Security & Business Considerations Critical Analysis and Response

KRYPTON
BY STORMTROOPERS

CSCI5409: Advanced Cloud Computing

Group 40 (Team 3): Stormtroopers

Tanvi Pruthi – B00875949

Sidharth Mahant – B00899439

Mayank Sareen – B00899565

DALHOUSIE UNIVERSITY

Table of Contents

Application Architecture Data Security and Vulnerabilities

Krypton is a cryptocurrency trading broker website where users can register/log in and buy/sell Cryptocurrencies. Customers can manage their cryptocurrency portfolios and conduct transactions with a user-friendly website hosted and maintained on the AWS cloud platform.

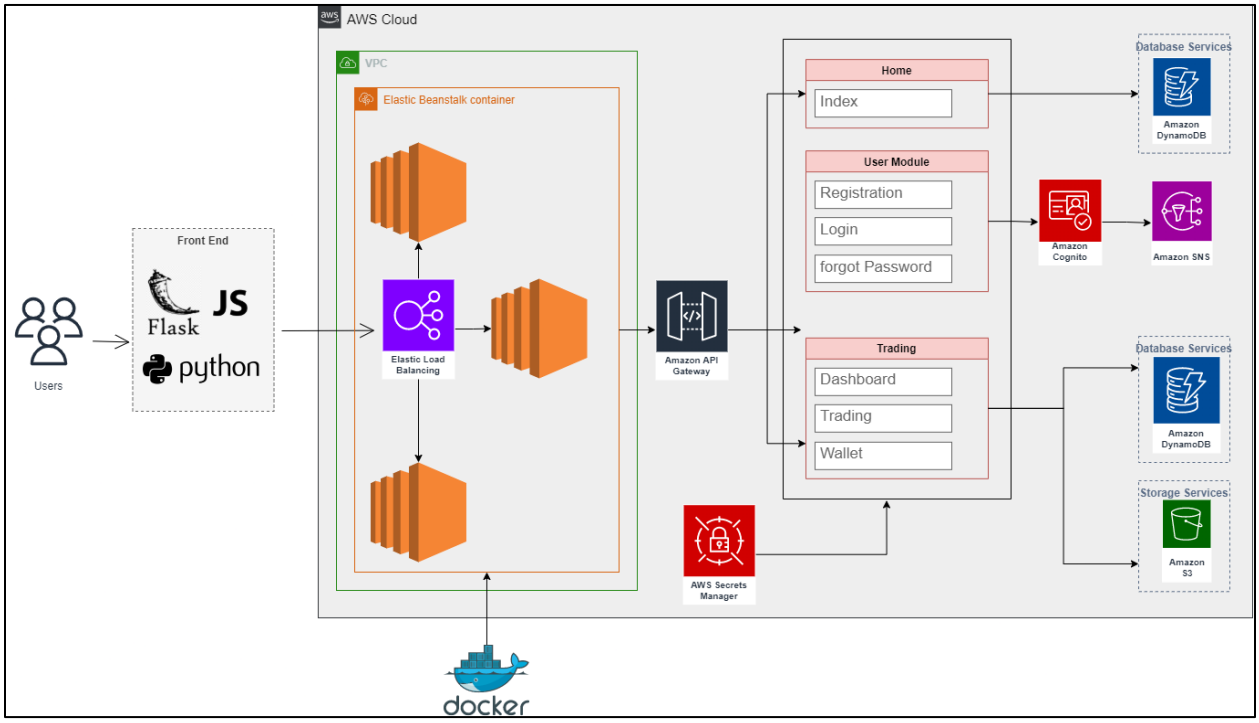

Figure 1: Architecture of Krypton

It is separated into two levels for this: frontend and backend. Our team is committed to ensuring data security across each of these layers. Therefore, we have taken the following steps:

- We have developed a multi-tier (N-tier) architecture, which divides the many application components into numerous tiers/layers based on their functions. Each layer is usually run on its system. That allows for compartmentalization and eliminates the possibility of a single point of failure. In the architectural diagram above, we will deploy an elastic load balancer to disperse traffic over numerous servers, removing the web server as a single point of failure. [2]

- For wallet services, we use Base58Check encoding to encrypt private keys in a wallet import format (WIF).

