating system is carefully constructed with a user-friendly GUI for applications, ensuring accessibility for individuals across varying levels of technical expertise. Diverging from conventional distributions, Securix places robust emphasis on inclusiveness, providing

***Author For Correspondence**
Neeraja Anil
E-mail: neerajaanil020@gmail.com

[1-4]Student, Department of Computer Science and Engineering, Saintgits College of Engineering, Pathamuttom, Kerala, India
[5,6]Associate Professor, Department of Computer Science and Engineering, Saintgits College of Engineering, Pathamuttom, Kerala, India

Received Date: June 03, 2024
Accepted Date: October 15, 2024
Published Date: November 04, 2024

Citation: Riona Maria Varghese, Neeraja Anil, Nidhin Anil, Nikhil John Jose, Arun Madhu, Justin Mathew. Securix: An Open-Source Security-based Linux Distribution for Novice Cyber Security Enthusiasts. Journal of Open Source Developments. 2024; 11(3): 1–6p.

dedicated applications that simulate a diverse range of cyberattacks. This unique approach empowers novice users to grasp the fundamentals of cybersecurity in a controlled environment, fostering a supportive and immersive learning experience. Securix offers a multitude of advantages that set it apart in cybersecurity education. The user-friendly GUI not only demystifies complex concepts but also accelerates the learning process, enabling individuals with limited technical backgrounds to navigate and comprehend intricate cybersecurity principles. The explicit focus on ethical hacking within Securix stands as a testament to our commitment to promoting the responsible and lawful exploration of cybersecurity techniques. Moreover, an inclusive learning environment ensures that users from all backgrounds can confidently develop their skills, contributing to the creation of vibrant cybersecurity professionals. According to Cisco's 2018 Annual Security Report, every web application surveyed has at least one vulnerability, which reflects the growing threat of web attacks. Cross-site scripting (XSS) was identified as the most common attack method, accounting for 40 attacks, and the cause of these vulnerabilities is that developers do not know how to implement server-side protection. This area of expertise makes it difficult for many organizations to implement comprehensive security measures. Consequently, developers of popular browsers such as Firefox, Chrome, and Internet Explorer have stepped in to create client-side filters designed to detect and mitigate XSS attacks before they become serious. Despite these efforts, XSS remains a significant threat owing to its ability to exploit multiple entry points in web applications [1, 2]. XSS attacks can be classified into three basic types: persistent, non-persistent, and XSS document object (XSS) attacks. Persistence XSS attacks occur when malicious documents are stored on the target server, such as databases, message boards, guest blogs, or discussion sites. Non-persistent XSS attacks, also known as reflections, are a part of the application. Document object model (DOM)-based XSS attacks use vulnerabilities in client code instead of server code to change the DOM environment of the victim's browser to perform the attack.

Another major web application vulnerability is SQL injection attacks (SQLIA), which are considered as one of the top three threats. SQLIA allows an attacker to compromise queries that an application executes against a database, potentially accessing unauthorized data including sensitive information about other users, logins, and the database itself. This type of attack can be dangerous because an attacker can read, modify, or delete data from a database, compromising the integrity and confidentiality of the stored information. An SQLIA search involves analyzing network data, a process that requires examining all traffic passing through the network, which can be complex and resource intensive [3]. Kali Linux is a powerful Linux distribution that is strong and focused on penetration testing, ethical hacking, and cybersecurity. Kali Linux, created by Offensive Security, has a large number of preinstalled tools that help with different information security assessment stages. Owing to its extensive toolkit, which includes tools for digital forensics, network analysis, vulnerability assessment, and password cracking, it is a priceless tool for both cybersecurity enthusiasts and professionals. Kali Linux is well known for its dedication to privacy and security and offers a safe environment for evaluating and enhancing computer system resilience [4].

METHODOLOGY

Operating System Development using Linux From Scratch

Linux From Scratch (LFS) is a project that provides instructions for building a Linux system entirely from source code. This involves manually constructing the essential components of Linux distribution, including the kernel, libraries, and utilities. The process is educational and allows customization based on specific needs. Building a bare Linux operating system from LFS involves several detailed steps to ensure a fully functional system. The process begins with preparation, which includes setting up the host system by verifying minimum hardware and software requirements, installing essential development tools such as GNU compiler collection (GCC), Make, and Binutils, ensuring that necessary libraries and dependencies are installed, and confirming that the user has the required permissions to create directories and install software. Additionally, creating a dedicated partition involves partitioning the hard drive, choosing a file system format, such as ext4, mounting the partition to a designated directory, and setting appropriate filesystem permissions. Next is Toolchain Installation,

