PROJECT BUXS

Jared Andraszek

B.S. Computer Science and Cyber Security

Professor Henderson

Statement of Purpose:

Project BUXS (Browser User eXfiltration System) was developed to address the growing need for practical, hands-on training in offensive cybersecurity. Many cybersecurity students graduate with theoretical knowledge but lack the experience needed to understand and execute real-world exploits. This project creates a realistic, sandboxed environment inspired by platforms like Capture The Flag (CTF) and Hack The Box, where users can explore and practice browser-based attack vectors such as GET header manipulation, Cross-Site Scripting (XSS), and SQL injections (SQLi). Designed for educational use at Charleston Southern University, Project BUXS empowers learners to think creatively, experiment safely, and gain confidence in ethical hacking techniques through a self-paced lab that simulates professional penetration testing scenarios.

Research & Background:

The development of Project BUXS was inspired by widely recognized cybersecurity training platforms such as Capture The Flag (CTF) challenges and Hack The Box. These platforms emphasize self-guided exploration, creativity, and a trial-and-error approach to learning offensive security. Project BUXS builds on these principles by focusing specifically on browser-based vulnerabilities within a controlled lab environment.

The research phase involved identifying common web application vulnerabilities that regularly appear in real-world attacks and industry-standard penetration tests. These include File Inclusion, Cross-Site Scripting (XSS), and SQL Injection (SQLi)—all of which can be leveraged to gain unauthorized access, exfiltrate sensitive data, or pivot within a network. To support these use cases, Project BUXS was built using the LAMP stack (Linux, Apache, MySQL/MariaDB, PHP) hosted on a VMware ESXi infrastructure provided by Charleston Southern University.

Additionally, security techniques and countermeasures for these vulnerabilities were studied in-depth to better understand how to intentionally weaken or bypass them in a safe and reproducible manner. This dual understanding of offense and defense shaped the design of the lab to ensure it provided educational value while maintaining a secure boundary for experimentation.

Project Languages(s), Software, and Hardware:

Languages:

- HTML/CSS – For designing the structure and appearance of the web interfaces.
- JavaScript – For dynamic client-side scripting and simulating XSS attacks.
- PHP – To handle server-side logic, session management, and integration with the database.

- SQL – For managing and querying data within the vulnerable MySQL/MariaDB databases.
- Bash & Python – For server automation, utility scripts, and the XSS exfiltration listener tool.

Software:

- Apache Web Server – Hosts both vulnerable web applications and delivers dynamic content.
- MariaDB (MySQL) – Acts as the backend database for user data, admin panels, and input storage vulnerable to SQLi.
- Debian 12 – Serves as the operating system for the virtual machines, offering stability and compatibility with the required software.
- VMware ESXi Server – Hosts the lab environment on Charleston Southern University's Lazarus server, allowing students to access isolated virtual machines.

Hardware:

- CSU's ESXi Server Lazarus - The physical infrastructure supporting the web servers.

Project Requirements:

- Accessibility for Students - The lab needed to be easy to access on-campus via Charleston Southern University's Lazarus server, and also adaptable for off-campus use through VPN or local VM setups.
- Realistic Simulated Environment - To enhance immersion, the lab environment mimics the internal infrastructure of a fictional tech company, Lumino, complete with authentic-looking websites and admin portals.
- Built-In Exploits - The virtual web servers were purposefully designed with common and realistic web vulnerabilities such as GET header manipulation, XSS, and SQL injection, ensuring a comprehensive learning experience.
- Guided Learning Path - The environment includes subtle hints and guidance embedded in the interface to help students progress while still encouraging critical thinking and exploration.
- VM Network Configuration - The project required a virtual network of at least two interlinked web servers to simulate lateral movement across trusted services, which is crucial for demonstrating multi-stage attacks.

Project Implementation Description & Explanation:

Project BUXS was implemented as a multi-tiered virtual lab consisting of two intentionally vulnerable web servers designed to simulate real-world web applications. The frontend for both servers was built using HTML, CSS, and JavaScript, while backend logic was handled via PHP and a MariaDB database. The servers were deployed on Charleston Southern University's Lazarus VMware ESXi server and configured to mimic a realistic corporate environment, complete with internal documentation, hidden files, and segmented access levels.

The first web server (lumino) acts as the entry point and is designed to look like a realistic cloud service provider that has a hero page, some fictional statistics and fake customer testimonials (see Fig 1 for the homepage and Fig 2 for the homepage testimonials).