- To maintain security, AWS Cognito handles sensitive data such as registration information.

- Users must be signed in to access some areas of the website, such as the dashboard and trading. Otherwise, those URLs will redirect to the login page.

- We are not exposing sensitive information such as login, password, or any file link in the application's URL while re-directing from one page to another.

- We've incorporated proper logging: If Something goes wrong at some point. We will be able to respond as promptly as possible [3].

The web security vulnerabilities are prioritized depending upon exploitability, detectability, and impact. These are few vulnerabilities present in krypton's front-end and back-end. The severity of online security vulnerabilities is determined by their exploitability, detectability, and effect. The following are some of the vulnerabilities in Krypton's front-end and back-end:

1. Vulnerabilities at frontend layer

a) Cross-site request forgery (CSRF): Cross-site request forgery is a social engineering technique in which an attacker convinces an authenticated user to click on a link, image, or any URL to gain control of the user's session and manipulate or steal data from the online application. (Refer to figure 2) As a result, it seems the attack is carried out by a genuine user who sends malicious requests to the web application to exploit the website [1].

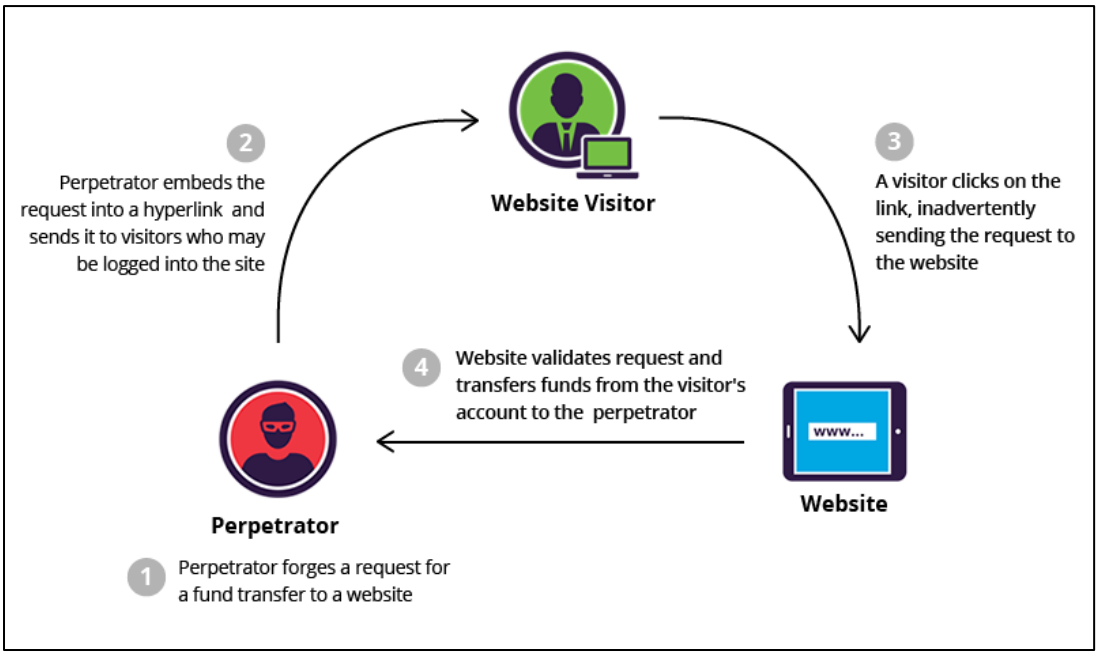

Figure2: Cross-site request forgery [4]

b) CSS injection: In this assault, developers employ a CSS file containing malicious import statements to carry out the attack. Several packages must be installed throughout the

development of a project to remove dependencies, and errors may arise at this time. The developer may download dependencies from an unknown website and run malicious code that exploits the website, such as CSS keylogger [1], which exposes credentials to the attacker.

c) XSS: Attackers can insert malicious code into the application's input fields, and the application will not identify the bad code and will show the identical input submitted, potentially exposing the application's sensitive data [1].

