Blackbox Penetration Testing with Ghost Box[1]

Web application Security Analysis Report

Grant Li, Nicolas Zheng, Han Cho, Phoenix Liu

Abstract

This project aims to find vulnerabilities in the Ghost Box's software system and provide recommendations to mitigate discovered vulnerabilities. Ghost Box provides storage infrastructure for the physical assets of users, and customers' personal information is used to secure users' physical assets. Since customer information is the main asset in protecting users' physical assets, a breach would cause a significant burden on both the user and the company. Ghost Box's system consists of a web application, a mobile application, a backend, and an IoT system. Our analysis involved black-box penetration tests and automatic scans on the system's web application part only and credential hunting to find potential password reuse attacks.

1 Introduction

The project aims to conduct a black-box security analysis for Ghost Box's software system to find as many vulnerabilities as possible and provide concrete recommendations they can follow to mitigate them. As common to small startups, Ghost Box operates on narrow profit margins. Their software is also entirely outsourced to contractors, meaning they do not have the expertise or the budget to conduct proper security analysis. This makes the software vulnerable and susceptible to cybersecurity attacks, especially considering its objective is to store user assets.

The importance of the analysis is that it ensures Ghost Box can eliminate as many attack vectors as possible and safeguard the company's reputation by preventing user information leaks, loss of user assets, and unauthorized access to internal systems. During the COVID-19 pandemic, cyberattacks have become an elevated risk, with one in five small businesses being affected by a cyberattack or data breach [1]. Customer information represents the main assets Ghost Box can be targeted for, and a breach entails a significant financial burden on the company in the form of a regulatory fine [2]. This information could also be used for spoofing and criminal offenses such as stalking, resulting in more damage to users. User physical assets stored in Ghost Box's IoT system are also incentives for cyberattacks, and the loss of these assets would inflict direct financial loss on the users. Compromise of either of the above can cause reputation damage to Ghost Box and loss of trust of users. Thus, our black-box testing effort to minimize the change of such damage is paramount.

Ghost Box's system aims to provide secure storage of assets. The system consists of a web application, a mobile application, a backend, and an IoT system. The analysis was only conducted as black-box penetration testing on the system's web application part; code-based security analysis is not possible because the system owner does not consent to give us access to the source code. Any testing attempt that would cause changes in the system was safely carried out in a testing environment provided by the client.

While our analysis is the first formal security analysis conducted on Ghost Box's system, we reference a security analysis done on a live online voting system and a black-box internal penetration testing guide for analysis methodology for black-box testing a web application. We will also consult web-based analysis projects from the previous cohort for detailed steps they performed to uncover flaws in the system.

Our analysis involved testing the web application component of Ghost Box's system through security scans and manual analysis to discover vulnerabilities in the system's design and implementation. We also attempted credential hunting to search for the leaked password for password reuse attacks. Thus our analysis covers both the technological and human aspect of cybersecurity.

While our primary focus was to compromise the confidentiality of data, which we succeeded in doing, we could also attack data integrity via API calls. Our conclusion is that Ghost Box's system was not designed or implemented with cybersecurity in mind. We alerted the Ghost Box system owner immediately when such high-risk flaws were discovered, and they were able to patch these with temporary fixes.

We recommend redesigning the current Ghost Box web application API to implement security design principles, notably least-privilege on client-side requests and reluctant-allocation to throttle password retries. They should also switch to server-side rendering to avoid credential leaks.

Our analysis results contributed to Ghost Box and the cyber community as a whole. Ghost Box stores sensitive client information including emails and addresses, and we identified a flaw which would otherwise compromise the confidentiality of this information. We also came up with recommendations for managing vulnerabilities in web applications that were easily adoptable by developers under tight deadlines, as in the case of Ghost Box and other startups, which contributed to security of the system. In doing so, we not only safeguarded Ghost Box's brand reputation and secured users' trust, but also fast tracked other startups' effort in making their web applications more secure, helping the Internet become a more secure place.

2 Analyzed System

2.1 Main Components

The following system components are identified from our discussions with Ghost Box and our independent investigation of their systems, it’s worth noting that the client wished us to conduct our analysis as outsiders without any insider knowledge, thus the information listed below are only from black-box testing perspective:

We do not have any information on this part of the system besides a list of public API endpoints Ghost Box provided to us.

- **IoT Storage Device**: These are physical devices placed across the operating area of Ghost Box. Users can reserve them via the web/mobile application, after which they can lock and unlock the device to secure and retrieve their physical assets.
- **Server**: The server is the backend of Ghost Box's system. It is responsible for communicating between IoT storage devices and client-side software, authenticating users, and storing user information. It communicates with web/mobile applications via API requests and the IoT storage device via a cellular

network. We do not have access to the backend code or have any knowledge on what technology is involved in the servers.

- **Client** (mobile and computer): These are devices with which clients access Ghost Box's web/mobile application. Clients authenticate to the application with an email and password pair, after which they can reserve an IoT device and lock and unlock it for storing and retrieving their physical assets.

2.2 Data Flow/High-Level Architecture

The diagram below illustrates these system components and the data flow between them.

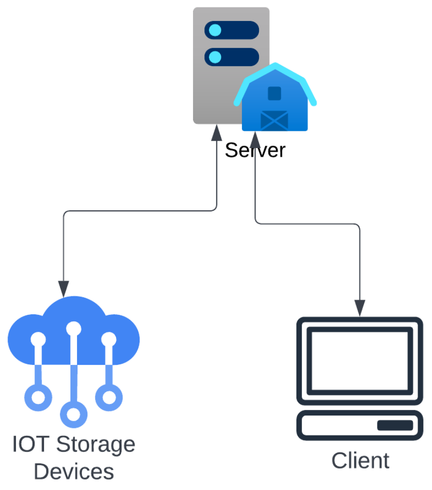


Fig. 2.0 High-Level Architecture Diagram of Ghost Box Systems

2.3 System Stakeholders and Interactions

The main stakeholders involved in this project include Ghost Box clients and employees. Ghost Box clients interact with the system using Ghost Box's web and mobile application, which enables them to view IoT storage device locations and availability, and reserve and use those devices. Before reserving an IoT device, clients must authenticate using an email and password scheme on the web/mobile application. However, they can view the device's locations on the web application without signing in. Clients also interact with the IoT device to store and retrieve their physical assets. We are not given details on how Ghost Box employees interact with the web/mobile application, but they must also physically interact with the IoT device for installation and maintenance.