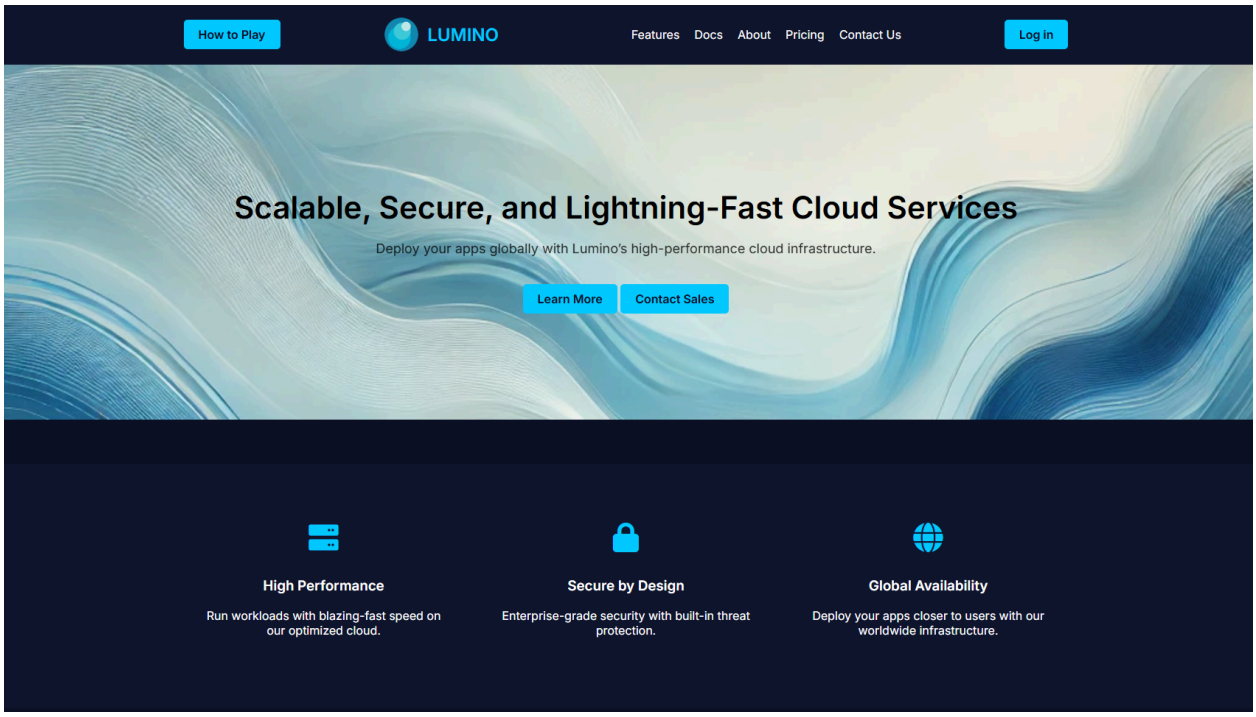


Fig 1. Lumino homepage

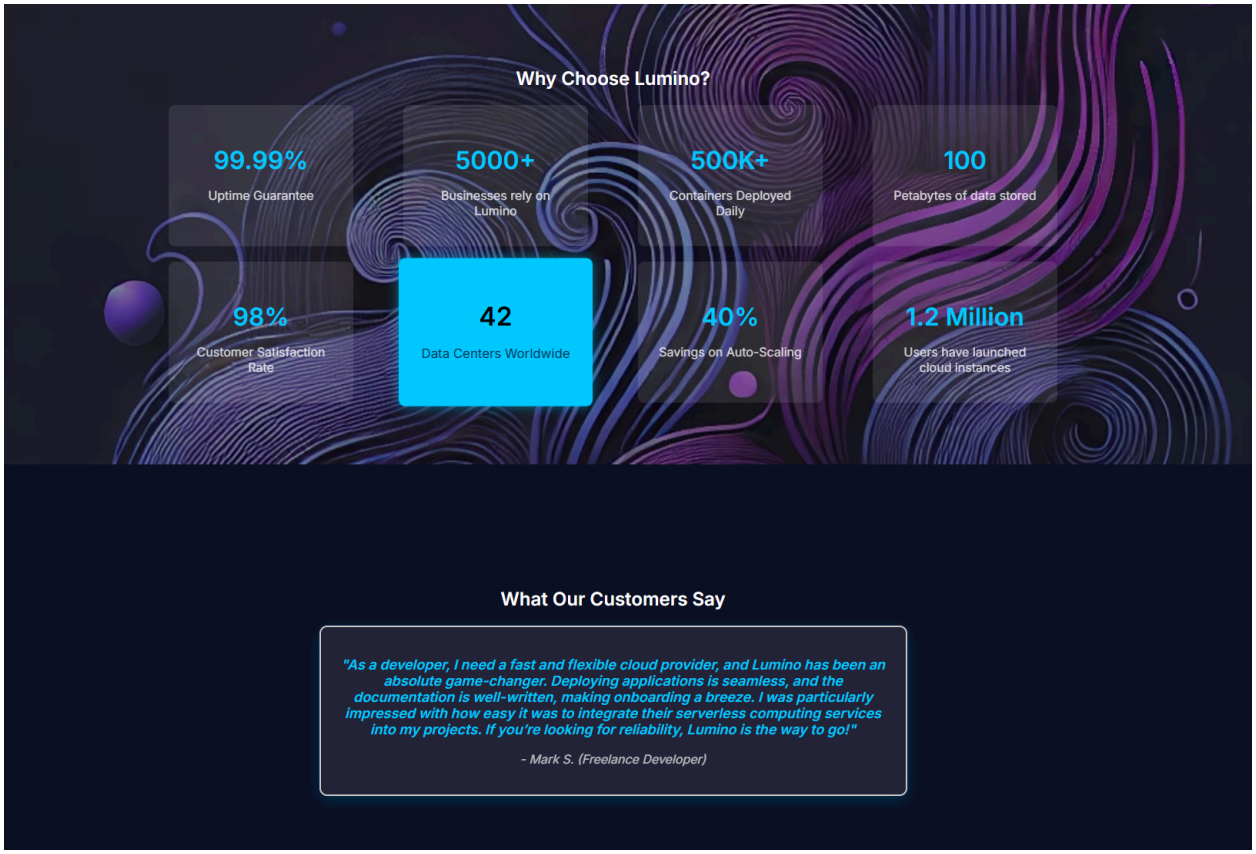


Fig 2. Lumino Homepage testimonials and information cards

The Lumino web server also has a built-in exploit page with a file inclusion vulnerability where users need to navigate to the documents page which shows a list of available files that the user can open and read as shown in Fig 3. For the exploit to work, the user must know the file name they are looking for. Through hints available around the website, the user can deduce that the file is called Flint.txt, which is not available in the dropdown. They will also need to change the GET headers to not say User=Access+File and instead remove the variable completely or change it to Admin=Access+File. Upon successful completion of changing the headers, the file reveals the location of the next web server.