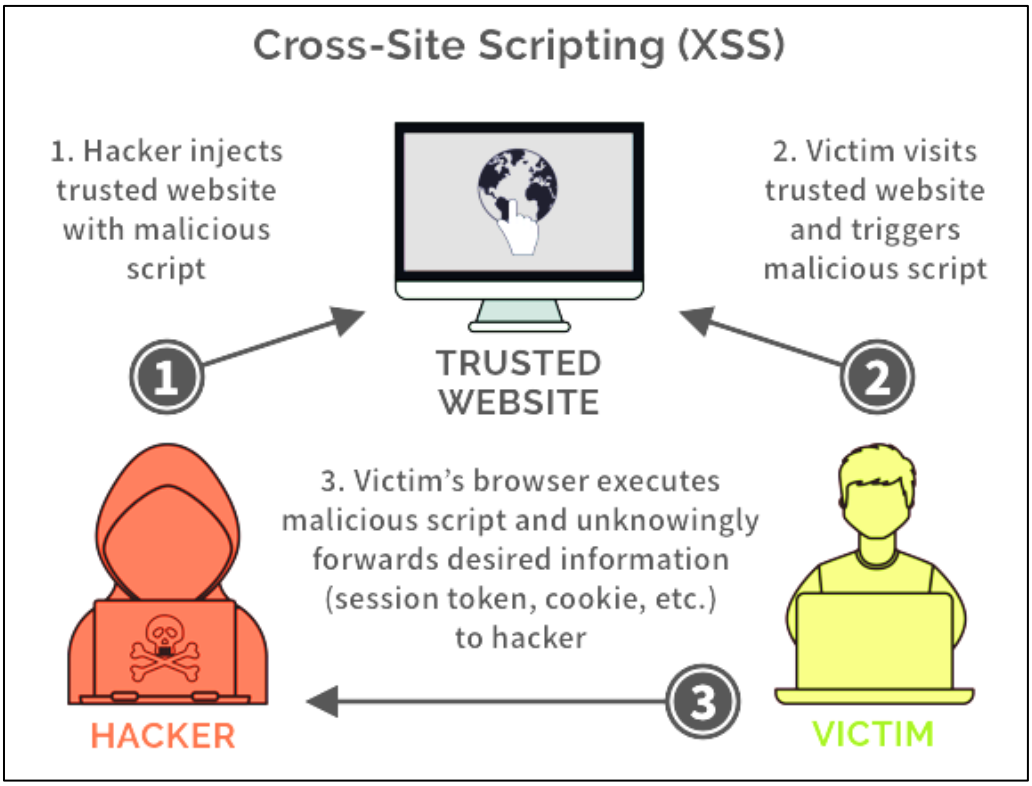

Figure 3: A high-level diagram for XSS [5]

2. Vulnerabilities at backend layer

a) SQL injection: SQL injection allows users to change the application's backend database. An attacker can run undesired instructions on the database, corrupting the data. (Refer to figure 4)

b) Broken Authentication and Session Management: The website keeps track of your session. Cookies containing sensitive information about the user, such as their login and password. When a user logs out, cookies are not declared invalid. Alternatively, if you close the browser suddenly, the sensitive data will remain in the browser. Cookies that are only used during a certain session. If the user is using a public computer, the cookies will remain on the device, exposing critical information to the attacker