3 Related Work

The "Security Analysis of the Democracy Live Online Voting System" [3] provided helpful instruction to understand the system's internal functions by exploring and documenting use cases from users' perspectives while not knowing the system's internal structure. This study also analyzed the website's client-server interactions. Their notable methods include verifying whether the user ID authorization tokens were secure in REST requests and injecting executable Javascript code attacks.

The difference between our study and this one would be that we would not conduct a complete system reverse-engineering to examine vulnerabilities. Nor would we need to analyze the different modes of operation.

The study “Black Box Internal Penetration Testing” [4] inspired us to use network security scanning tools and try intercepting data in transit. The study showed how admin credentials might be intercepted to compromise other user accounts further and that some logging messages in transit may reveal vulnerabilities in the backend code. However, we do not have access to our client’s backend, so we would not focus on identifying compromises on the underlying server.

The study “Verdi Agriculture” [5] had a similar approach to analyzing their system. The main difference was that in their analysis they had access to the internal code to reference for certain methods employed. For the methodologies used, both analyses had similar approaches for analyzing the front end client and the back end server. Additionally, we referenced their potential social hacking and phishing vectors and adapted them for our purposes.

4 Analysis Methodology

4.1 System Analysis

We scanned the website for vulnerabilities using online scanning tools, after which we attempted to exploit those vulnerabilities manually. Unlike most commercial black-box penetration testing procedures provided on the internet, we did not have access to paid vulnerability scanning tools. Thus, our analysis mainly focuses on human-driven testing combined with a secondary component of security scanning with available free tools. We also performed common website attacks and penetration testing to see if the website was negatively affected or yielded sensitive information and assets.

We also deployed credential hunting: We performed online stalking of Ghost Box employees for contact information such as emails and phone numbers, cross-referencing breached passwords online, and attempting impersonation by logging into their accounts. The above is illustrated in our attack tree (Appendix C). The system analysis was broken down into the following components:

1. Website Analysis via Scanning Tools
    a. ***HostedScan*** [6]. HostedScan discovers network, server, and website-related security risks. HostedScan implements scanning using NMAP [7], OpenVAS by Greenbone [8], and OWASP ZAP [9], which are all recommended tools by OWASP [10]. In particular, OpenVAS tests the following areas: gaining admin privilege, denying service, and obtaining information/assets.
    b. ***Maltego*** [11]**.** Maltego uncovers Ghost Box's network infrastructure.
    c. ***Nikto.*** We used Nikto which tests the backend server against thousands of commonly identified threats in the form of malicious programs and files. This also checks for outdated server software as well as version specific issues.
2. Credential Hunting
    a. ***Cyberstalking.*** We used Google search, social media, the company website, and other publicly available sources to learn about Ghost Box’s product, employees, and system. We then search for employees' personal contact information.

   b. ***Passworld Reuse Attack.*** All employee emails and phone numbers were put through Haveibeenpwned Transform for Maltego[11], [12] to see if they were involved in data leaks. A positive result would prompt us to search for the passwords from the source leak and dark web pastebin[13]. Any passwords found would be used to attempt login on the client website, as the likelihood of password reuse is high [5].

3. Human-driven tests and attacks
   a. ***Dictionary Login Attack.*** We used Hydra to check for password collisions between a list of the usernames we gained from the database in URL manipulation attacks and the RockYou password list [14]. We performed an dictionary login attack with a password dictionary and employee emails found from step 2.
   b. ***URL Manipulation.*** We used Postman to launch URL manipulation attacks by reusing authorization bearer tokens gained in the header of any successful get requests in Chrome inspector through logging into the web application using our own account and keep trying different common words used in URLs.
   c. ***SQL Injection Attacks.*** We first inspected potential input fields that could possibly communicate with the database: sign-in, sign-up, edit payment information and edit profile information fields. We injected SQL queries in the input fields and looked for SQL errors. If there is no SQL errors visible on the website, the website is safe from Error-Based SQL Injection. We then injected SQL queries inside the POST and PATCH request body to backend server, but if this didn't change the values in the database, it means the inputs are sanitized.

4.2 Iterative Analysis Steps

We follow the iterative steps listed below to gain information and inform our next steps in the analysis:

1. We scanned the app manually and with automatic scan tools to find useful information, such as user-facing API endpoints and third-party API tokens.
2. Using the gathered information, we tried to gain more information from the system. For example, we tried calling user-facing API endpoints with engineered parameters to see if they would return private data from the backend.
3. We then see if the information gathered could be indirectly used for malicious purposes. For example, a third-party API token could potentially be used for denial-of-service attacks. An adversary can write a script that calls the according to API service repeatedly with the key until all credit the system owner has is used up, and the service can no longer be used before the next payment cycle.
4. We find ways to place or alter data in the system. For example, we inject executable code to change the current user's name. In addition, we also attempted to alter the UI and insert new UI components that could be used to bait users into leaking sensitive information. For example, we tried to inject an alert box that asks for users' passwords to phish the password from them.
5. We repeat step 1 to 4 with different scanning procedures and tools in step 1 to find more attack vectors.

4.3 Ethical Considerations

The project was conducted under strict compliance with the Engineering Code of Ethics [15] and the EC-Council Code of Ethics for Penetration Testers [16]. All team members researched and understood related laws to avoid legal violations, emphasizing the Canadian Personal Information Protection and Electronic Documents Act [17] and the Criminal Code on Invasion of Privacy [18]. We adopted the most related part of the standards as mentioned earlier and our knowledge of ethical considerations to create a Ethical Guideline (Appendix E).