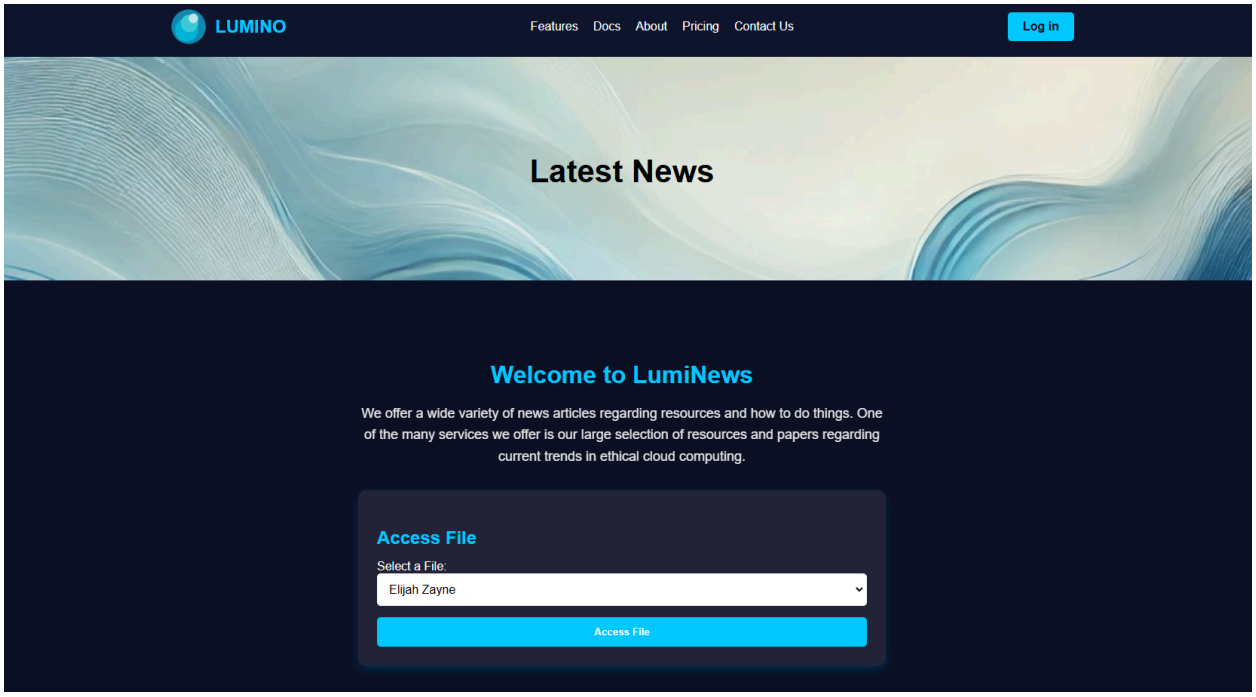


Fig 3. LumiNews Page

The second server (Lumino-portal) contains more complex vulnerabilities including a stored XSS and SQL injection. It simulates an internal user portal where users must create an account (see Fig 4) and exploit a ticket form (see Fig. 5) in order to gain privileged access to an admin account via a session hijacking exploit. Users get immediate feedback on if their inject worked by seeing the exploit run on their screen (see Fig. 6). To complete the lab, students exfiltrate session cookies through a custom Python-based XSS listener and ultimately achieve administrator-level access. Generally students can do this by creating a Python HTTP listener and send the cookie information to themselves.

