## Vulnerability argumentation

Group K(rypto Boyz)

# vLAN - A

This part of the network won't be accessible from the attackers, since they won’t know this part exists. It consists of the following:

* **Managment Machine and Server** for monitoring logs from the systems. It contains the management side/monitoring side of things. This is because we don’t want intruders to know that we are monitoring them so it’s outside the honeypot and we want to be able to manage all the pfsense routers without potentially leaking access to intruders.

# vLAN - C

The main part of the network. It consists of the following services:

* WebServer since the application that we’re hosting will be a web app. Thus they use some kind of a hosting service regardless if it’s on Premises or in the Cloud. Example websites:
  + [https://www.coinbase.com](https://www.coinbase.com/en-gb/partner/affiliate-incentives)
  + [https://robinhood.com/eu/en/about/crypto](https://robinhood.com/eu/en/about/crypto/?irclickid=3EjydcSJMxyPTNE2IdUp23P9UkHWycz9C3wERU0&irgwc=1&utm_source=impact&utm_campaign=170911&utm_content=Nerdwallet%2C%20Inc&utm_term=1017843&source=impact)
* Database/SQL Server for storing key information about users. Identifiable information (usernames, password, email’s, payment information, etc.), Wallet information and etc.
* Mailserver for handling communication between users, providing email services such as sending, receiving, and storing emails

A screenshot of a computer

Description automatically generated

# Services from the user perspective :

* Wallet System - When initially the user is making an account the system will ask him if he wants to create a new wallet or add an existing one.
* Tracking Crypto Prices System - When the user is logged in he will be allowed to find the most recent prices of xyz cryptocurrency

# Services with their corresponding vulnerability:

**SQL Server**

Vulnerability : SQLi

Implementation:

* Lack of Input Sanitization: Do not implement proper input validation or sanitization techniques for user inputs accepted by the SQL Server. Allow the user-supplied data to directly influence SQL queries without any filtering or validation.
* Dynamic Query Construction: Construct SQL queries dynamically by concatenating user inputs directly into SQL statements.

Ex:

**let query = “Select \* from ${userInput}”;**

* Error Reporting: Refrain from implementing comprehensive error handling mechanisms that might inadvertently disclose the existence of SQL injection vulnerabilities to potential attackers. Allow the SQL Server to generate detailed error messages that could potentially expose sensitive information regarding the database schema or SQL queries.
* Limited Access Controls: Provide the SQL Server user with broad permissions, allowing unrestricted access to sensitive data and database functionality.

**SSH Connection**

Vulnerability: Storing SSH Connection Credentials on the Web Server

Implementation:

* Credential Storage: Store SSH connection credentials, such as usernames and passwords, directly within configuration files or scripts on the web server. Avoid encryption or secure storage mechanisms to keep the credentials easily accessible.
* Access Control: Provide unrestricted access to the configuration file containing the SSH credentials, allowing potential attackers to read and extract the sensitive information without any authentication requirements.

**File Uploading (via web or mail)**

Vulnerability: Insecure File Upload

Implementation:

* Lack of File Type Validation: Implement minimal or no validation checks on file uploads, allowing users to upload files of any type to the web server without restriction.
* No File Size Limit: No limitations on the size of uploaded files, allowing users to upload excessively large files that may consume server resources or cause denial of service (DoS) conditions.
* Publicly Accessible Upload Directory: Store uploaded files in a publicly accessible directory on the web server's filesystem, allowing anyone to view, download, or execute uploaded files directly through the web server.