4.4 Risk Management

Our main way of preventing and managing risks is to strictly follow our Ethical Guideline. Our analysis of Ghost Box's system and technology was authorized by the company's CEO, the system owner and our point of contact. A test environment was provided to us by Ghost Box's leading product developers. If we were to engage in any behavior that could cause alterations or damages to the client's system, we must conduct them in this test environment.

4.4.1 Legal Risks

Marginal risk occurs when we communicate our findings to the system owner in unencrypted emails. Such risk is reduced by only communicating summarized information in emails and sending detailed and sensitive information in encrypted Word document files.

Violating any items listed in the project authorization agreement or accidentally conducting analysis outside the authorized system range could cause legal repercussions. Such risk is managed through careful design of analysis procedure and double checking with the client before carrying out any activity that we believe in having the potential of breaching any agreements.

4.4.2 Risks to Client & End Users

Any sensitive information gained through the analysis and appended to this report or presentation has the sensitive parts blurred. This evidence of security breaches was only used to demonstrate the existence of vulnerabilities in the system and nothing more.

The tools used during the analysis are either open source or recommended by highly respected industrial parties. The main tools we used for auto-scanning the vulnerabilities on the web application and the API were recommended by the Open Web Application Security Project (OWASP)[10]. A careful investigation of each tool's potential is carried out and documented before a tool is used.

As per requirement in the authorization agreement, the analysis project will not be made public until 12 months after the conclusion of the project to allow the client to resolve issues discovered during the project or implement any recommendation provided. Furthermore, the identity of the client is protected by using aliases in the report and presentation.

5 Results

5.1 Scanning Tools

5.1.1 ***HostedScan.*** The scanning report (Appendix F, Fig F1) indicates vulnerabilities, notably open ports and application error disclosure. These vulnerabilities leak internal information about the system.
5.1.2 ***Maltego.*** We obtained IP addresses, Domain Name Servers, and domains associated with Ghost Box's web application. See Appendix F, Fig F2 for the Maltego-generated graph.
5.1.3 ***Nikto.*** Nikto scanning on the backend identified certain server security headers not being set. (Appendix F)

5.2 Surface Web Credential Hunting

5.2.1 ***Cyberstalking.*** The employees' names were available on the company website. By using their names, we were able to collect personal information, such as email, phone number, and social media accounts.
5.2.2 ***Password Reuse Attack.*** Maltego's HavaIBeenPwned Transform**[12]** indicated some employee contact information found in 5.2.1 were involved in data breach incidences and gave us a description of each incidence. We do not include the HaveIBeenPwned Transform**[12]** graph to protect Ghost Box employees' personal information. We searched the contact information with positive results on pastebins**[13]** and for actual data from the data leaks for associated account passwords. We were however unable to uncover any associated password in either plaintext or encrypted form, so password reuse attack was not possible.

5.3 Attacks

5.3.1 ***Dictionary Login Attack.*** No password collision was found, but the fact that we were able to launch a brute-force password attack indicates that no rate-limiting feature is in place to stop a potential DOS attack or any related system disruptions caused by a sudden surge of requests.
5.3.2 ***URL Manipulation Attack.*** We were able to gain all users' names, emails, addresses, and SHA-512 encrypted passwords (Appendix G).
5.3.3 ***SQL Injection Attack.*** We weren't able to inject SQL query to the backend via input field or request body, and SQL error is invisible on the website.

5.4 Other Vulnerabilities

5.4.1 ***Account Creation Loophole.*** We found out that one user could register for multiple accounts using the same email because if the user adds a dot between any of the characters before @ in the email address, the account confirmation email will be sent to the user's email as normal, but the system would recognize this address as a new user email. And since each user gets 1 month free trial on the system's product, malicious users could keep adding dots and creating new accounts to use the system for free.

5.5 Summary of Assets Found

We were successful in obtaining the following Ghost Box assets:

- Login credentials and user personal information: all user's account emails, SHA-512 encrypted account passwords, full name, and addresses
- Ghost Box's Google Map API token
- Personal contact information of employees, including email and phone number

We were also able to register for multiple accounts with one email by adding dots in email addresses, which allows us to get multiple free trials on the system.

6 Discussion of the Results

6.1 Interpretation of the Results

Our conclusion is that Ghost Box's web application has not built security principles into their design, as is common for startup companies. The results and assets we acquired from our attack attempts include very sensitive personal information (email and address) of great significance, as well as information enough to cause serious system disruptions. Most of these vulnerabilities are easy to exploit, considering all adversaries need to do is manually scan through the website and try out some common URL routes.

Some of these problems are very easy to fix through adding authority level checks in the code, some of these problems are harder to fix and may require implementation of new features and restructure of code. These vulnerabilities were likely introduced due to developers rushing to make the website up and running without taking time to conduct careful design of security, or a lack of proper QA testing procedures.

6.2 Adversary Model

We constructed the following plausible adversary models against Ghost Box's system. They have the same capabilities: access to Ghost Box's public website and web application; the same funding level: minimal funding with a personal email and a device with Internet connection; and all of them are outsiders. An adversary with these attributes can acquire multiple free trials to avoid paying for Ghost Box's service by creating multiple new accounts by signing up with their email address with dots inserted (see 5.3.2). They can also acquire Ghost Box clients' personal information via a simple URL manipulation attack described in 5.a in section 4.1. Finally, they can use Ghost Box's Google Map API token for their own use on Ghost Box's credit.

6.3 Principles of Designing Secure Systems

We noted in our analysis that cybersecurity did not appear to be Ghost Box's priority, and as such it was not surprising to find the web application violates multiple design principles. It, however, does follow the principle of complete mediation since users are logged out after a period of inactivity and we did not find ways to bypass authentication.

Below are some of the principles that were violated and make the system vulnerable:

- **Reluctant-Allocation:** Log in is not rate limited and repeat attempt with the wrong credential does not trigger longer wait time or account lock, open to brute-force attack and DDoS attack.
- **Open-Design:** The code for the web application is not open source.

