ProctorBox: The Open-Source Operative System Designed to Guarantee the Integrity & Security of Online Assessments

Roberto D. Solis, Sam Houston State University, United States
Narasimha Shashidhar, Sam Houston State University, United States
Cihan Varol, Sam Houston State University, United States
Amar Rasheed, Sam Houston State University, United States

The IAFOR International Conference on Education in Hawaii 2024
Official Conference Proceedings

Abstract
Computerized proctoring systems play an essential role in protecting the integrity and security of assessments in several modes of instruction, such as online learning. The most recent literature focuses on adding features that may impede the student from committing academic dishonesty and fraud by combining several elements that may protect the integrity of the remote assessment. The majority of the features we can explore in contemporary literature are the detection of mobile devices, eye movement tracking, extracting verbal communication to text, and other complex solutions involving pre-trained models to improve the efficiency of detecting objects. However, there needs to be an initiative to solve the issue of academic dishonesty in proctoring systems. Protecting the integrity and security of online assessments is possible by providing the student with a robust combination of security measures embedded into an operative system. Therefore, we present ProctorBox, the open-source platform explicitly engineered to protect the integrity and security of an online assessment. We implement state-of-the-art security by hardening the system with essential concepts that work as a synergetic mechanism to impede fraud and academic dishonesty. First, we obtain the image of an open-source system to eliminate licensing issues and the cost of the computerized proctoring system. Then, we establish a baseline image of the system by removing all the unnecessary components that may aid in committing dishonesty. Moreover, we secure the guest account from executing third-party applications and handle internet browser security by building a Chromium extension to safeguard the integrity of the online assessment.

Keywords: Proctoring Systems, Online Assessments, Assessment Integrity, Distance Education

Introduction

The current proctoring solutions available in the market and the solutions presented by academics in the most recent literature focus on features that can prevent or make a student doubt before committing academic dishonesty or cheating in an online examination. Face detection, eye movement, audio analysis, extraction of voice into text, and detection of external devices are the most common features in modern proctoring applications. However, as proved by Solis et al. (2023), A proctoring solution is limited to its built-in functionalities, and it is possible to bypass such security features with little to no effort on behalf of the student. Therefore, the security of online assessments goes beyond providing a third-party application to protect assessment integrity. To ensure the safety and integrity of an online examination, give the test taker a secure platform capable of halting any attempts to commit academic dishonesty and fraud on an online assessment.

Therefore, ProctorBox follows rigorous principles such as system hardening, the least privilege, and state-of-the-art browser security to prevent malicious activity while an online exam is in session. ProctorBox is aimed to limit the functionalities and access to external resources that a student can have during an examination, thus, reducing the opportunities to cheat in an online test. ProctorBox is a groundbreaking open-source operating system designed explicitly to assure any faculty member of the integrity of any online examination. The open-source nature of our platform gives it a significant edge, promoting transparency and abiding by the most rigorous academic standards. Nonetheless, our robust platform allows the system owner to add or remove any necessary applications that an assessment may require. The portability and user-friendly interface let students take an assessment effortlessly, making the examination process as seamless and secure as possible.

To showcase the security, integrity, and effectiveness of ProctorBox, we follow rigorous security standards to harden our portable operating system. Our baseline distribution contains no unnecessary applications, making it impossible for the student to commit dishonesty. Moreover, the system can handle user switching, allowing faculty members to add necessary features to the guest account. These features can be accessed by third-party applications provided by our Debian-based system. Moreover, we offer a trailblazing browser security mechanism by implementing a custom browser extension with the ability to block the left click on any window, and that allows only one active tab at a time. In addition, we are pioneers in the academic industry in offering students a complete proctoring platform at no cost. ProctorBox is proven to work on the leading hardware architectures, such as AMD and Intel processors, making it superior to any proprietary proctoring solution available.

Literature Review

Saurav et al. (2021) present a proctoring system that leverages AI features to detect cheating in online examinations. The method comprises seven elements working synergistically by collecting low-level information from each feature. Then, the data is analyzed on a secondary window to identify traits that may lead to cheating. The first component is user verification and facial landmarks detection. The second feature is composed of eye-tracking components that are based on coordinate points. The third feature detects whether the test taker opens their mouth while an examination progresses. The fourth component is a trained model capable of detecting other devices in use during the examination. The fifth feature, head pose detection, identifies whether the test taker is looking at a secondary device. The sixth component, tab switching, is built into Java script so the student does not require additional software. The last component is voice detection, which monitors and extracts audio, which is then converted to text using Google Speech Recognition.

Although we can see significant use of AI technology in exam proctoring, the authors must realize how feasible this proposed model can be at a larger scale. Moreover, we need to find out where the exam is hosted. For example, it would be ideal to know if the proposed system can support modern learning management systems such as Blackboard, Moodle, or Canvas. On the other hand, it would be more beneficial to see a chart to examine the efficiency of such a model.

Ganidisastra and Bandung (2021) propose using a fully automated proctoring system using face recognition to solve the need for continuous user verification while the student is taking an examination. The model consists of grabbing images that are captured at the same time when the lecture is taking place. The incremental training consists of two deep learning models: multi-task cascaded convoluted network and YOLO-face. The two proposed methods demonstrate a higher performance under different conditions, such as low light and alternative poses. The process starts by capturing the student's identity for registration. An image of the student is captured using a mobile device. The system requires several angles of the student, such as front view, left view, right view, and so on. The incremental training occurs at the end of each lecture session once a new data set is collected. The results show that the two proposed models, multi-task cascaded convoluted network and YOLO-face, perform better over viola-jones and LBP, which are also utilized for face detection model processing images as a batch.

While the proposed model requires incremental training for improvement and better performance, it will fail under certain circumstances. For example, lectures that are taken place in large auditoriums will most likely cause students to switch positions during lecture time. Also, if the model requires continuous snapshots of the student using different angles, this will result in less lecture time and consume considerable time. Nonetheless, it is a reasonable effort in automated proctoring systems that require continuous user verification.

Madhusudan et al. (2022) developed a proctoring system to fulfill the needs and wants of current automated proctoring solutions. The system utilizes several multimedia features such as face recognition and capturing of objects such as mobile devices. The plan aims to eliminate the traditional pen-and-paper examination and remove the human factor from an online exam; in this example, a proctor is nonexistent to reduce the cost of proctoring a test. The proposed system consists of a website with two separate logins. One login pertains to the student, and the second login screen belongs to the faculty member. The system can allow the faculty member to generate exams. After that, a unique key is generated, which is then given to the student along with a password to access the exam. A text editor shows when a mobile device is detected to capture discrepancies during the online examination. Moreover, a warning is also displayed to the student if the test taker attempts to switch tabs. Some of the limitations of the proposed system are the use of proper lighting during registration, which may cause improper face detection. Also, a minimum distance is required between the webcam and the student.

The features implemented in this multimedia proctoring system are good features to consider reducing the cost of proctoring software. However, as a faculty member, there is no clear evidence of a student intending to cheat since all the evidence is based on text. A student can build a case based on these observations and appeal that no intent of cheating occurred during an examination. Furthermore, the faculty members may question the system's performance if several flags happen during an examination session.

Padilla et al. (2021) proposed a cloud-based system to manage remote proctoring of examinations. The method comprises several AWS features such as Dynamo DB, S3, Lambda functions, Cloudfront, and API Gateways. The remote proctoring software shall consist of two applications, one for faculty members and a front end for the students. Moreover, there is an additional machine-learning component capable of detecting head movements during the examination, which are recorded as notifications for playback. An examination was given to thirty students to test the system's performance. Then, the exam was accompanied by a post-assessment survey to measure the system's effectiveness. Of a sample of the thirty students, twenty-one admitted some cheating or plagiarism. However, the system could identify a different group of five students who also committed some plagiarism.

The features presented in this work and the type of technology implemented meet the current demands of proctoring solutions. This is a platform that proved to be efficient and that has excellent performance to detect cheating or plagiarism. However, a critical factor should have been considered, which is the security of the test taker system. The security features in this system, which is meant to run in Windows distributions, can quickly be canceled with a remote desktop connection to the test-taker machine. This will create no alerts for the faculty since the student will have the proper head placement during the examination. However, the faculty will not be aware that a person is taking the test remotely on behalf of the student.