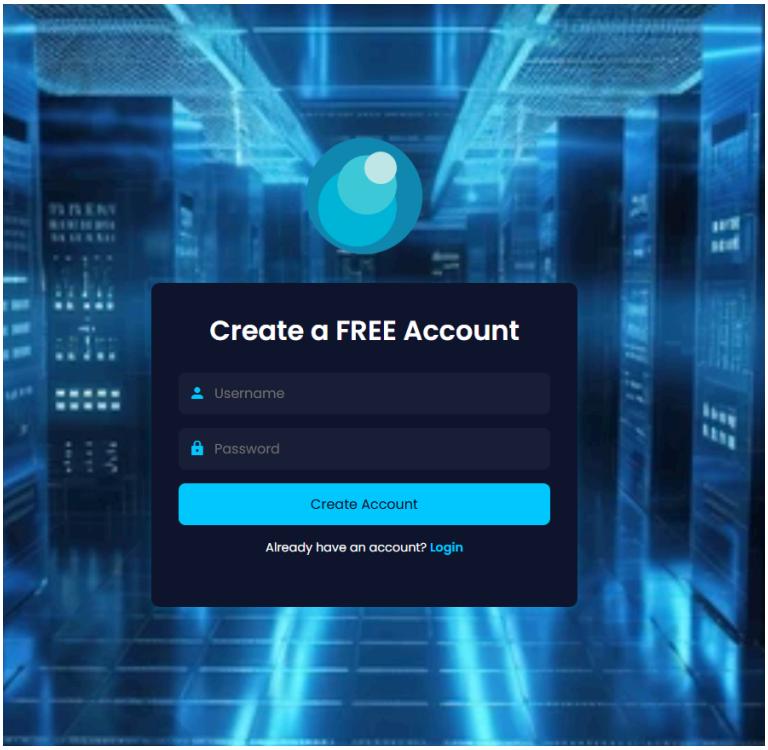


Fig 4. Lumino-portal create account page

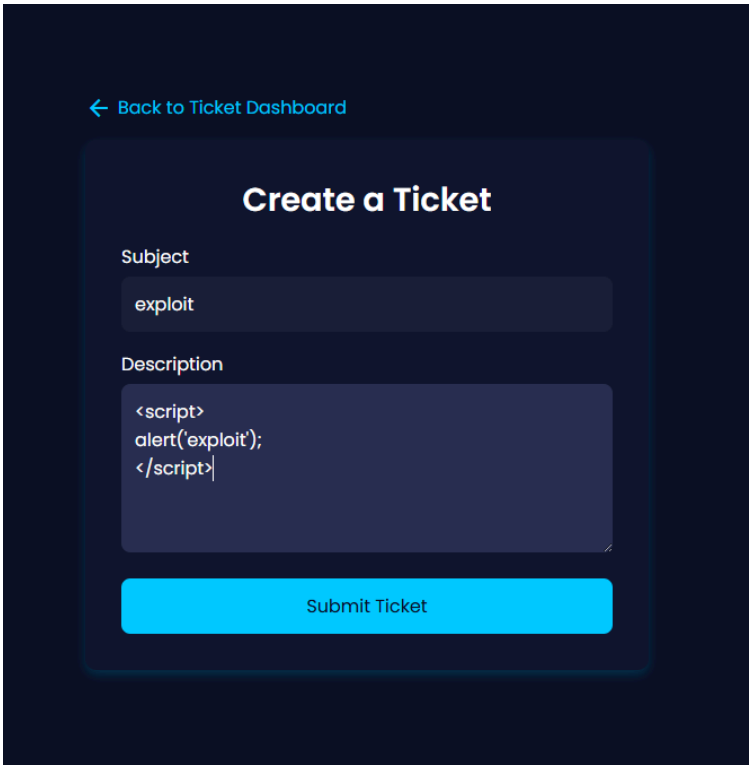


Fig 5. Vulnerable XSS inject ticket form

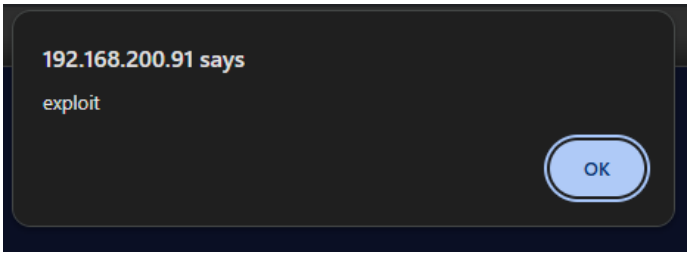


Fig 6. Exploit showing immediate user feedback

Once the player gets the admin cookies, they can change their PHPSESSID cookie to that of the admin's and they are now logged in as the admin. Once they are admin, they can either create themselves an admin account or view the Internal Operations page (see Fig 7) which has an SQL injection exploit. Users can search for more information about the table in the internal operations page via the input field which takes their input and places it into an SQL statement with no input validation. Allowing users to change the script and find the secret file.

Fig 7. Internal Operations Page with SQLi exploit

Throughout development, emphasis was placed on ensuring the environment felt authentic and provided learning opportunities rather than simple “checklist” hacking steps. Subtle hints are embedded in the interface to encourage students to explore and discover exploits organically (see Fig. 8. Embedded hint in the contact section). Additionally, a “How To Play” page introduces users to the lab and links to a Google Form, which, upon completion, triggers the generation of a customized Certificate of Completion (see Fig. 9 and 10. How to Play and Certificate of Completion).