- **Least-Privilege:** Users are able to obtain other users' accounts and personal information and add assets to other users' accounts.
- **Time-Tested-Tools:** Implementing their own login mechanism instead of using established tools such as Google authentication.

7 Recommendations

We came up with the following recommendations for Ghost Box to improve the security of their system, and for any other startup companies to design their systems to be more secure.

- Frontend recommendations
  - hide API keys from the client side
  - Remove error logs in the frontend console to avoid leaking system error state information.
- Backend recommendations
  - Have clear API endpoint privilege documentations and verification logic.
  - Open ports should be examined and closed if necessary. For example, if a vulnerable version of the server is listening on port 80 and a secure server is listening on port 443, it would be safer to consider closing port 80.
  - Implement rate-limiting features to guard against DoS attacks, especially for login related endpoints. For example, consider locking the user out for a period of time after the user enters the wrong password for 5 times in a row.
  - Suppress response headers like “X-Powered-By” to better protect system framework information.
  - Configure “X-Frame-Options”, “X-XSS-Protection” and “X-Content-Type-Options” response headers to provide additional security features.
- General recommendations
  - Establish security protocols to regularly check for sensitive information leaks in the frontend.
  - Use server-side rendering.
  - Incorporate more exhaustive human-driven QA testing to make sure each API endpoint can only be called with appropriate authorization.
  - Educate and train company members to use strong passwords and avoid reusing passwords. Consider using a secure password manager for any company related accounts in the system.
  - Consider using Two Factor Authentication to make authentication more secure.

8 Conclusion

The rise in cyber crimes since the pandemic means it's ever more important for startups like Ghost Box to fortify their systems against attacks. Currently, the Ghost Box system contains serious vulnerabilities that need to be addressed immediately and ideally via system redesign.

We discovered that an adversary could:

- Conduct brute-force password attacks in a short timespan
- Obtain account and personal information of all users in the system

- Get multiple free trials using one email by registering new accounts with dots added to that email
- Obtain Ghost Box's Google Map API key and use the associated credits
- Add physical assets to other users' accounts

The vulnerabilities discovered and their associated risk should serve as a wake-up call for Ghost Box to put security a priority, and our recommendations should aid them in doing so to ensure their web application is robust and ready for the company's expansion. While a security analysis on their storage hardware was not in the scope of our analysis, we strongly recommend that they conduct such analysis to achieve defense in depth.

References

[1] "Cyber Risks: An Increased Threat During COVID-19," *Insurance Bureau of Canada*. http://www.ibc.ca/ns/business/risk-management/cyber-risk/an-increased-threat-during-covid-19 (accessed Dec. 04, 2022).

[2] "Penalties for Non-compliance | Canada | Global Data Privacy & Security Handbook | Baker McKenzie Resource Hub." https://resourcehub.bakermckenzie.com/en/resources/data-privacy-security/north-america/canada/topics/penalties-for-non-compliance (accessed Dec. 04, 2022).

[3] M. A. Specter and J. A. Halderman, "Security Analysis of the Democracy Live Online Voting System," p. 26.

[4] "Penetration Testing Services | SecureTriad." https://securetriad.io/ (accessed Dec. 04, 2022).

[5] A. Vazquez, E. Mir-Mohammadsadeghi, J. Hladyshevsky, S. Wong, and S. Zhou, "VerdiAgriculture.pdf," *Verdi Agriculture*. https://www.dropbox.com/s/76etdg0y8d0xrty/4_VerdiAgriculture.pdf?dl=0&unfurl=1 (accessed Dec. 04, 2022).

[6] HostedScan.com, "HostedScan.com," *HostedScan.com*. https://hostedscan.com (accessed Dec. 06, 2022).

[7] "Nmap: the Network Mapper - Free Security Scanner." https://nmap.org/ (accessed Dec. 04, 2022).

[8] "OpenVAS - Open Vulnerability Assessment Scanner." https://www.openvas.org/ (accessed Dec. 06, 2022).

[9] "OWASP ZAP | OWASP Foundation." https://owasp.org/www-project-zap/ (accessed Dec. 06, 2022).

[10] "Vulnerability Scanning Tools | OWASP Foundation." https://owasp.org/www-community/Vulnerability_Scanning_Tools (accessed Dec. 04, 2022).

[11] "What is Maltego?," *Maltego Support*. https://docs.maltego.com/support/solutions/articles/15000019166-what-is-maltego- (accessed Dec. 04, 2022).

[12] "Have I Been Pwned?" https://www.maltego.com/transform-hub/haveiben-pwned/ (accessed Dec. 04, 2022).

[13] "What is Pastebin and Why Do Hackers Love It?" http://www.echosec.net/blog/what-is-pastebin-and-why-do-hackers-love-it (accessed Dec. 04, 2022).

[14] OhMyBahGosh, "RockYou2021.txt WordList:" Dec. 04, 2022. Accessed: Dec. 04, 2022. [Online]. Available: https://github.com/ohmybahgosh/RockYou2021.txt

[15] "Code of Ethics." https://www.egbc.ca/Complaints-Discipline/Code-of-Ethics/Code-of-Ethics (accessed Dec. 04, 2022).

[16] "Code Of Ethics," *EC-Council*. https://www.eccouncil.org/code-of-ethics/ (accessed Dec. 04, 2022).

[17] L. S. Branch, "Consolidated federal laws of Canada, Personal Information Protection and Electronic Documents Act," Jun. 21, 2019. https://laws-lois.justice.gc.ca/eng/acts/P-8.6/page-7.html#docCont (accessed Dec. 04, 2022).

[18] L. S. Branch, “Consolidated federal laws of Canada, Criminal Code,” Oct. 26, 2022. https://laws-lois.justice.gc.ca/eng/acts/C-46/page-26.html#docCont (accessed Dec. 04, 2022).
[19] “X-Content-Type-Options - HTTP | MDN.” https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options (accessed Dec. 04, 2022).
[20] “X-XSS-Protection - HTTP | MDN.” https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection (accessed Dec. 04, 2022).