Abozaid and Atia (2022) offer a semi-automated proctoring solution with three main features: head-pose, a feature that locates the head movement of an individual using computer vision technology. The second component, object detection, is used to identify resources not allowed during an examination, such as phones, persons, and books. The third component, eye-gazing, is used to track the position of the eyes during an investigation. The system captures a picture of the student at an interval of every ten seconds. Then, the image is processed, and a report is generated for the faculty member if an abnormality is found. Four different experiments validated the effectiveness of this proposed solution. The first experiment was composed of head pose testing, and the VGG16 model had the highest accuracy. A second experiment was done to test the efficacy of the three features separately. The overall accuracy was 96.66%. The third and last experiment involved students doing different activities to simulate cheating.

Overall, the proposed multi-modal system has exciting features that can minimize the efforts of having a proctor in person and reduce the cost of a proctoring solution. However, the testing was done following a strict set of parameters given to the students. On the other hand, it would be ideal to include detailed screenshots of the report generated once an event is captured by one of the three features. Moreover, it would be great to know how the video is stored and processed on the back end so that faculty can rely on playback if necessary.

Grigoriev (2022) presents a detailed review of the market's most popular proctoring solutions and briefly describes each system's flaws. However, a method is implemented to improve the current proctoring solutions with four distinct functionalities. To detect abnormalities during the examination, the first component uses YOLOv3, a model frequently used in proctoring solutions to detect objects such as mobile devices and more than one person. The second component, which detects abnormalities in the audio, uses FFmpeg. This feature depends on the DB levels captured during the testing session to see if cheating is happening. Moreover, the feed from the audio is converted to text using Google Speech Recognition API, which gives the proctor a list of the most used words. The third component is to detect abnormalities in the test taker's desktop. As discussed in the first component, this feature also utilizes a model based on convolutional neural networks to identify social media platforms, calculators, or additional software not allowed during the examination. The final component is a control to reduce the system's false positives. This final module provides the proctor with a time interval of the video along with a description of the violation committed during the session.

Although the solution is meant to improve the flaws of modern proctoring solutions available in the market, each of the features of the system has been presented by other authors using the same convolutional network models. The only district feature which makes this solution unique is the ability to detect abnormalities based on the DB levels captured during an examination. Furthermore, it would be ideal for reviewing the system's performance on a larger scale and observing if it can simultaneously process multimedia from several sessions.

Malhotra et al. (2022) leverage artificial intelligence to implement a proctoring system that eliminates needing a physical proctor in the room. Three essential components make this system possible. The first component is emotion detection. This feature is possible by using convolutional neural networks. According to their model, emotion detection can predict if the student commits fraud during the examination. The second component of this system is head pose movement. This feature measures 0 to 90 degrees at angles when motion is detected. As a starting point, the student starts at an angle of 0. The model uses six facial points, which are then converted to coordinates. The third and last feature is malicious object detection, which was used in previous work. The model is based on YOLOv3 and can detect objects such as people, mobile devices, and books. This combination of models can aid in identifying if fraud or cheating is committed in an online examination.

Overall, the system uses three essential features that have been in use by several authors. However, the head pose movement feature has yet to be seen as detailed as the one presented in this work. Getting more details on this model's efficacy in a real-world environment would be ideal. There is no additional data to support the efficiency of this model. Moreover, it would be excellent also to include different steps on how it would be possible to scale this solution to support multiple exam sessions at any given time.

Definitions and Tools

Definitions

- Integrity – The state of being whole, entire, or undiminished (Dictionary.com, 2019).
- Least Privilege – The principle that a security architecture should be designed so that each entity is granted the minimum system resources and authorizations needed to perform its functions (Editor, n.d.).
- Proctor – US (in a college or university) a supervisor or monitor who invigilates examinations, enforces discipline, etc. (Definition of Proctor | Dictionary.com, 2019).
- Open-Source – Pertaining to or denoting software whose source code is available free of charge to the public to use, copy, modify, sublicense, or distribute (Open-Source Definition & Meaning, n.d.).
- Operating System – The collection of software that directs a computer's operation controlling and scheduling the execution of other programs, and managing storage, input/output, and communication resources (Definition of Operating System | Dictionary.com, n.d.).
- System Hardening – A process intended to eliminate a means of attack by patching vulnerabilities and turning off nonessential services (Editor, n.d.-a).

Tools

- Chromium Browser – The foundation of our trailblazing browser security relies on altering certain browser functions and behaviors, such as enabling only one tab at a time and turning off the right click. To make this happen, our Slax distribution is optimized to run the latest stable version of the Chromium browser, which is open source. Our custom Chromium extension provides the necessary browser-level security to assure the integrity of any online examination (The Chromium Projects, n.d.).
- LXDE – Desktop environment used to disable Fluxbox, the default built-in application that runs on any Slax distribution (LXDE, n.d.).
- LXDM – The installation of LXDM is necessary to void the auto-login function in Slax. Also, it allows user-switching without any conflicts (LXDM - ArchWiki, n.d.).
- Slax – To provide the test taker with a state-of-the-art secure platform, ProctorBox utilizes the Debian-Based Slax distribution, which is open source, and its design allows the system owner to customize the distribution by turning modules on and off. Slax is stored on a USB drive, and all the system changes are persistent (M, n.d.).
- USB Drive – A USB drive is required to store and boot the ProctorBox system on the student machine.
- AMD or Intel computer system – To test the effectiveness and efficiency of ProctorBox, a baseline image of the system was loaded on both hardware architectures. The system is proven to run on computer systems with AMD and Intel processors.

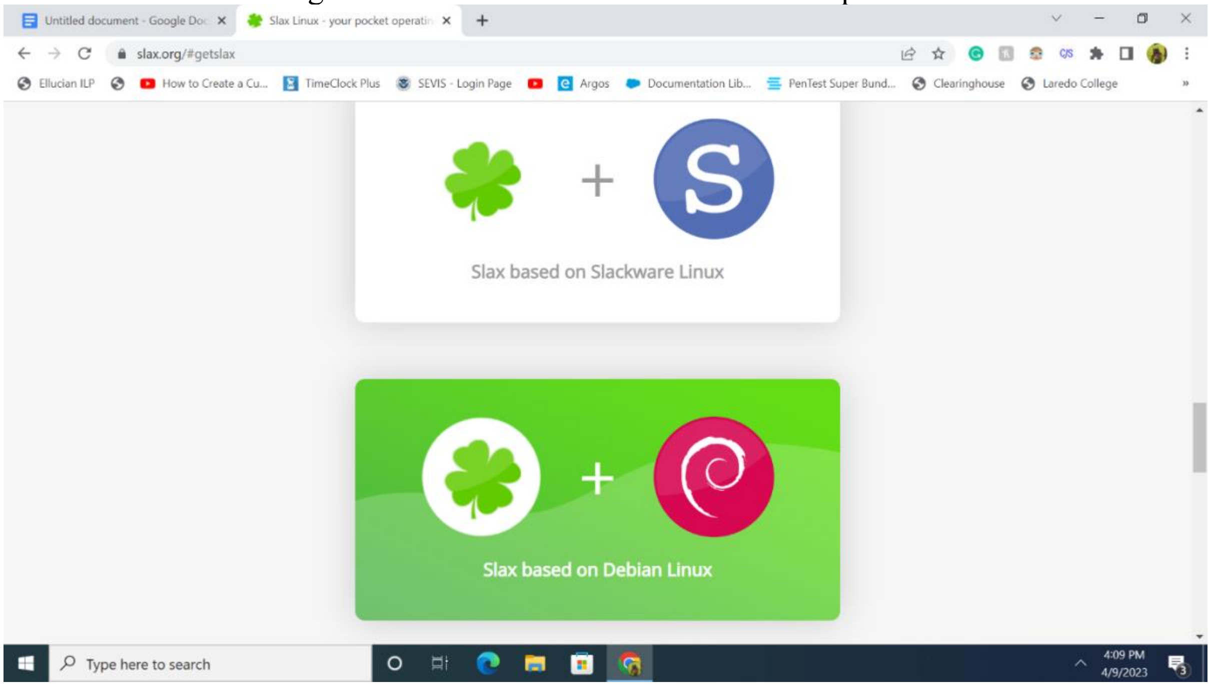
Methodology

ProctorBox aims to achieve maximum security and integrity when it is time to proctor an online assessment. We abide by the most rigorous guidelines to comply with all the requirements to restrict academic dishonesty while maintaining academic integrity. Our leading goal is to make ProctorBox a lightweight and user-friendly platform. We achieve this requirement by obtaining a baseline image of the Slax distribution and applying cyber security concepts such as least privilege, system hardening, and securing the system using only two accounts. The root account is meant for system owners, which in this case is the faculty member. The root account can install new applications and load new modules. On the other hand, the secondary account is the guest account, which has access to a network manager, the Chromium browser, and our custom browser extension. We looked deeply at all baseline software and removed unnecessary tools from the system.