in which a cross-toolchain is compiled. This includes understanding cross-compilation, where tools are built to generate code for the LFS system, and compiling components, such as GCC, Binutils, and Glibc, to create a cross-compiler toolchain. This toolchain runs on the host system but targets an LFS system. A proper configuration ensures compatibility with the target architecture. The temporary system step involves building a minimal self-contained system on the host using a cross-compiler and essential tools. Critical tools such as Coreutils, Binutils, and shell utilities were compiled and installed to form the initial building blocks for the LFS system. The minimal libraries necessary for the tools to function were also installed. Transitioning to the temporary system requires configuring environmental variables to prioritize the newly built tools and switching the default shell to maintain consistency. In the Final System phase, additional libraries essential for the proper functioning of the LFS system, such as Glibc, were compiled and installed. Configuration files are edited to reflect the desired setup, including kernel settings and system files, such as /etc/fstab and /etc/passwd. For the Linux Kernel, source code was downloaded and extracted. Kernel configuration involves customizing options using tools like 'make menuconfig,' selecting device drivers, filesystems, and networking options based on hardware requirements. The kernel is then compiled using the 'make' command, generating the kernel image (vmlinuz) and associated modules, which are installed in the boot directory. Booting involves configuring a bootloader, such as grand unified bootloader (GRUB) or line in line out (LILO) to the Master Boot Record (MBR), allowing the system to load the kernel during boot. The bootloader was configured to recognize the kernel image location and parameters. Creating an initial RAM disk (Initramfs) is essential for systems requiring early driver loading. System Configuration includes creating user accounts using tools like 'useradd' and 'passwd,' setting secure passwords, and defining user groups. Critical system configuration files such as network settings are edited to manage various aspects of the system. Installing System Software involves downloading additional software packages, compiling them using LFS system tools and libraries, and installing them with 'make install.' Customization of software packages, including adjusting configuration options, ensures smooth integration with the LFS system. Finalizing an LFS requires configuring network settings, editing network configuration files, assigning a hostname, and conducting system checks. This includes verifying user accounts, testing network connectivity, and ensuring that the kernel modules are loaded. Finally, Beyond LFS, further customizations and package management were explored. Installing a package manager simplifies software management, allowing easy installation, upgrading, and removal of packages. Customizations can include kernel tweaks, configuration of the user environment, and installation of additional tools or software based on user preferences, as shown in Figures 1 and 2. Each of these steps is crucial for building a robust and functional Linux operating system, ensuring that every component is tailored to the specific requirements of the user and hardware [5].

Application Development and Integration

Building applications using the Qt framework in Python with the PyQt6 library for a Debian-based OS involves a detailed and structured approach that aims to educate users about cyberattacks such as XSS, SQL Injection, DoS, and Wi-Fi attacks. The process begins with defining clear educational objectives. This includes specifying the targeted cyberattacks and outlining the desired learning outcomes, such as understanding attack methodologies, understanding the tools used, and implementing preventive measures. The next step was to design a user-friendly interface. This involves careful consideration of the layout to ensure clarity and ease of navigation and incorporating graphics and interactive elements to effectively convey simulated cyberattacks. Tools such as Figma are used to create a responsive, basic frame layout, ensuring that the design is adaptable across various devices. The codebase layout was then established using the Qt and PyQt6 libraries. This foundational structure supports cross-platform development, ensuring that the application runs smoothly on multiple operating systems. The base code is written to establish the overall layout, navigation, and essential components of the user interface, serving as the backbone for further functionalities [6, 7]. A well-structured SQLite database is used to store the tutorial and level content. SQLite was chosen for its simplicity and efficiency, making it ideal for embedded applications. The database was seamlessly integrated with the application, allowing for dynamic data interactions. The application structure included several key