Appendix A Project Code of Conduct

1. Confidentiality and Integrity
    a. No identifying info of Ghost Box will be leaked or revealed during and after our security analysis.
2. Honesty and Trustworthy
    a. Prior to conducting the security analysis on Ghost Box, we obtained permission and informed all relevant personnel.
3. Contribute to the betterment of society
    a. The results of this analysis would be made public with the information presented in an easy to understand manner.
4. Do no harm
    a. The security analysis will be performed on a test server as opposed to a production server. This will minimize any unwanted side effects.
5. Abide by Engineering Code of Ethics
    a. Conduct an Ethical analysis

Appendix B Responsible Disclosure

B-1 Responsible Disclosure Timeline

Our findings will be shared throughout the course with the course teaching staff to obtain feedback during the project and for marking.

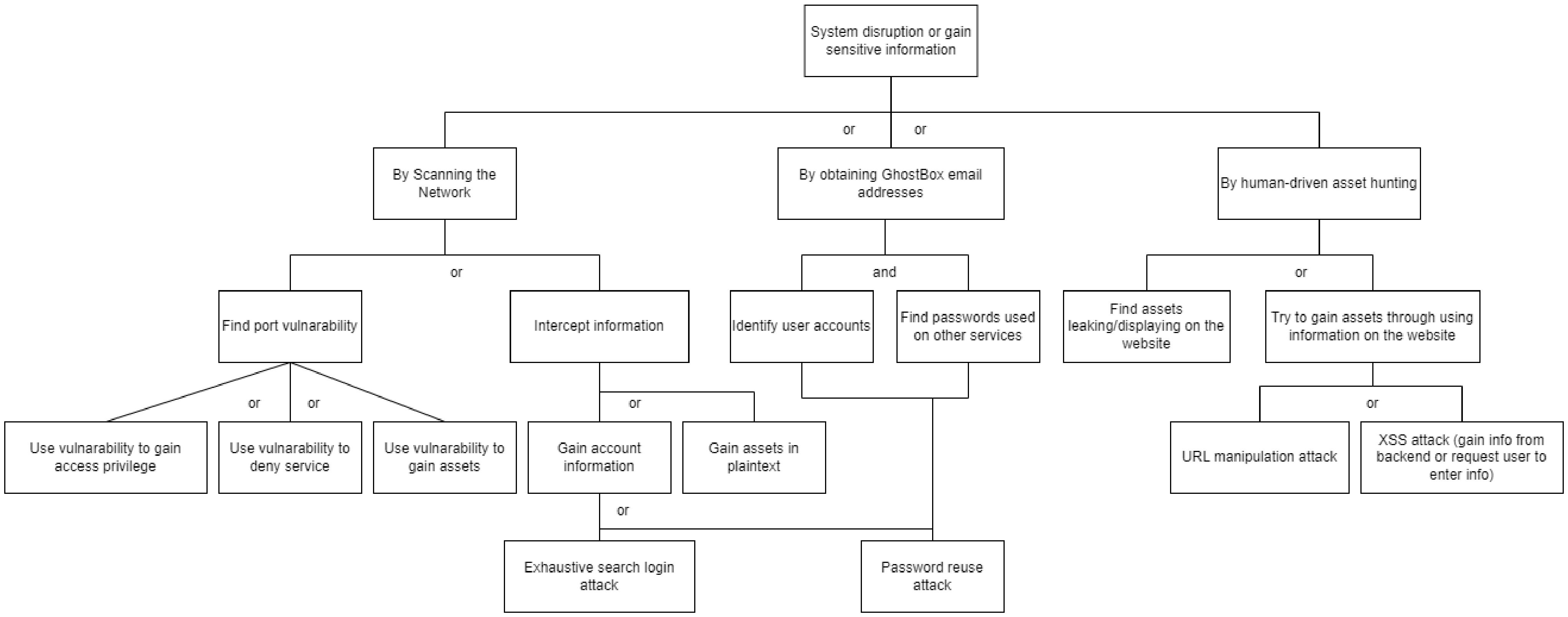
We will arrange a virtual meeting with Ghost Box system owners on December 8th, 2022, from 2-3 pm, after the final report is prepared. We will not use the system owner's name in the body or title of the report, and we will discuss our main findings with suggestions for improvements. After this meeting, we will email the technical contact from Ghost Box a summary of our analysis results and recommendations with the final report as an attachment. Any vulnerabilities that we identify as high-risk, however, will be immediately (within a day) communicated to Ghost Box with actionable recommendations so that they can avert the significant damage from attacks exploiting these vulnerabilities.

After the meeting, the course professor and the students will be available for consultations regarding the findings and follow-ups toward improving the security of the analyzed system. Ghost Box will also be able to inform us of any findings or information they wish to omit before we present our findings as part of the course's mini-conference. Peers in the course and IT security professionals from UBC and local commercial companies will be present at the mini-conference.

Finally, the report will be available to the general public 12 months after the final report's due date.

Appendix C Attack Tree

Fig C. Attack tree based on our analysis methodology


System disruption or gain sensitive information
or
or
By Scanning the Network
By obtaining GhostBox email addresses
By human-driven asset hunting
or
and
or
Find port vulnarability
Intercept information
Identify user accounts
Find passwords used on other services
Find assets leaking/displaying on the website
Try to gain assets through using information on the website
or
or
or
or
Use vulnerability to gain access privilege
Use vulnerability to deny service
Use vulnerability to gain assets
Gain account information
Gain assets in plaintext
URL manipulation attack
XSS attack (gain info from backend or request user to enter info)
or
Exhaustive search login attack
Password reuse attack

Appendix D Iterative Analysis Steps

6. We scanned the app manually and with automatic scan tools to find useful information, such as user-facing API endpoints and third-party API tokens.