Obtaining a Baseline System Image

The first step was to compare all open-source operating systems that require minimum hardware resources to run on any student machine, preferably any Linux distribution. The idea behind adopting a Linux distribution is to eliminate the proctoring cost to the student and to keep this project open-source for the academic community. After extensive trial and error with several Linux distributions, we adopted Slax for the foundation of our project. Slax is offered in two different architectures, a 32-bit version, and a 64-bit version. Nowadays, all computer systems are based on a 64-bit architecture. Therefore, we opted for a 64-bit architecture based on the Debian distribution.

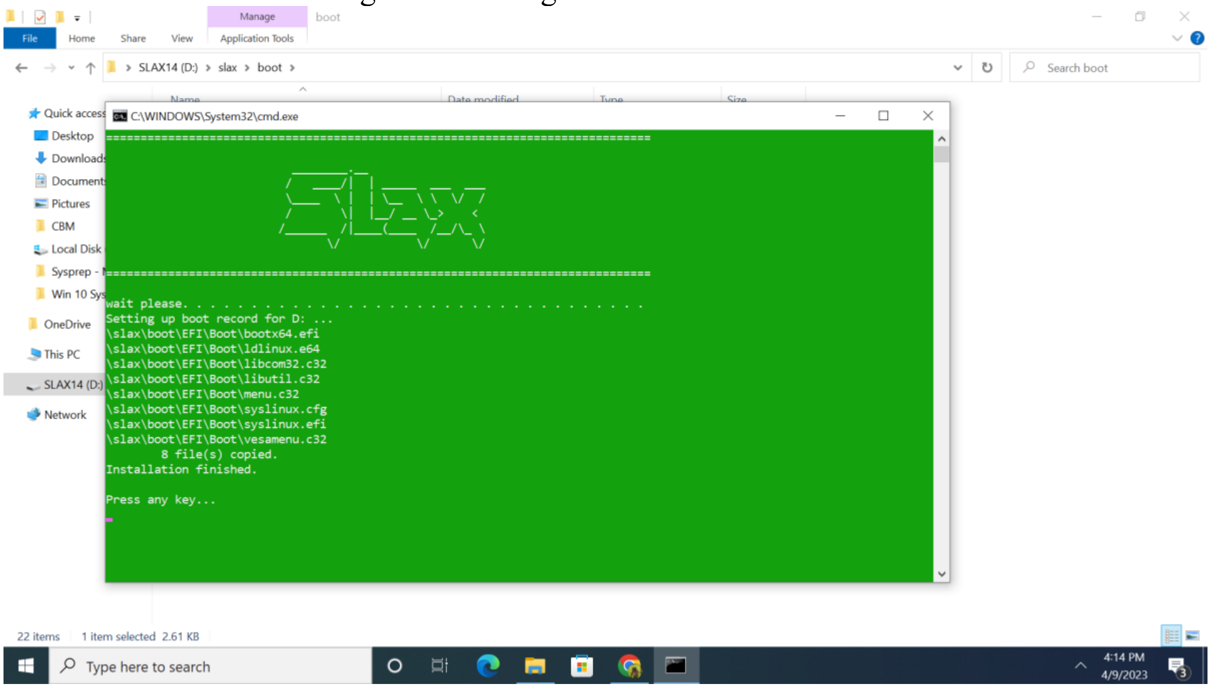
Figure 1: Slax distributions available to the public.


Untitled document - Google Docs
Slax Linux - your pocket operating
slax.org/#getslax
Slax based on Slackware Linux
Slax based on Debian Linux
Type here to search
4:09 PM
4/9/2023

Preparing Our USB Drive for the First Time

Our next step is to prepare the USB drive to store and boot the Slax distribution for the first time. Formatting the target drive using the FAT32 file system is essential. Choosing any other type of file system will make the boot process fail. This concept applies to both Intel and AMD computer systems. Next, we must copy the file's contents obtained in Figure one and paste the complete folder into our target drive. Once the files are moved to our target drive, we must navigate to the Slax folder and double-click the boot directory to access the boot files. Now, we must locate the bootinst file and run it as administrator on our Windows system. Once the process is complete, the green screen will indicate to press any key to exit. At this point, the target USB drive is ready to boot Slax for the first time. Some computers may require additional modifications to the BIOS. Restart the PC and press the appropriate keys to launch the boot menu. From the boot menu, select the USB drive.

Figure 2: Making our bootable USB drive.

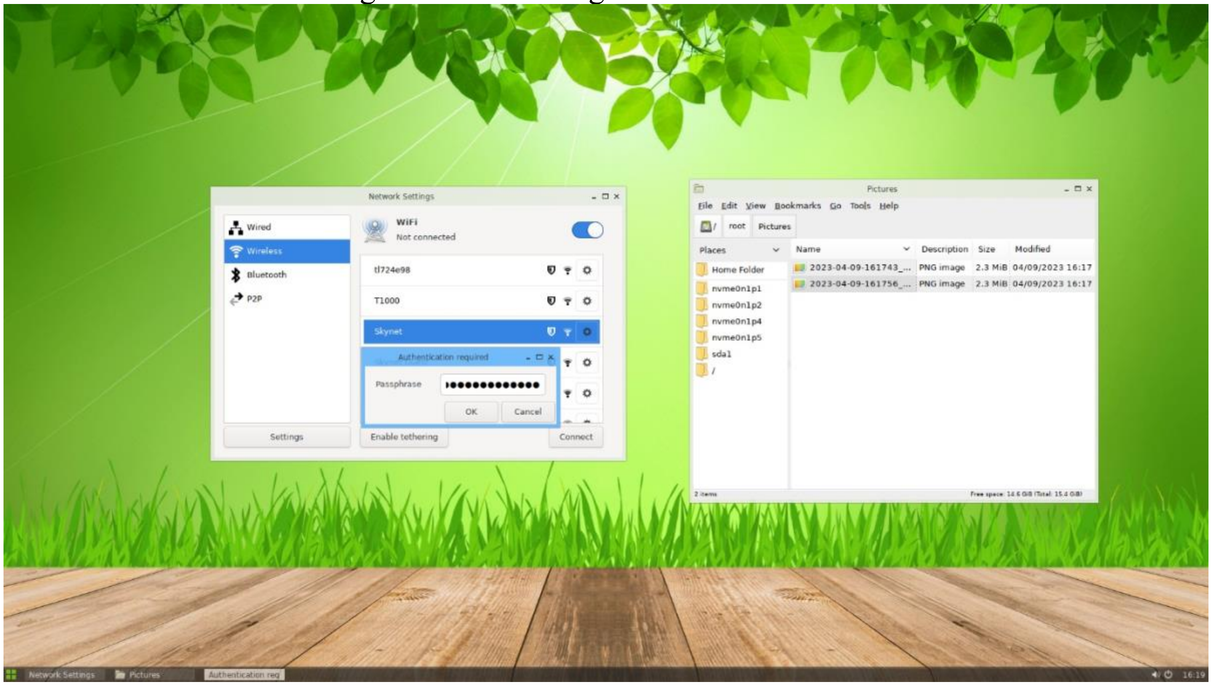


```
================================================================================

================================================================================

wait please. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Setting up boot record for D: ...
\slax\boot\EFI\Boot\bootx64.efi
\slax\boot\EFI\Boot\ldlinux.e64
\slax\boot\EFI\Boot\libcom32.c32
\slax\boot\EFI\Boot\libutil.c32
\slax\boot\EFI\Boot\menu.c32
\slax\boot\EFI\Boot\syslinux.cfg
\slax\boot\EFI\Boot\syslinux.efi
\slax\boot\EFI\Boot\vesamenu.c32
        8 file(s) copied.
Installation finished.

Press any key...
```

Connecting to the Network

Note that after a successful boot for the first time, Slax does not require a username and password and has an automatic login function to the root account. We will modify this process in the following steps. At this point, we need to download the necessary applications for the Slax distribution. From the start menu, launch the Net Manager application, select wireless, choose your home network, and enter the passphrase for your network.