sections. The Welcome Page serves as the initial landing page, providing a warm introduction and navigation to the Index Page. The Index Page acts as the central hub, facilitating easy access to all other sections. The Tutorial Section offers comprehensive and beginner-friendly content on various types of cyberattacks, guiding users through step-by-step instructions, and real-world case studies. The Levels Section provides an interactive learning experience through quizzes, reinforcing the knowledge gained in tutorials. The Help Section offers a comprehensive walkthrough of the application, ensuring that users can navigate and use it effectively. The About Section provides information on the creation and objectives of the application, including project details, mission statements, and contact information. Figure 3. Welcome Page: The content of the application is finalized, ensuring that there are no informative errors, and that navigation is flawless. Security measures were implemented to protect the content. The application underwent thorough testing, focusing on different modules, such as loading time, data retrieval, and updates. Feedback was gathered from users to identify areas for improvement. Deployment involves creating comprehensive documentation outlining the deployment process, including the system requirements and step-by-step instructions. The deployment environment is checked to ensure that it meets all specified requirements. Effective error handling was implemented to capture and log errors for subsequent analyses. Post-deployment testing was conducted to identify any remaining issues, and the documentation was updated accordingly. Building these educational applications involves a systematic and multifaceted approach, starting with defining educational objectives and designing a user-friendly interface. The code base was laid out using the Qt framework and PyQt6 library, and a well-structured SQLite database was integrated [8–10]. The application structure was carefully designed to include various sections, each with a specific purpose. Content is finalized and secured, followed by thorough testing and feedback collection. Finally, the application is deployed with comprehensive documentation and post-deployment testing to ensure seamless user experience and effective institutional implementation, as shown in Figure 4.

Figure 1. GRUB menu.

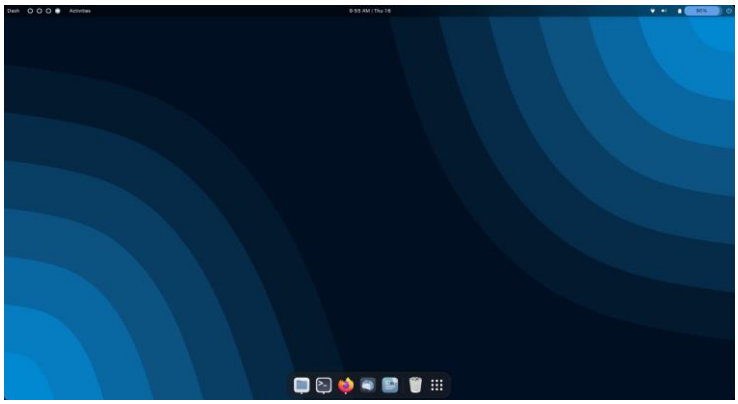

Figure 2. Securix desktop.

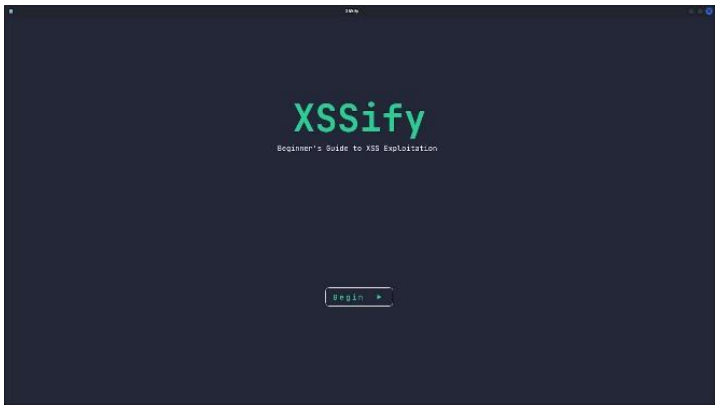

Figure 3. XSSify welcome page.