7. Using the gathered information, we tried to gain more information from the system. For example, we tried calling modified versions of user-facing API endpoints to see if they would return private data from the backend.

8. We then see if the information gathered could be indirectly used for malicious purposes. For example, a third-party API token could potentially be used for denial-of-service attacks. An adversary can write a script that calls the according to API service repeatedly with the key until all credit the system owner has is used up, and the service can no longer be used before the next payment cycle.

9. We find ways to place or alter data in the system. For example, we inject executable code to change the current user's name. In addition, we also attempted to alter the UI and insert new UI components that could be used to bait users into leaking sensitive information. For example, we tried to inject an alert box that asks for users' passwords to phish the password from them.

10. We repeat step 1 to 4 with different scanning procedures and tools in step 1 to find more attack vectors.

Appendix E Ethics Guideline

1. Law and Safety

    1.1. Hold paramount the safety, health, and welfare of the public, including the protection of the environment and the promotion of health and safety in the workplace.

    1.2. Have regard for the common law and any applicable enactments, federal enactments, or enactments of another province.

    1.3. Have regard for applicable standards, policies, plans, and practices established by the government or Engineers and Geoscientists BC and the EC-Council.

    1.4. Report to applicable authorities, if we, on reasonable and probable grounds, believe that:

        1.4.1. The continued practice of a regulated practice by another person on the team or by the system owner might pose a risk of significant harm to the environment or to the health or safety of the public or a group of people; or

        1.4.2. Any illegal or unethical decision was made.

    1.5. Protect the intellectual properties of the client.

    1.6. Never knowingly use software or process that is obtained or retained either illegally or unethically.

    1.7. Use the property of a client or employer only in ways properly authorized and with the owner's knowledge and consent.

    1.8. Not to neither associate with malicious hackers or black hats nor engage in any malicious activities.

2. Documentation & Presentation

    2.1. Provide accurate information in respect of qualifications and experience.

    2.2. Provide professional opinions that distinguish between facts, assumptions, and opinions.

    2.3. Present clearly to the client the possible consequences if professional decisions or judgments are overruled or disregarded.

    2.4. Undertake work and documentation with due diligence and in accordance with any requirements provided by the course or the client.

    2.5. Provide the client with full disclosure of risk.

3. Project Management

3.1. Be upfront about our own technical limitations and capabilities with the client.

3.2. Avoid situations and circumstances in which there is a real or perceived conflict of interest and ensure conflicts of interest, including perceived conflicts of interest, are properly disclosed and necessary measures are taken so a conflict of interest does not bias decisions or recommendations.

3.3. Conduct themselves with fairness, courtesy, and good faith towards clients, colleagues, and others give credit where it is due, and accept, as well as give honest and fair professional comments.

3.4. Ensure ethical conduct and professional care at all times without prejudice.

4. Privacy and Security

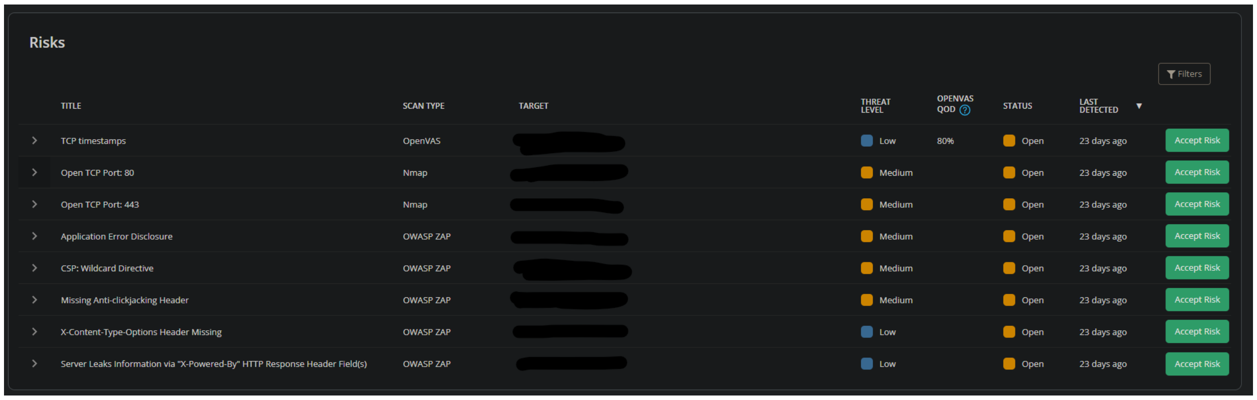
4.1. Keep and respect private or confidential information gained during the analysis.

4.2. Not collect, retain, give, sell, or transfer any personal information to a third party without the client's prior consent.

4.3. Not to purposefully compromise or allow the client organization's systems to be compromised. And if such a compromise is needed for analysis purposes, we should seek consent from the client and only conduct such activities within the authorized realm of the system.

4.4. Do not use any untrusted tools or software.

Appendix F Auto-Scanning Reports



Risks

Filters

	TITLE	SCAN TYPE	TARGET	THREAT LEVEL	OPENVAS QOD	STATUS	LAST DETECTED ▼	
>	TCP timestamps	OpenVAS	[redacted]	🟦 Low	80%	🟧 Open	23 days ago	Accept Risk
>	Open TCP Port: 80	Nmap	[redacted]	🟧 Medium		🟧 Open	23 days ago	Accept Risk
>	Open TCP Port: 443	Nmap	[redacted]	🟧 Medium		🟧 Open	23 days ago	Accept Risk
>	Application Error Disclosure	OWASP ZAP	[redacted]	🟧 Medium		🟧 Open	23 days ago	Accept Risk
>	CSP: Wildcard Directive	OWASP ZAP	[redacted]	🟧 Medium		🟧 Open	23 days ago	Accept Risk
>	Missing Anti-clickjacking Header	OWASP ZAP	[redacted]	🟧 Medium		🟧 Open	23 days ago	Accept Risk
>	X-Content-Type-Options Header Missing	OWASP ZAP	[redacted]	🟦 Low		🟧 Open	23 days ago	Accept Risk
>	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	OWASP ZAP	[redacted]	🟦 Low		🟧 Open	23 days ago	Accept Risk