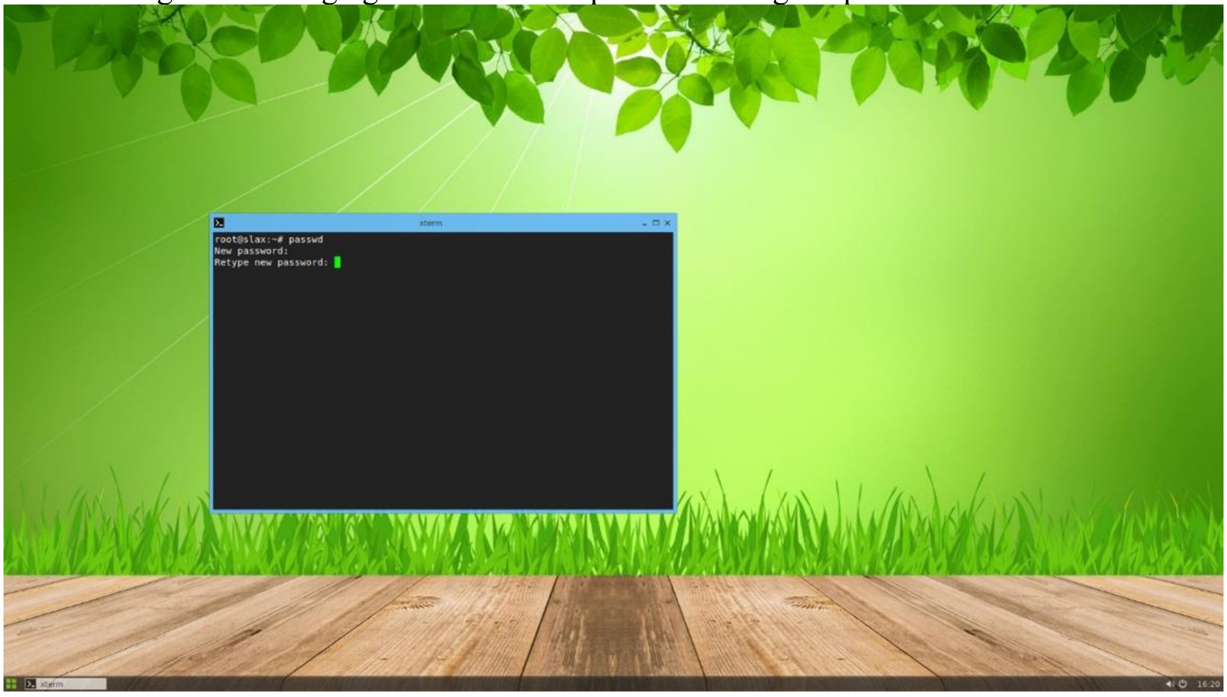
Figure 3: Connecting Slax to the Network.



Changing the Default Root Password

It is time to take ownership of our custom Slax distribution. To make this happen, click on the start menu, and select 'Terminal.' A prompt will open to enter the desired commands. By default, Slax has an automatic login to the root account, and the password is visible during boot time. We want to prevent students from making changes as root. Therefore, enter passwd in the terminal and press the enter key. This action will prompt us to enter the new password to the root account. For testing purposes, the root password is 'shsu.'

Figure 4: Changing the default root password using the passwd command.

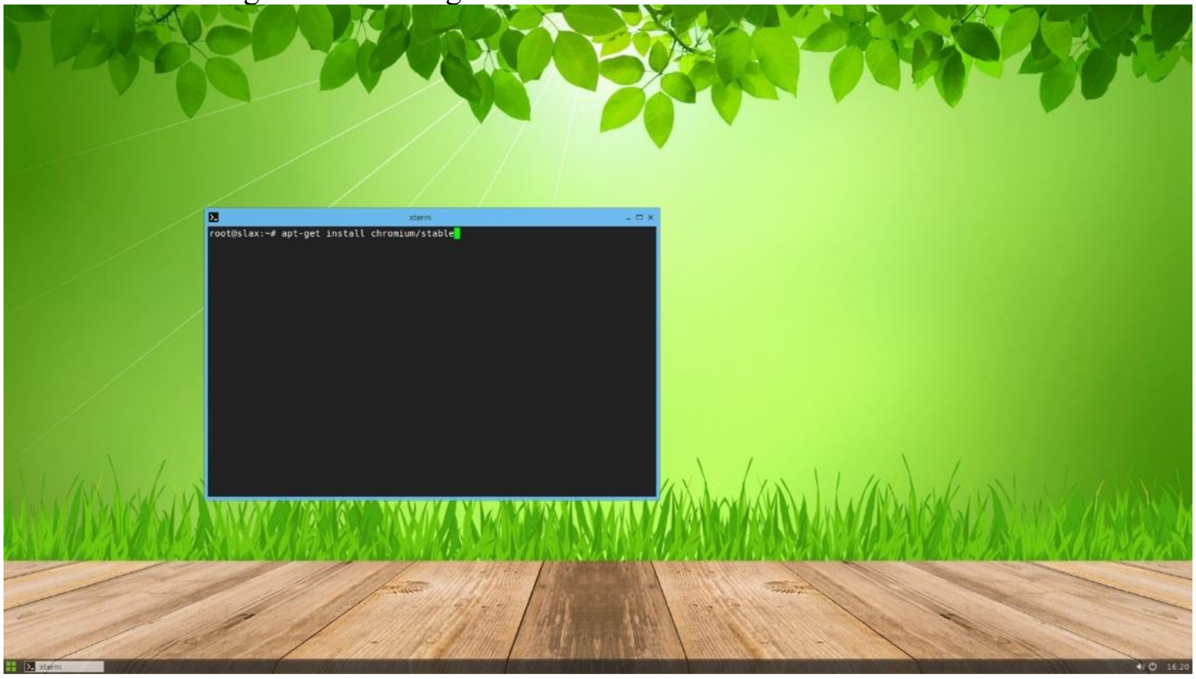


```
root@slax:~# passwd
New password:
Retype new password:
```

Installing Chromium Browser From the Terminal

By default, it is possible to install Chromium using the shortcut under the start menu. However, the installation process may fail due to unexpected versions. It is best to use the command apt-get install chromium/stable to solve this problem. Once the process is complete, we can test that chromium is working. Note that any changes made to chromium as root will transfer to the guest account.

Figure 5: Installing Chromium Browser From the Terminal.