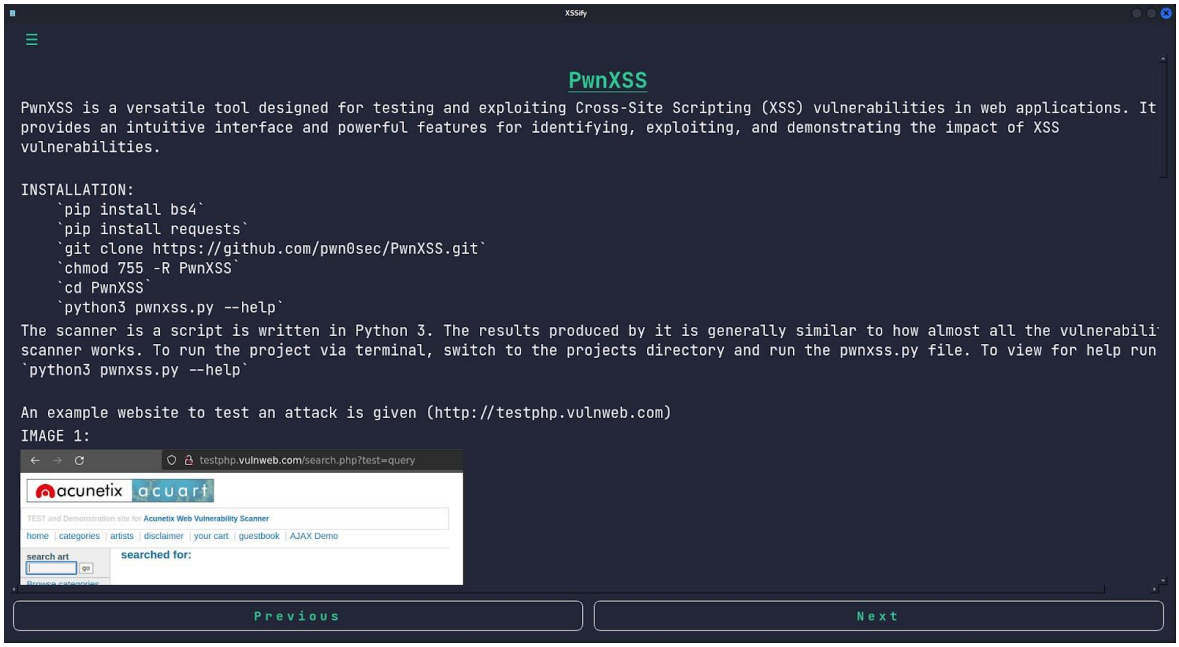

Figure 4. Tutorial content.

CONCLUSIONS

Securix stands as a beacon of innovation in cybersecurity education, poised to redefine learning experiences for enthusiasts entering this dynamic field. Securix unlocks a new age in cybersecurity education by promoting ethical values, accessibility, and simplicity. This initiative was driven by recognition of the rising demand for easily accessible cybersecurity education. With its user-centered methodology, inclusive design, and useful applications, Securix hopes to empower a wide range of students and foster a community dedicated to ethical cybersecurity practices. Securix is committed to increasing access and promoting ethical consciousness. Its mission is to provide a thriving ecosystem in which people from all backgrounds can pursue cybersecurity professionalism. In conclusion, Securix embodies transforming power, a precursor to a future where cybersecurity education is not just comprehensive, but also ethical and accessible to all.

Acknowledgments

We express our sincere gratitude to all those who contributed to the successful completion of this project. Our deepest appreciation goes to our coordinators, Dr. Anju Pratap and Dr. Jo Cheriyan, for their invaluable guidance, support, and encouragement throughout this journey. We are also thankful to our colleagues and friends for their support and constructive feedback. Special thanks to the Department of Computer Science and Engineering, Saintgits College of Engineering, for providing the necessary

resources and environment conducive to research and development. Finally, we would like to extend our gratitude to our families, friends, and wishers for their unwavering support and understanding.

REFERENCES

1. Shrivastava P, Jamal MS, Kataoka K. EvilScout: Detection and mitigation of evil twin attack in SDN enabled WiFi. IEEE Trans Netw Serv Manag. 2020;17:89–102. DOI: 10.1109/TNSM.2020.2972774.
2. Rodríguez GE, Torres JG, Flores P, Benavides DE. Cross-site scripting (XSS) attacks and mitigation: A survey. Comput Netw. 2020;166:106960. DOI: 10.1016/j.comnet.2019.106960.
3. Crespo-Martínez IS, Campazas-Vega A, Guerrero-Higueras ÁM, Riego-DelCastillo V, Álvarez-Aparicio C, Fernández-Llamas C. SQL injection attack detection in network flow data. Comput Secur. 2023;127:103093. DOI: 10.1016/j.cose.2023.103093.
4. Balarezo JF, Wang S, Chavez KG, Al-Hourani A, Kandeepan S. A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks. Eng Sci Technol Int J. 2022;31:101065. DOI: 10.1016/j.jestch.2021.09.011.
5. Kali Linux. (2024). Should I Use Kali Linux? Kali Linux Documentation. [online] Kali Linux. Available from: https://www.kali.org/docs/introduction/should-i-use-kali-linux/
6. Riverbank Computing. (2024). What is PyQt? Introduction. [online] Available from: https://riverbankcomputing.com/software/pyqt/intro
7. Gupta S, Gupta BB. Cross-Site Scripting (XSS) attacks and defense mechanisms: Classification and state-of-the-art. Int J Syst Assur Eng Manag. 2017;8:512–530. DOI: 10.1007/s13198-015-0376-0.
8. Halbouni A, Ong LY, Leow MC. Wireless security protocols WPA3: A systematic literature review. IEEE Access. 2023;11:112438–112450. DOI: 10.1109/ACCESS.2023.3322931.
9. Anonymous. (2014). BlackArch Linux - Penetration Testing Distribution. [online] Hacking Reviews. Available from: https://www.hacking.reviews/2017/09/blackarch-linux-penetration-testing.html
10. Linux From Scratch (LFS). (2024). Welcome to Linux From Scratch! [online] Available from: https://www.linuxfromscratch.org/