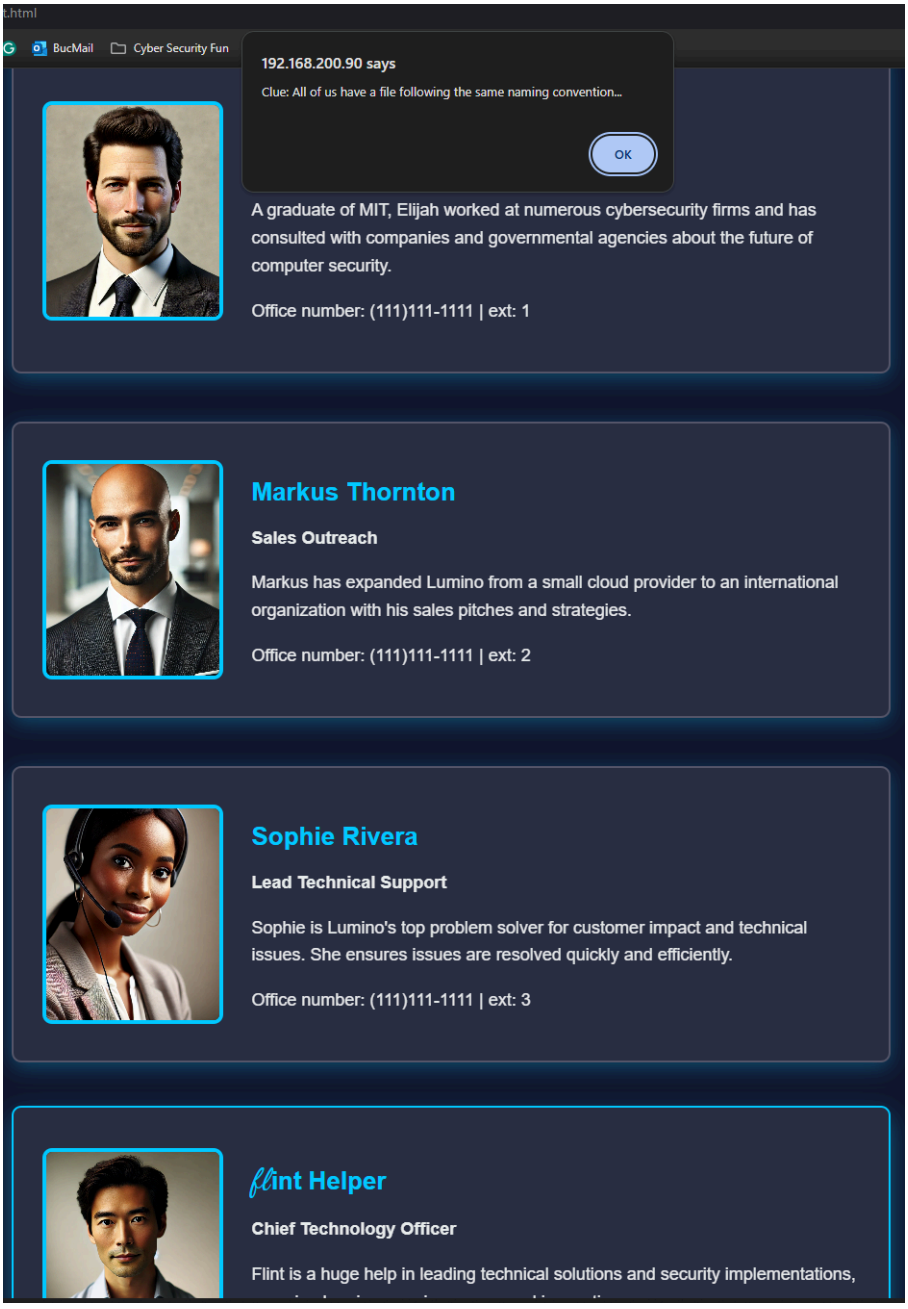


Fig 8. Flint Hint on Contact page

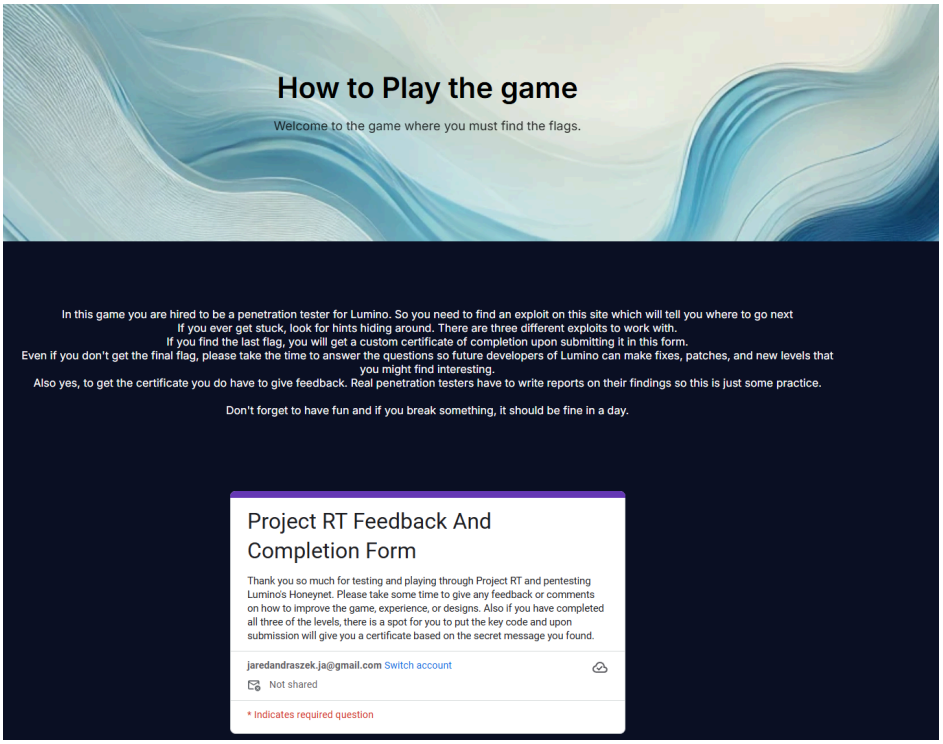


Fig 9. Lumino How to Play page

Certificate of Completion

Jared Andraszek

Completed the Charleston Southern University Hacking Challenge

PROJECT RT

This Certificate shows that Jared Andraszek has completed introductory level challenges for manipulating GET header variables to access a hidden file, use a XSS attack to steal a PHP session token, and use SQL injections against a MySQL database to find a secret message.

3/31/2025

Fig 10. Certificate of Completion

The source code for Project BUXS is publicly available for educational use and contributions on GitHub at: https://github.com/JaredAndraszek42/Project_BUXS

Test Plan:

The testing process for Project BUXS was designed to ensure the system's stability, realism, and educational effectiveness. It was conducted in multiple phases, each targeting a specific aspect of the project's functionality and user experience. During unit testing, individual components such as PHP authentication scripts, SQL query handling, and GET header parsing were evaluated in isolation. This allowed me to confirm that each feature worked as expected before integrating it into the web servers. For example, special attention was paid to XSS and SQL injections to ensure they would respond properly to crafted payloads.

Integration testing followed, focusing on the interactions between various system components. This phase tested the exploit flow, confirming that progressing from one exploit to the next functioned seamlessly. A round of user acceptance testing (UAT) was conducted by cybersecurity students at Charleston Southern University. Participants used campus lab devices connected via Ethernet to access the simulated corporate environment. They followed the in-built "How To Play" guide and submitted feedback through a Google form embedded in the site. This feedback was crucial in identifying usability improvements, bugs, and additional opportunities to enhance the learning experience.

The primary test objectives were successfully met. Users were able to access both web servers without issue, the exploits worked as intended, and the hint system effectively guided students without revealing solutions outright. Feedback indicated that the websites felt realistic and immersive, closely mimicking the look and feel of a small technology company.