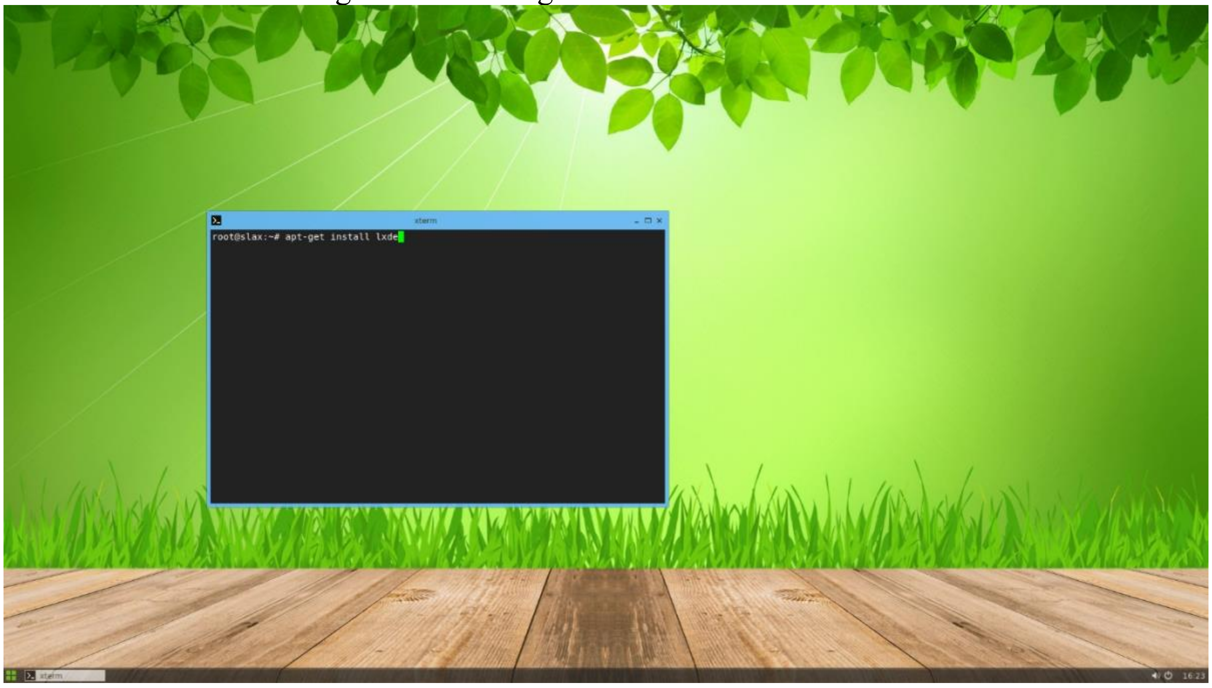


```
root@slax:~# apt-get install chromium/stable
```

Installing a Desktop Environment for the Guest Account

Another default setting in Slax is the desktop environment. As a default setting, Slax uses Fluxbox as the desktop environment. By default, Fluxbox is limited in options and does not allow features such as a login screen, user switching, and a custom desktop environment. Attempting to log in with the guest account, which is built-in by default, will give us a blank desktop with no applications available. Several desktop environments exist, such as GNOME, KDE, & Xfe. To proceed with installing LXDE, launch a terminal window and type the command apt-get install lxde.

Figure 6: Installing LXDE From the Terminal.

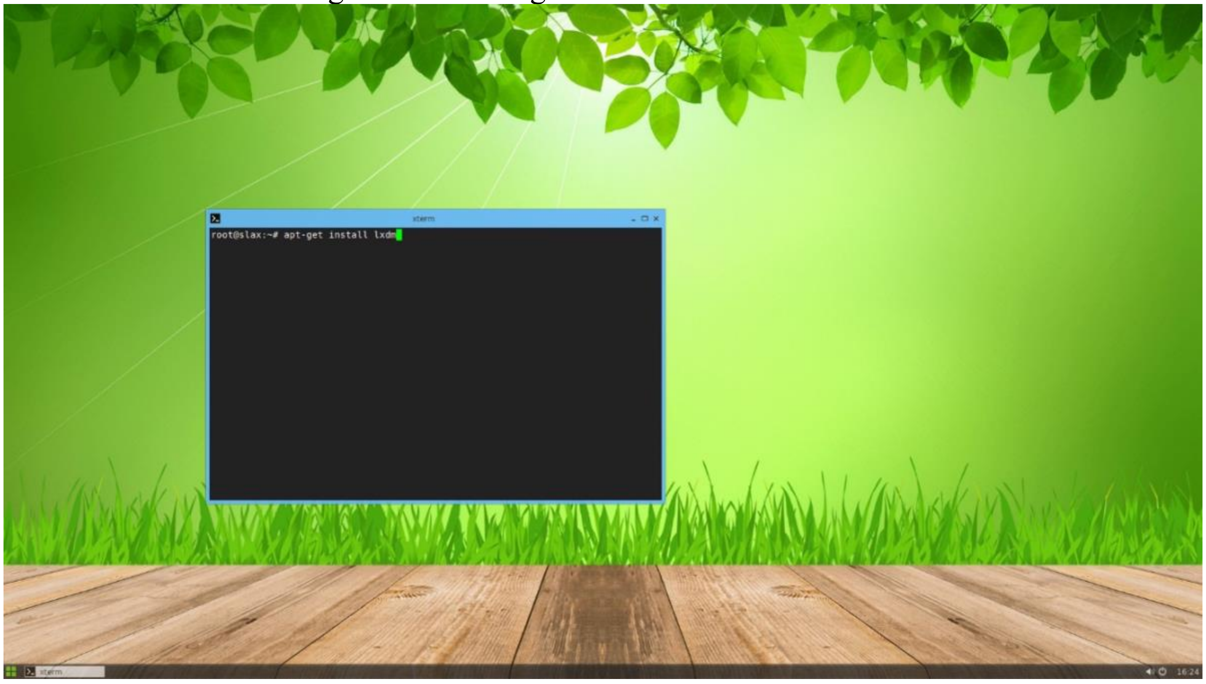


```
root@slax:~# apt-get install lxde
```

Installing a Display Manager for our System

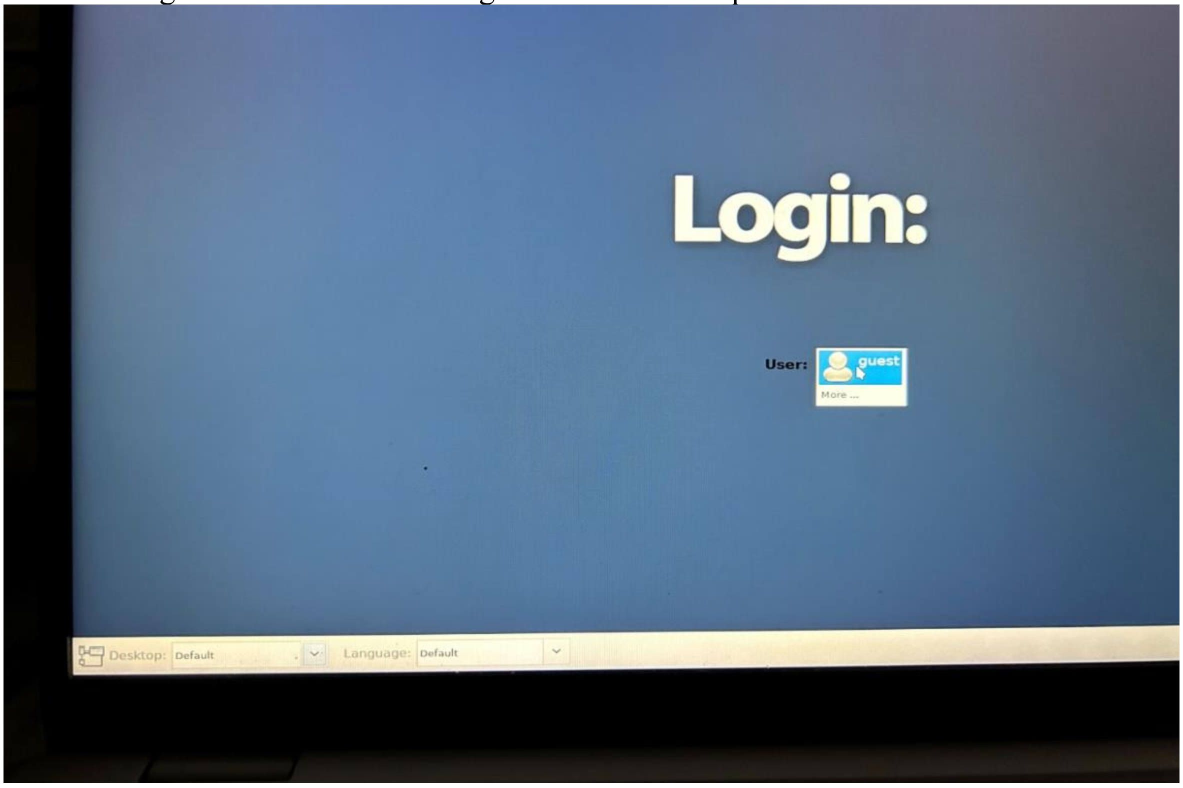
We must incorporate a separate display manager to prevent the auto login feature as root. LXDM can handle user switching without conflicts. Also, this feature manages the users and goes along with LXDE. Moreover, the root account is hidden from the login screen, and the student will only see the Guest account. To install LXDM, launch a terminal window and type the command apt-get install lxdm. We are still logged in as root. Once the installation of LXDM is complete, restart the system for the first time. At this point, the autologin function as root has been turned off by installing LXDM. Now, upon a restart, we are presented with a login screen. By default, the 'Guest' user is shown; we can click 'more' and manually type any desired user. To continue our setup, click 'More' and type 'root' as the user. It is essential to notice how the desktop is set below the user selection box as 'Default.' From now on, the student must log in using the guest account with the default password of 'guest.' And with the desktop setting set to 'Default.'