Fig F1. HostedScan Reportxss

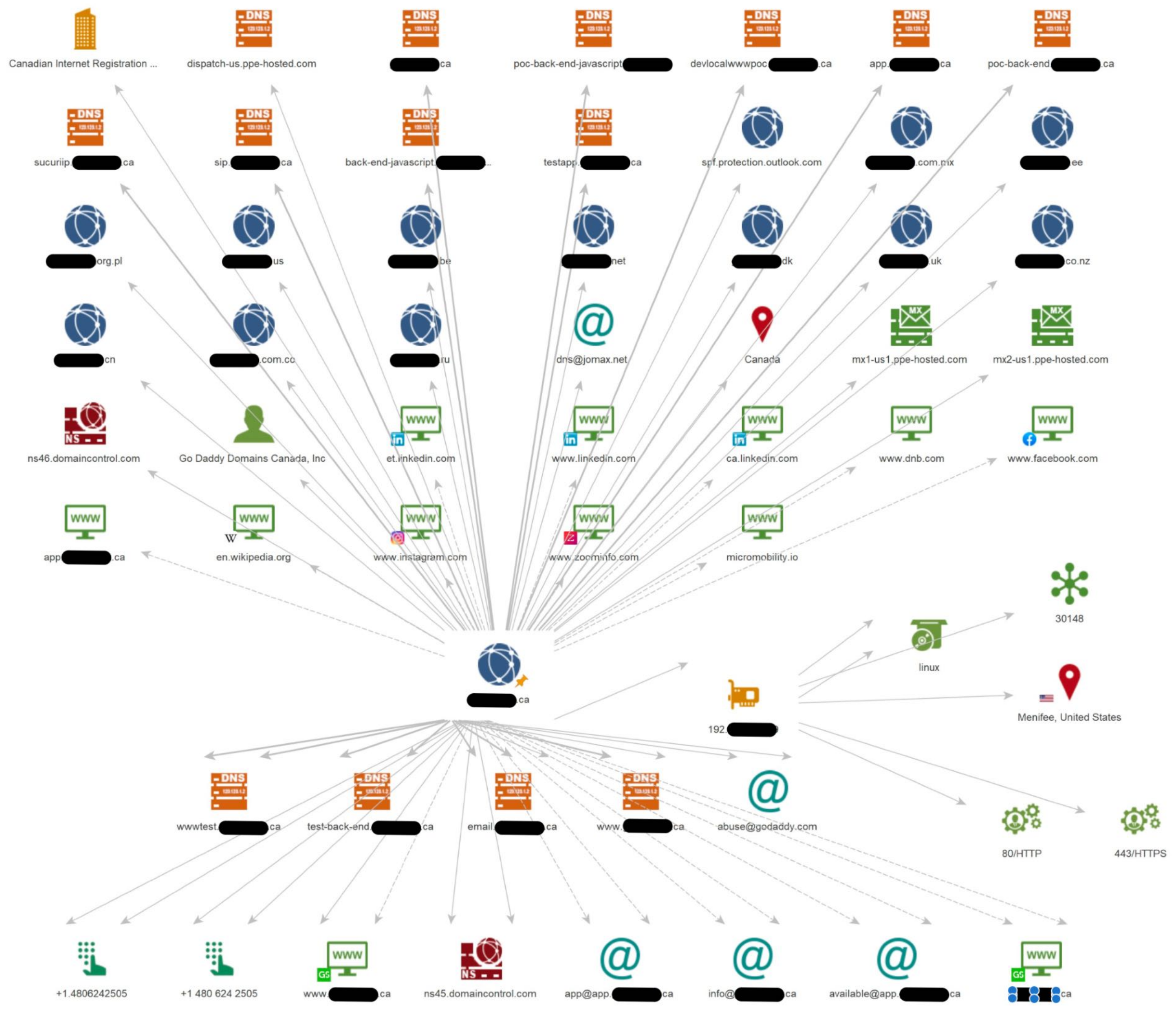


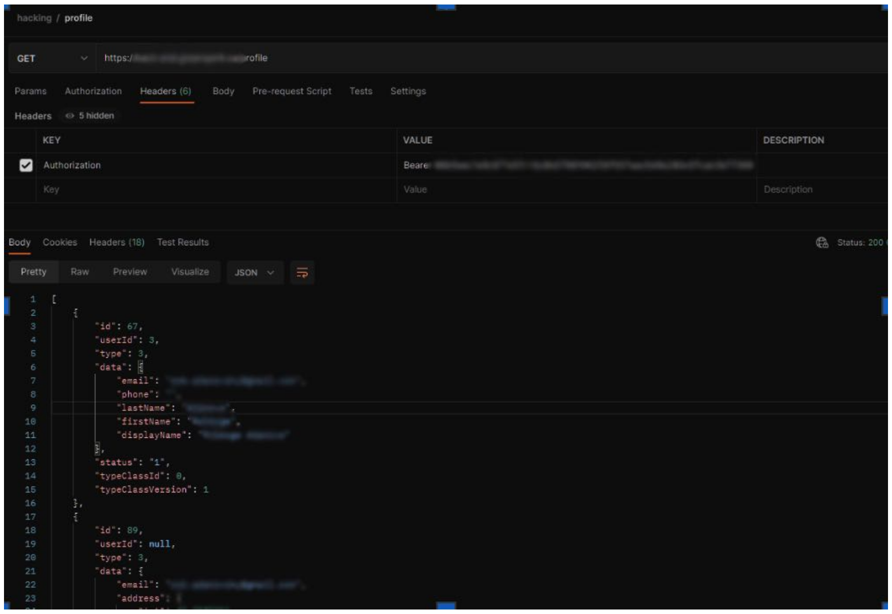
Fig F2. Maltego Analysis Graph

Nikto Report

```
% nikto -h <backend server DNS address,REDACTED> -C all -output HTML

- Nikto v2.1.6

---------------------------------------------------------------------------

+ Target IP:             <backend server IP address,REDACTED>

+ Target Hostname:    <backend server DNS address,REDACTED>

+ Target Port:        80

+ Start Time:         2022-11-27 22:01:27 (GMT-8)

---------------------------------------------------------------------------

+ Server: No banner retrieved

+ The anti-clickjacking X-Frame-Options header is not present.

+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect
against some forms of XSS

+ The X-Content-Type-Options header is not set. This could allow the user agent to render the
content of the site in a different fashion to the MIME type

+ Root page / redirects to: <backend server DNS address,REDACTED>

+ 26147 requests: 0 error(s) and 3 item(s) reported on remote host

+ End Time:           2022-11-27 22:14:02 (GMT-8) (755 seconds)

---------------------------------------------------------------------------

+ 1 host(s) tested
```

Report explained:
The X-Frame-Options header is not present. This header specifies the resource rendered within a frame or an iframe. This vulnerability can be exploited by clickjacking, where an attacker uses multiple transparent layers to trick users into clicking on a different button or link [6].
The X-Content-Type-Options header is not set. This can be exploited where attackers “sniff” the type contents of websites to help narrow their attack. This could also allow the user agent to potentially render non-executable MIME types into executable MIME types.[19]