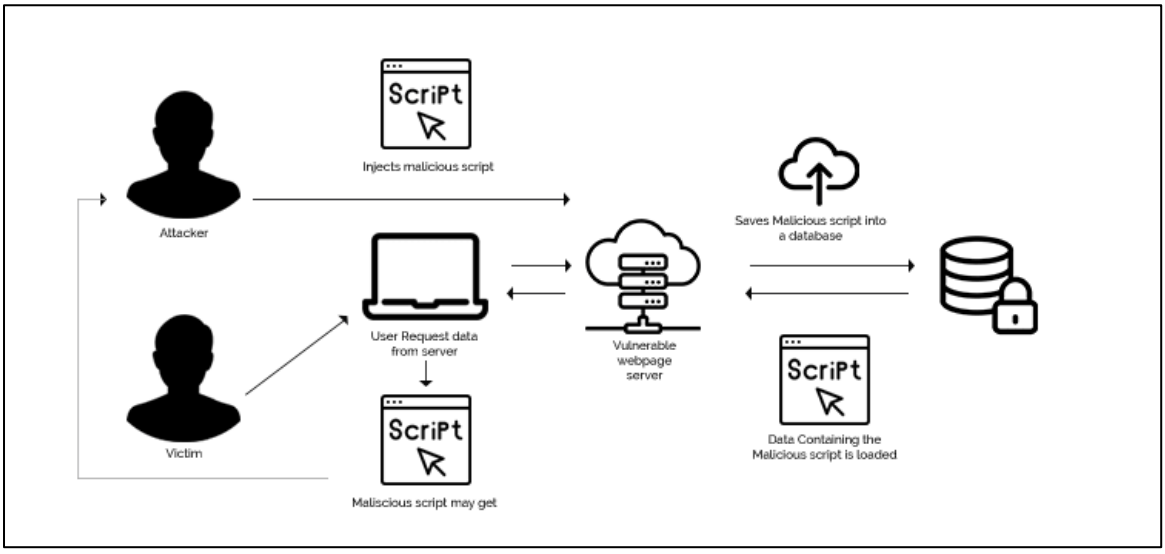

Figure 4: SQL injection diagram [9]

Security Mechanisms for Application Data Security

The above-mentioned vulnerabilities can be avoided in our project using the techniques described below.

1. Solutions at frontend layer

a) Cross-site request forgery (CSRF): Because the CSRF occurred because of cookie or session theft, we may simply add a random CSRF Token. The CSRF Token may be provided to each session with a random string and validated with each request after login, ensuring that no cross-site request is permitted. (The defense against CSRF is aligned with AWS Cognito as shown in figure 5)

b) CSS injection: CSS injections occur most frequently when we employ untrusted CSS URLs that have not been verified. We can apply the validated and self-hosted CSS libraries as the right approach.

c) XSS: One of the most common assaults is cross-site scripting or XSS. XSS attacks may be avoided by cleaning frontend input values. Additionally, using untrusted JavaScript libraries leads to vulnerabilities, thus only using trustworthy JavaScript libraries should be the preferred option. For creation of charts throughout, we are using chart.js which is a trusted JavaScript library.