Figure 7: Installing LXDM From the Terminal.



```
root@slax:~# apt-get install lxdm
```

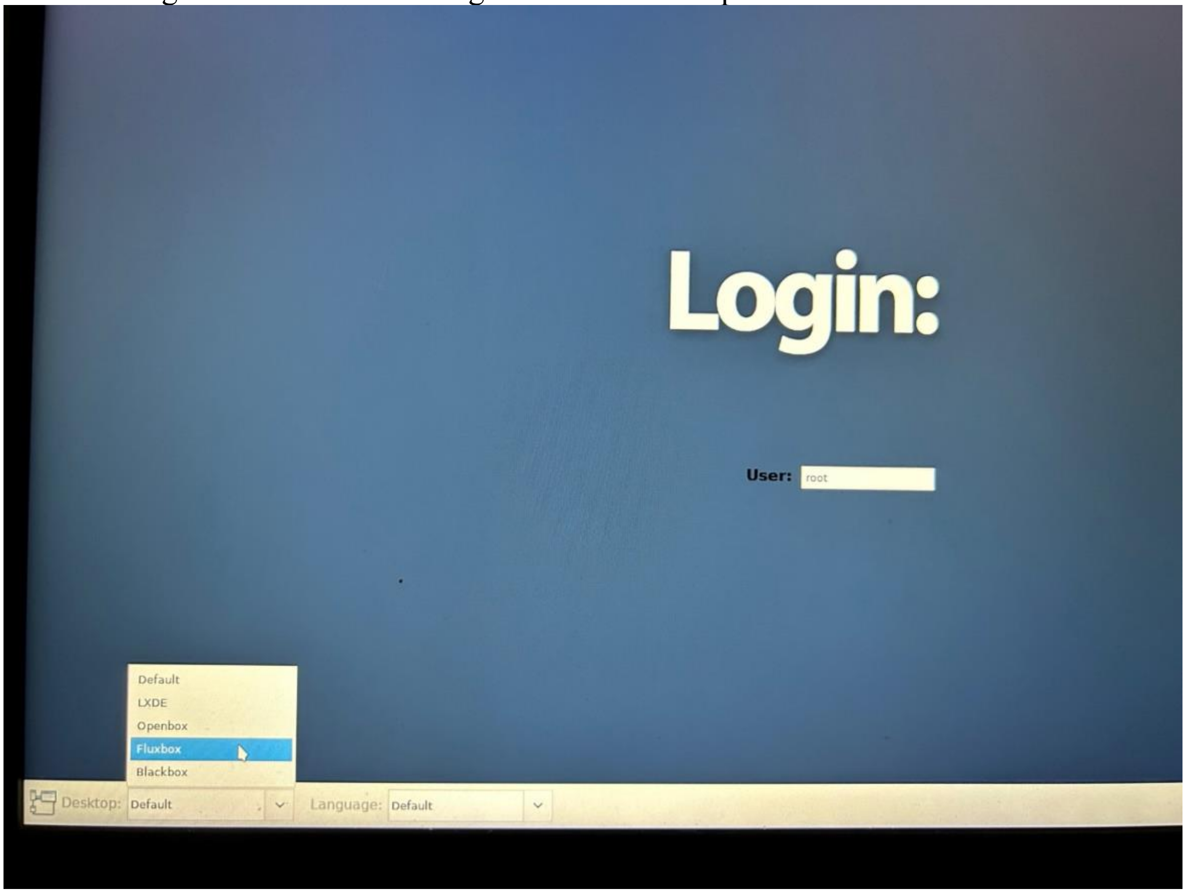
Figure 8: Guest Account log in with the desktop environment 'Default.'



Log in as Root and Remove Unnecessary Applications From the System

To continue setting up our environment, we must log in to the system as root. It is essential to notice how the root account is strictly tied to the 'Fluxbox desktop, unlike the guest account, which must use the default desktop environment.

Figure 9: Root Account log in with the desktop environment 'Fluxbox.'


Login:

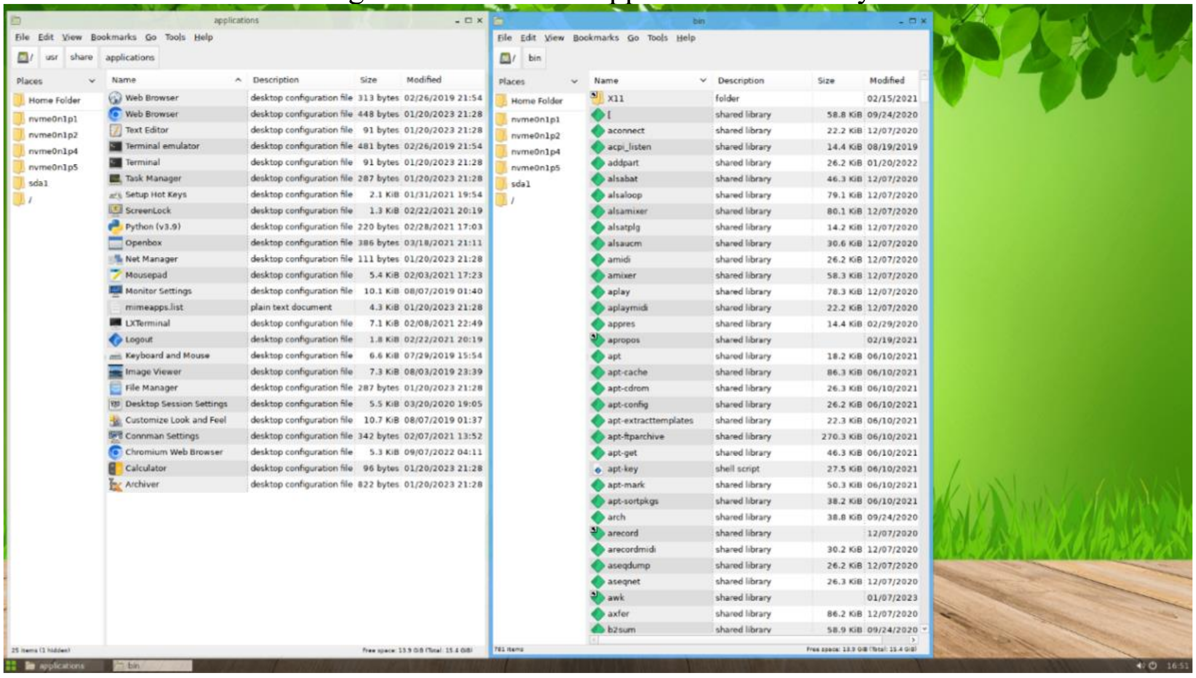
User: root

Desktop: Default

Language: Default

Upon logging in as root with the Fluxbox desktop, we can use the file explorer to navigate into the following directory: /usr/share/applications. Once we locate the directory, we will see several applications, such as a text editor, a terminal emulator, a calculator, etc. Each application is a shortcut to the executable file under the /bin directory. We can delete the shortcut from /usr/share/applications and its corresponding binary. Doing so may damage the display manager settings and the desktop environment manager. After intensive testing, we concluded that leaving the binary files intact is best. Moreover, the applications folder comprises the applications available to the guest user. To prevent any act of academic misconduct during testing, we decided to give access to the web browser only, along with some display manager settings. We can log out from the root account and log in with the guest profile to see what has been made available for the guest account, which is the account that will be utilized for testing.