The X-XSS-Protection header is not defined. Attackers can employ certain forms of XSS attacks on input fields. However, in modern browsers where inline JavaScript execution is disabled, this is mostly just an extra precaution.[20]

Appendix G Succeeded Attacks



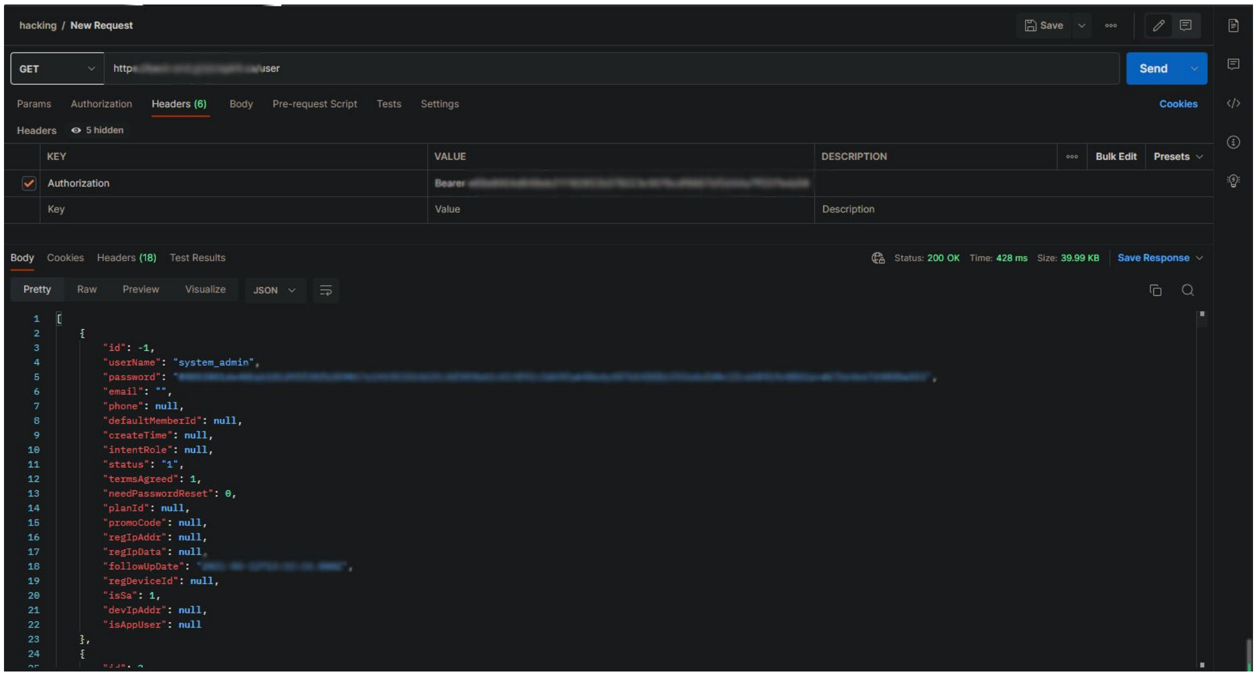
```
https:/[redacted]rofile
```

KEY	VALUE	DESCRIPTION
✓ Authorization	Beare[redacted]	
Key	Value	Description

Status: 200

```
[
    {
        "id": 67,
        "userId": 3,
        "type": 3,
        "data": {
            "email": [redacted],
            "phone": [redacted],
            "lastName": [redacted],
            "firstName": [redacted],
            "displayName": [redacted]
        },
        "status": "1",
        "typeClassId": 0,
        "typeClassVersion": 1
    },
    {
        "id": 89,
        "userId": null,
        "type": 3,
        "data": {
            "email": [redacted],
            "address": [redacted]
```

Fig G1 User Profile Gained



```
http[redacted]user
```

KEY	VALUE	DESCRIPTION
✓ Authorization	Bearer [redacted]	
Key	Value	Description

Status: 200 OK Time: 428 ms Size: 39.99 KB

```
[
    {
        "id": -1,
        "userName": "system_admin",
        "password": [redacted],
        "email": "",
        "phone": null,
        "defaultMemberId": null,
        "createTime": null,
        "intentRole": null,
        "status": "1",
        "termsAgreed": 1,
        "needPasswordReset": 0,
        "planId": null,
        "promoCode": null,
        "regIpAddr": null,
        "regIpData": null,
        "followUpDate": [redacted],
        "regDeviceId": null,
        "isSa": 1,
        "devIpAddr": null,
        "isAppUser": null
    },
    {
```

Fig G2 Encrypted User Password Gained