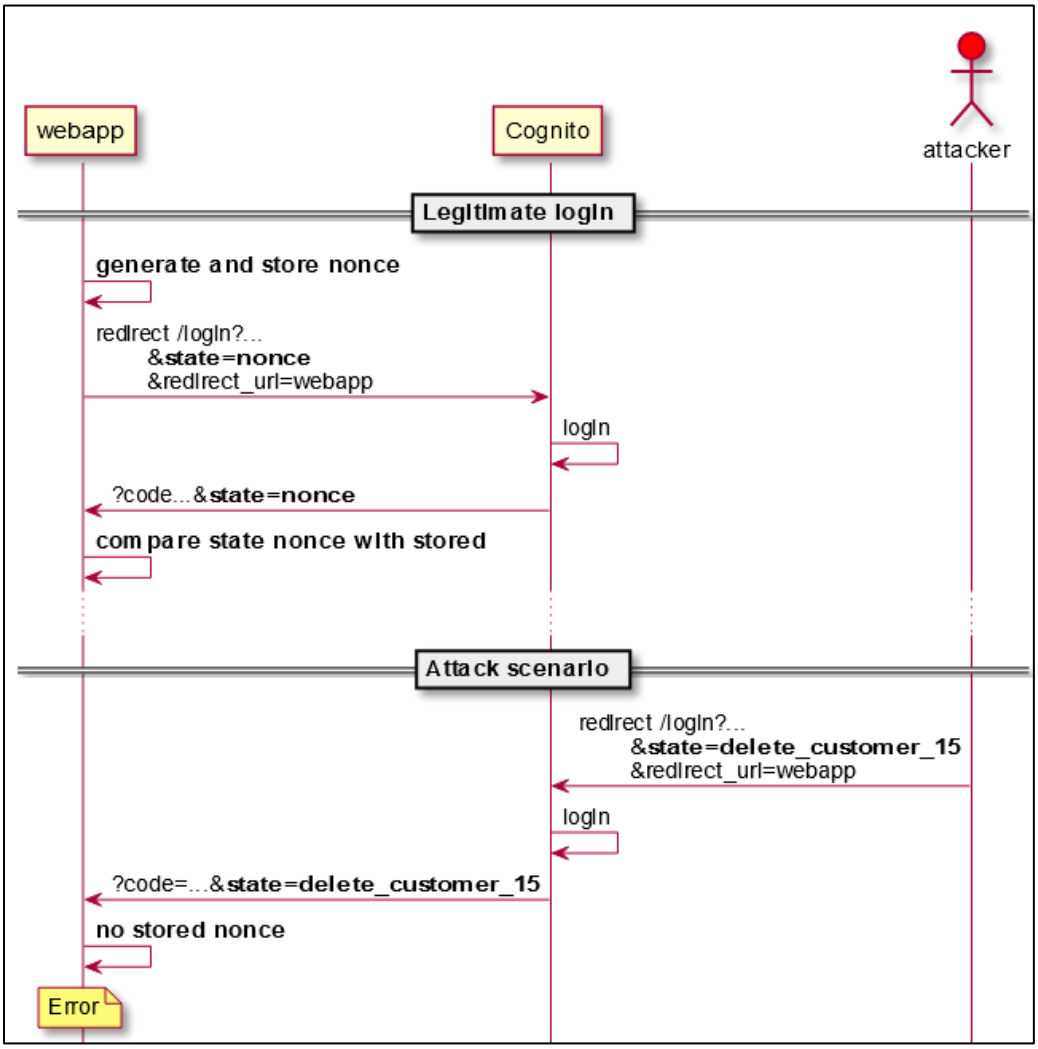

Figure 5: CSRF attack prevention with AWS Cognito [7]

2. Solutions at backend layer

a) SQL injection: SQL injection allows users to change the application's backend database. An attacker can run undesired instructions on the database, corrupting the data. SQL injection is a common back-end vulnerability exploit. In general, while retrieving data from the GET or POST methods in the back end, some sections of SQL are included, which alters the back-end query and causes data leaks. We can sanitize strings for SQL keywords and any query breaking symbols like double quotes ("), single quotes ('), equals (=), and others to avoid SQL injections.[6]

b) Broken Authentication and Session Management: This vulnerability occurs when users leave a web application with unmanaged sessions and cookies. The easiest way to avoid this type of vulnerability is to delete user sessions and cookies as soon as they exit the web application.

Resources Required for a Private Cloud Architecture and the Relatable Cost Estimation

It is costly to own a private cloud and host an application there. There are a variety of charges associated with owning a private cloud.

The total cost of ownership (TCO) for a small-scale website like our website, can be understood in figure 6. VM vRealize, Red Hat OpenStack and Charmed OpenStack are the private cloud providers in this scenario.