Figure 10: /usr/share/applications directory



Name	Description	Size	Modified
Web Browser	desktop configuration file	313 bytes	02/26/2019 21:54
Web Browser	desktop configuration file	448 bytes	01/20/2023 21:28
Text Editor	desktop configuration file	91 bytes	01/20/2023 21:28
Terminal emulator	desktop configuration file	481 bytes	02/26/2019 21:54
Terminal	desktop configuration file	91 bytes	01/20/2023 21:28
Task Manager	desktop configuration file	287 bytes	01/20/2023 21:28
Setup Hot Keys	desktop configuration file	2.1 KiB	01/31/2021 19:54
ScreenLock	desktop configuration file	1.3 KiB	02/22/2021 20:19
Python (v3.9)	desktop configuration file	220 bytes	02/28/2021 17:03
Openbox	desktop configuration file	386 bytes	03/18/2021 21:11
Net Manager	desktop configuration file	111 bytes	01/20/2023 21:28
Mousepad	desktop configuration file	5.4 KiB	02/03/2021 17:23
Monitor Settings	desktop configuration file	10.1 KiB	08/07/2019 01:40
mimeapps.list	plain text document	4.3 KiB	01/20/2023 21:28
LXTerminal	desktop configuration file	7.1 KiB	02/08/2021 22:49
Logout	desktop configuration file	1.8 KiB	02/22/2021 20:19
Keyboard and Mouse	desktop configuration file	6.6 KiB	07/29/2019 15:54
Image Viewer	desktop configuration file	7.3 KiB	08/03/2019 23:39
File Manager	desktop configuration file	287 bytes	01/20/2023 21:28
Desktop Session Settings	desktop configuration file	5.5 KiB	03/20/2020 19:05
Customize Look and Feel	desktop configuration file	10.7 KiB	08/07/2019 01:37
Connman Settings	desktop configuration file	342 bytes	02/07/2021 13:52
Chromium Web Browser	desktop configuration file	5.3 KiB	09/07/2022 04:11
Calculator	desktop configuration file	96 bytes	01/20/2023 21:28
Archiver	desktop configuration file	822 bytes	01/20/2023 21:28

25 items (1 hidden)   Free space: 13.9 GiB (Total: 15.4 GiB)

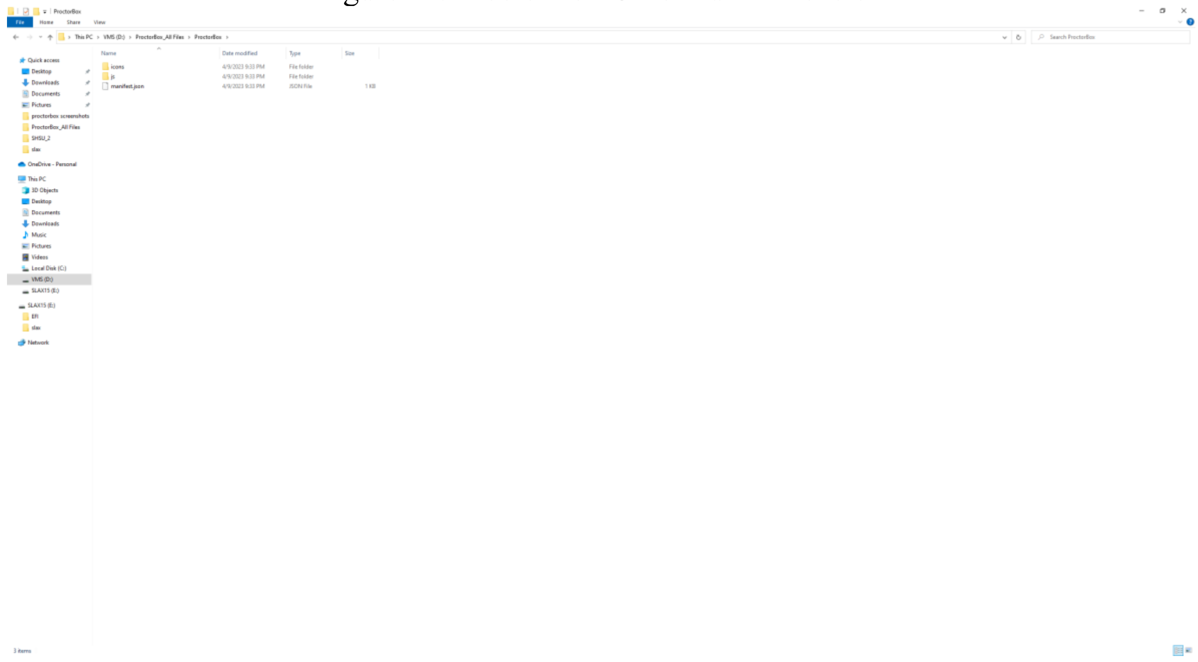
Name	Description	Size	Modified
X11	folder		02/15/2021
[	shared library	58.8 KiB	09/24/2020
aconnect	shared library	22.2 KiB	12/07/2020
acpi_listen	shared library	14.4 KiB	08/19/2019
addpart	shared library	26.2 KiB	01/20/2022
alsabat	shared library	46.3 KiB	12/07/2020
alsaloop	shared library	79.1 KiB	12/07/2020
alsamixer	shared library	80.1 KiB	12/07/2020
alsatplg	shared library	14.2 KiB	12/07/2020
alsaucm	shared library	30.6 KiB	12/07/2020
amidi	shared library	26.2 KiB	12/07/2020
amixer	shared library	58.3 KiB	12/07/2020
aplay	shared library	78.3 KiB	12/07/2020
aplaymidi	shared library	22.2 KiB	12/07/2020
appres	shared library	14.4 KiB	02/29/2020
apropos	shared library		02/19/2021
apt	shared library	18.2 KiB	06/10/2021
apt-cache	shared library	86.3 KiB	06/10/2021
apt-cdrom	shared library	26.3 KiB	06/10/2021
apt-config	shared library	26.2 KiB	06/10/2021
apt-extracttemplates	shared library	22.3 KiB	06/10/2021
apt-ftparchive	shared library	270.3 KiB	06/10/2021
apt-get	shared library	46.3 KiB	06/10/2021
apt-key	shell script	27.5 KiB	06/10/2021
apt-mark	shared library	50.3 KiB	06/10/2021
apt-sortpkgs	shared library	38.2 KiB	06/10/2021
arch	shared library	38.8 KiB	09/24/2020
arecord	shared library		12/07/2020
arecordmidi	shared library	30.2 KiB	12/07/2020
aseqdump	shared library	26.2 KiB	12/07/2020
aseqnet	shared library	26.3 KiB	12/07/2020
awk	shared library		01/07/2023
axfer	shared library	86.2 KiB	12/07/2020
b2sum	shared library	58.9 KiB	09/24/2020

781 items   Free space: 13.9 GiB (Total: 15.4 GiB)

Implementing Browser-Level Security

Now that the system meets our desired requirements, it is time to work on the browser-level security before making the changes permanent. To accomplish this task, the best approach is to build a Chromium extension with two basic functionalities. The first task of the Chromium extension is to turn off the right click; this will prevent the student from having access to the context menu, which may facilitate academic dishonesty during an examination. The second function of our Chromium extension is to allow only one tab at a time. One tab only aims to take full advantage of the exam logs provided by our learning management systems. The higher ed market is compromised by Blackboard and Canvas users. Each time an assessment takes place, the learning management system provides full logs until the student submits an exam. Therefore, by allowing only one tab at a time, we can read the logs and identify if there was a period when the student started another session. Moreover, we discovered that the one-tab limit prevents the student from opening the settings in Chromium.