Several features were successfully implemented based on test feedback, including the use of a domain name (http://lumino) to enhance immersion, the integration of a "How To Play" guide directly on the site, and an automated Certificate of Completion system using a Google Form. Numerous bugs and unintended vulnerabilities were identified and resolved throughout the process. However, some planned features remain unimplemented. Notably, the virtual machines currently do not restore from a snapshot automatically each night, and a preconfigured XSS attack that does not require a remote listener is still under development. Additionally, the addition of more levels and exploit scenarios is planned for future versions but was not within the scope of this release.

Overall, the test plan ensured that Project BUXS delivered on its goals of providing a hands-on, realistic offensive security environment. The iterative feedback loop between students and the developer helped refine the experience and confirm the educational value of the platform.

Challenges Overcome:

Throughout the development of Project BUXS, several technical and design challenges had to be addressed to ensure the project was not only functional but also educational and immersive. One of the earliest challenges was crafting a compelling and realistic narrative

environment. To keep students engaged, the lab needed to feel like a real-world scenario rather than a generic penetration testing exercise. Achieving this required thoughtful design of both the user interface and the in-game content to resemble a legitimate tech company's web infrastructure.

Another major hurdle was setting up and maintaining a reliable full-stack environment. The project relied on a combination of Apache, HTML, CSS, JavaScript, PHP, and MySQL that had to function consistently across both virtual machines. Integrating these technologies, especially within Lumino-portal, introduced complications such as PHP and MySQL integrations, database syncing, and security misconfigurations that had to be carefully crafted and tested. I also had to prioritize the quality of the exploits over the quantity. Rather than overloading the project with numerous shallow vulnerabilities, I shifted the focus to building a few deep and well-structured attack paths that allowed students to explore multiple angles of exploitation rather than the multitude of exploits. This approach demanded more development time per exploit but ultimately led to a richer learning experience.

Automation posed additional difficulties. A custom Python script was developed to automatically check for XSS injection success, which required precise handling of cookie values and HTTP responses. Ensuring this script worked reliably in a virtualized environment added to the complexity since cron jobs and other automation tools were not working for me. The development workflow also presented logistical challenges. Initial development was conducted in Oracle VirtualBox as that was my home lab setup, but the final deployment had to be transitioned to a VMware ESXi server hosted by Charleston Southern University. Migrating the virtual machines between these platforms while preserving configurations and ensuring compatibility was a time-consuming step which required me to manually reset all of the setup and transfer the files.

Finally, implementing a custom Certificate of Completion system was a unique challenge. It required the integration of a Google Form that could collect participant data, trigger certificate generation, and send it to users automatically, while fitting seamlessly into the narrative and usability goals of the lab. Despite these hurdles, each challenge was met with persistence and a focus on the project's core goal: to provide students with an engaging, realistic, and hands-on offensive security environment.

Future Enhancements:

While I am happy with the progress of Project BUXS, there are several enhancements that I would like to add or see added in the future. One of the primary goals for future versions is to introduce additional levels and exploit scenarios. This would include vulnerabilities beyond the current ones such as buffer overflows, sending packets in plaintext, and misconfigured authentication mechanisms, providing students with a broader challenge set. Another major enhancement involves the integration of microservices that communicate over a network, simulating real-world enterprise architecture. This would allow students to observe and exploit

traffic between services using tools like Wireshark, deepening their understanding of lateral movement and network-based attacks.

To support a more seamless setup experience, I would like future versions to transition to Docker. Containerization would simplify the installation process, reduce dependency issues, and allow instructors or students to spin up the lab environment on nearly any machine or cloud platform with minimal configuration. There are also plans to improve the Certificate of Completion system, making it more robust and secure. This may involve shifting to a server-side certificate generation system that does not rely on third-party services, while still providing automated delivery and verification options. It would also be nice to not have it send to my google drive every time it generates a certificate.

Lastly, unimplemented features from the current version, such as automated VM snapshot restoration and built-in XSS payloads that do not require player-hosted listeners, remain high priorities. Implementing these features will improve reliability and reduce setup complexity, especially in classroom or competition environments.