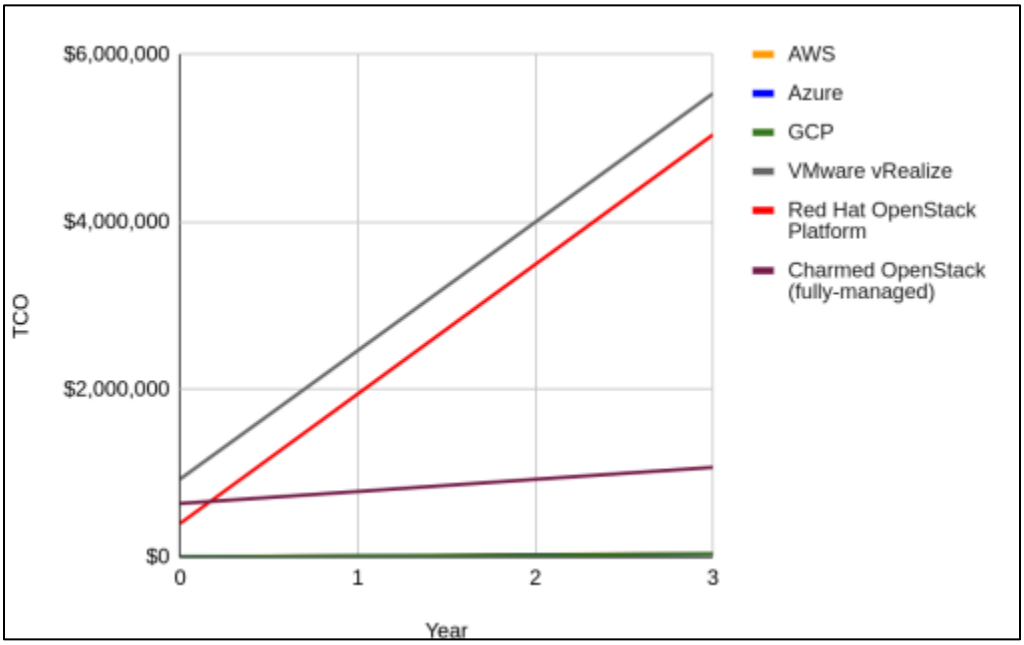

Figure 6: Total cost of ownership for some leading private and public cloud providers [8]

Below is a list of services and their related costs that an organization must purchase to recreate the Krypton architecture in a private cloud (Refer to figure 1):

- Server
- Storage
- Network
- Security

The associated costs would be as given in the table below:

Table 1: A preliminary estimate of the service expenses as well as the resources

Services to purchase	Resource option available	Cost per month (CAD)
Server	CPU, RAM	$1000 approx.

Storage	HDD and SSD	$200 approx.
Network	Bandwidth, IPs	$100 approx.
Security	Data protection, Network protection, Application protection	$500 approx.

Furthermore, the architecture necessitates a one-time investment in the project's setup and configuration. Finally, there will be costs associated with maintaining the total arrangement.

The Expected Costly Cloud Mechanisms

Amazon EC2, Amazon S3, and Amazon DynamoDB are the cloud services for our web application Krypton that may result in increased charges. These services need a lot of processing power and storage space. As a result, we need to monitor these services. The additional application services, such as Amazon Secret Manager, Amazon SNS, and Amazon Cognito, gets charged at a lower cost as and when utilized.

Of all the cloud services used in the Krypton, Amazon EC2 is the most expensive. Amazon EC2 operates 24 hours a day, and even if the EC2 instance is off, the storage cost for hosting the application on Amazon EC2's Elastic Block Store (EBS) gets charged.

References

[1] D.E. Team, “Frontend security vulnerabilities: Why you should care,” *Debricked* [Online]. Available: https://debricked.com/blog/frontend-security-vulnerabilities/ [Accessed: March 28, 2022].

[2] Tejas Dakve, “Attributes of secure web application architecture,” *Synopsis* [Online]. Available: https://www.synopsys.com/blogs/software-security/attributes-of-secure-web-application-architecture/[Accessed: March 29, 2022].

[3] Mathew, “Building Secure Applications: Top 10 Application Security Best Practices,” *Sqreen.com* [Online]. Available: https://blog.sqreen.com/best-practices-build-secure-applications/[Accessed: March 28, 2022].

[4] “Cross site request forgery (CSRF) attack,” *Imperva* [Online]. Available: https://www.imperva.com/learn/application-security/csrf-cross-site-request-forgery/ [Accessed: March 28, 2022].

[5] Brian Rutledge, “Cross-Site Scripting — Web-based Application Security, Part 3,” *Spanning* [Online]. Available: https://spanning.com/blog/cross-site-scripting-web-based-application-security-part-3/[Accessed: March 29, 2022].

[6] Lawrence Williams, “10 Most Common Web Security Vulnerabilities,” *Guru99* [Online]. Available: https://www.guru99.com/web-security-vulnerabilities.html[Accessed: March 28, 2022].

[7] Tamás Sallai, “How to secure the Cognito login flow with a state nonce and PKCE,” *Advancedweb,hu* [Online]. Available https://advancedweb.hu/how-to-secure-the-cognito-login-flow-with-a-state-nonce-and-pkce/ [Accessed: March 28, 2022].

[8] “Cloud pricing report 2021,” *Ubuntu* [Online]. Available: https://ubuntu.com/engage/cloud-pricing-report-2021[Accessed: March 29, 2022].

[9] “What is SQL Injection? | SQLi Attack Types and Preventions,” *Comodo.com* [Online]. Available: https://cwatch.comodo.com/blog/website-security/what-is-an-sql-injection-sqli/ [Accessed: March 28, 2022].