Figure 11: ProctorBox Chromium Extension



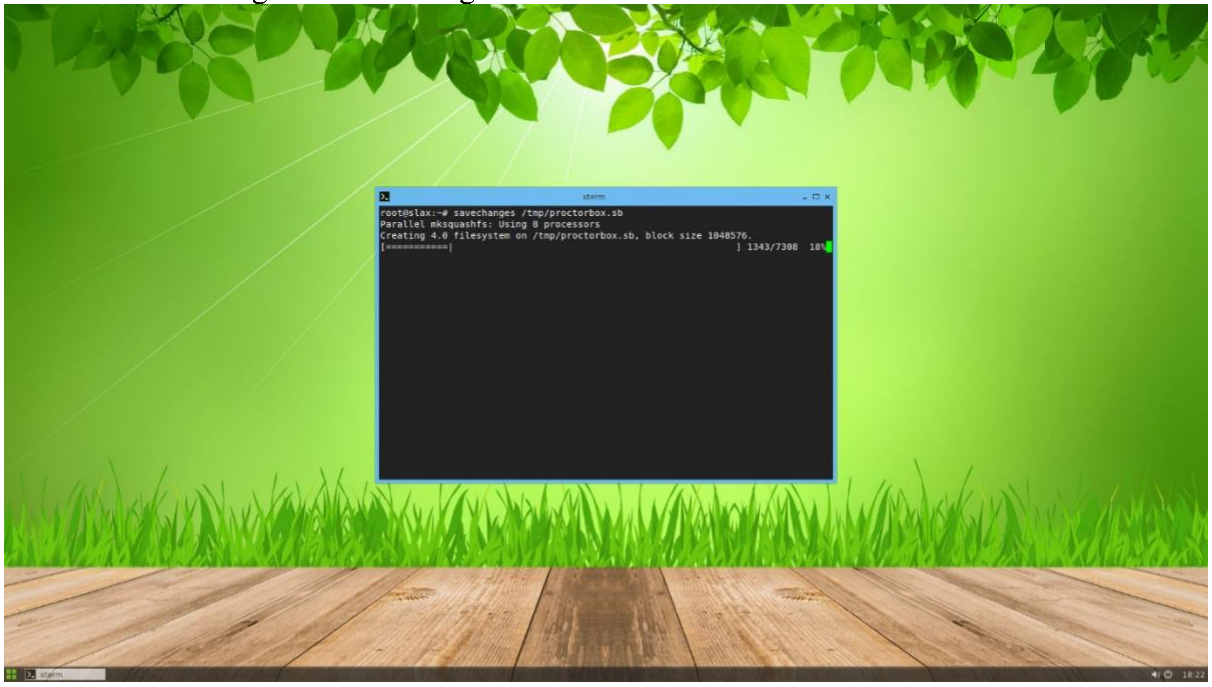
Name	Date modified	Type	Size
icons	4/9/2023 9:33 PM	File folder	
js	4/9/2023 9:33 PM	File folder	
manifest.json	4/9/2023 9:33 PM	JSON File	1 KB

Figure 11 represents the root folder of our Chromium extension. The icons folder inside the root contains the required extension icons. Next, the js folder has the two scripts that handle the extension's behavior. In this case, there is a script to turn off the right click and another script to allow only one tab at a time. Finally, we have manifest.json file, which contains all the configurations for the ProctorBox extension. To access this folder and its content, log in to the system as root and navigate to /usr/share/applications. A copy of the folder is located there and has been loaded into Chromium for the root and guest accounts.

Building the ProctorBox Module to Make the Changes Permanent.

Now that the system has met all our requirements, we can make the changes permanently by creating a module. Note for future system owners, you will use the savechanges command anytime you make any adjustments to the ProctorBox system. To start the process, click the start menu and launch the terminal. Once the terminal appears, type the command savechanges /tmp/proctorbox.sb. The purpose of this command is to create the proctorbox.sb module under the /tmp folder. A progress bar will appear after a few seconds. Please wait for this process to finish before any other system modifications. Once the savechanges command is complete, locate the file in the /tmp directory and click copy. Now, Paste the proctorbox.sb module into /sda1/slax/modules directory.

Figure 12: Creating the ProctorBox Module from Terminal.



```
root@slax:~# savechanges /tmp/proctorbox.sb
Parallel mksquashfs: Using 8 processors
Creating 4.0 filesystem on /tmp/proctorbox.sb, block size 1048576.
[===========|                                                    ] 1343/7308  18%
```

Preparing ProctorBox for Distribution.

We can start distributing our ProctorBox platform to the public, end users, students, and system owners, and the process is simple. Go back to your Windows PC and plug in the proctor box USB drive that contains the proctobox.sb module. Prepare a secondary blank drive similar to the process we completed in Figure 2. Once the drive has been formatted, paste the two folders 'EFI' & 'slax' into the root directory of your blank drive. Finally, on your new target drive, navigate to /slax/boot and run the bootinst file to prepare our new target drive for boot.

Conclusion and Future Work

ProctorBox represents a significant leap in computerized proctoring applications and platforms, serving as a benchmark for the integrity and security of online examinations. The open-source nature of our platform empowers a global community of academics to constantly contribute to its evolution resulting in an ever-improving, robust proctoring platform that can effectively meet the demands of modern distance learning education. From advanced proctoring features such as trailblazing browser security to its intuitive user interface ProctorBox caters to educators and students alike, promoting an environment of fairness and ease. Moreover, ProctorBox is not just a proctoring platform but a pioneer in offering no-cost proctoring to students worldwide. Our unparalleled scalability and interoperability make ProctorBox a leading platform for online examinations, setting the gold standard for secure, fair, and transparent online examinations.

References

Abozaid, A. M., & Atia, A. (2022). Multi-Modal Online Exam Cheating Detection. *International Conference on Electrical, Computer and Energy Technologies (ICECET).* https://doi.org/10.1109/icecet55527.2022.9873527

Alfredo Barrientos Padilla, Cuadros, M. A., Alba, J., & Becerra, A. (2021). *Implement a remote system for the supervision of online exams through cameras with artificial intelligence*. https://doi.org/10.1109/eircon52903.2021.9613352

The Chromium Projects. (n.d.). Www.chromium.org. https://www.chromium.org/chromium-projects/

Definition of operating system | Dictionary.com. (n.d.). Www.dictionary.com. Retrieved June 1, 2023, from https://www.dictionary.com/browse/operating-system

Definition of proctor | Dictionary.com. (2019). Www.dictionary.com. https://www.dictionary.com/browse/proctor

Dictionary.com. (2019). *Definition of Integrity | Dictionary.com*. Www.dictionary.com. https://www.dictionary.com/browse/integrity

Editor, C. C. (n.d.-a). *Hardening - Glossary | CSRC*. Csrc.nist.gov. Retrieved June 1, 2023, from https://csrc.nist.gov/glossary/term/Hardening

Editor, C. C. (n.d.-b). *least privilege - Glossary | CSRC*. Csrc.nist.gov. Retrieved June 1, 2023, from https://csrc.nist.gov/glossary/term/least_privilege

Ganidisastra, A. H. S., & Bandung, Y. (2021, April 1). *An Incremental Training on Deep Learning Face Recognition for M-Learning Online Exam Proctoring*. IEEE Xplore. https://doi.org/10.1109/APWiMob51111.2021.9435232

Grigoriev, I. S. (2022). *Service for Monitoring and Control of Remote Testing by Video Information*. https://doi.org/10.1109/scm55405.2022.9794842

LXDE. (n.d.). Www.lxde.org. Retrieved June 1, 2023, from http://www.lxde.org/

LXDM - ArchWiki. (n.d.). Wiki.archlinux.org. Retrieved June 1, 2023, from https://wiki.archlinux.org/title/LXDM

M, T. (n.d.). *Slax Linux - your pocket operating system*. Www.slax.org. Retrieved June 1, 2023, from https://www.slax.org/

Madhusudan, Godbole, S., Meshram, H., & Joshi, J. (2022, May 1). *Automated Exam Proctoring using a Support Vector Machine and Histogram of Oriented Gradients*. IEEE Xplore. https://doi.org/10.1109/ICAAIC53929.2022.9792952

Malhotra, N., Suri, R., Verma, P., & Kumar, R. (2022, February 1). *Smart Artificial Intelligence-Based Online Proctoring System*. IEEE Xplore. https://doi.org/10.1109/DELCON54057.2022.9753313

Open-source Definition & Meaning. (n.d.). Dictionary.com. Retrieved June 1, 2023, from https://www.dictionary.com/browse/open-source

Shakti Priya Saurav, PANDEY, P., Sharma, S., Pandey, B. K., & Kumar, R. (2021). AI-Based Proctoring. *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. https://doi.org/10.1109/icac3n53548.2021.9725547

Solis, R. D., Narasimha Shashidhar, & Cihan Varol. (2023). *Automated Proctoring Solutions: Modern Techniques to Evade and Lure Computerized Proctoring Systems*. https://doi.org/10.22492/issn.2189-1036.2023.30

Contact emails: rds050@shsu.edu
nks001@shsu.edu
cxv007@shsu.edu
axr249@shsu.edu