Slides:

PROBLEM STATEMENT

- Cybersecurity students feel a lower retention in Cyber Security hands-on exercises
- Students may graduate college with a degree, but with no firm understanding of common exploits
- No environment to practice ethical hacking and penetration testing

- Solution Requirements:
  - Promoting user creativity with multiple solutions to a problem
  - Covering multiple common web exploits
  - Self-paced sandbox environment

RESEARCH & BACKGROUND

- Capture The Flag and Hack The Box inspired
- Researched common web application vulnerabilities
- Apache server running the HTML, PHP, JS, CSS, MySQL development stack
- How to secure File inclusion, Cross Site Scripting (XSS), and SQL injections (SQLi) vulnerabilities and purposefully make them less secure

LANGUAGES, SOFTWARE, AND HARDWARE

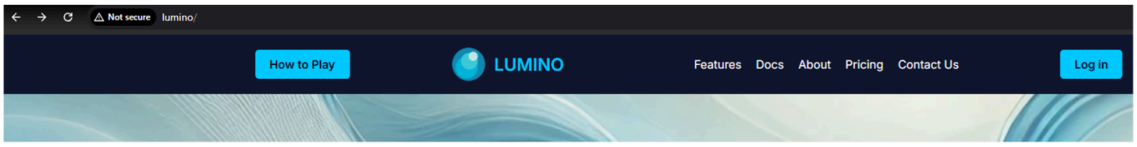
- Languages:
  - HTML/CSS
  - JS
  - PHP
  - SQL
  - BASH
  - Python
- Software/OS:
  - Vmware ESXi server
  - Apache webserver
  - Debian 12 OS
  - MariaDB (MySQL)
- Hardware:
  - CSU's Lazarus ESXiserver

PROJECT REQUIREMENTS

- Make it accessible to students
- Create a network of VMs hosting webservers
- Contains common exploits built-in to the servers
- Feel realistic to where it looks like you are working with a real tech company
- Include guidance and tips to progress the students forward

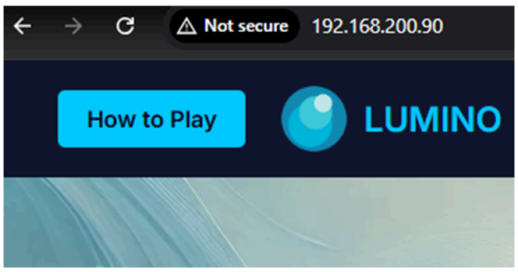
HOW TO GET STARTED – ON CAMPUS

- You must have connection to the Lazarus server. So on a lab device in Ashby Hall 203, 207, or208
- Or a laptop/personal device with an ethernet port plugged into the ethernet wall jacks
- Go to http://Lumino
- Click on How To Play in the top left
- Have fun



HOW TO GET STARTED – OFF CAMPUS

- Request VPN permissions and then follow ON CAMPUS instructions
- Or set up VMs on a personal network (Recommend Oracle VirtualBox)
- Change the hidden file to point to the VM2 IP address
- Go to the IP address set for VM1 and only after completing VM1 should you move to VM2's IP address
- Have Fun



DEMO

DEMO TIME

Go to http://lumino

CERTIFICATE OF COMPLETION

- Upon completion of the Google form:
  - Puts the player's name in
  - Puts the date
  - Sends the completed certificate to their email

Certificate of Completion

Jared Andraszek

Completed the Charleston Southern University Hacking Challenge

PROJECT RT

This Certificate shows that Jared Andraszek has completed introductory level challenges for manipulating GET header variables to access a hidden file, use a XSS attack to steal a PHP session token, and use SQL injections against a MySQL database to find a secret message.

3/31/2025

TEST PLAN

- Testing Plan:
  - Unit tests during development
  - Design and improvement feedback during development
  - User acceptance and bug hunting from Cyber Security students

- Main Testing Cases:
  - Users can access the webservers and properly use their functionalities
  - The websites feel like a real website
  - The exploits function as intended
  - The hints work and direct and guide players as intended
  - The VMs restores from to a snapshot every night

TEST RESULTS - IMPLEMENTED

- Users requested a domain name for more realism
- Created a How to Play with the Google submission form built in to the website
- Created a Certificate of Completion based on Google form submission
- Implemented bug and unnecessary vulnerability fixes

TEST RESULTS – NOT IMPLEMENTED

- Not yet implemented/Work in progress
  - XSS inject that does not require the player to create a remote listener
  - VMs do not restore from a snapshot every night
- Do not plan to implement (in this version)
  - More levels

CHALLENGES OVERCAME

- Creating a narrative and realistic design
- Apache, HTML, CSS, JS, PHP, MySQL full-stack environment
- Prioritizing quality of the exploits over the number of exploits
- Automating the XSS check
- Developing in Oracle Virtualbox and transferring to Vmware ESXi
- Creating a Certificate of Completion

FUTURE ENHANCEMENTS

- In future versions of this application, I would like to see/add:
  - More levels
    - Wireshark
    - Buffer Overflow
    - More common web exploits
  - Microservices that communicate over a network that can be captured or read
  - Fixing unintended user reported bugs
  - Creating a better Certificate of Completion
  - Transition to Docker for easy transportation and setup

QUESTIONS?