# **Step 1: Scoping Our Work**

## **Threat Model Information**

**Application Version:** 1.0

**Description:** The Gaming Console Store is a web application that serves as an online platform for customers to browse and purchase gaming consoles, accessories, and video games. Since this is the initial rollout of the website, its functionality is currently limited. While features for staff roles have been successfully implemented, customer functionality is not yet available. The application provides different functionalities based on the user's role. The primary users of the application are:

* **Admins**
* **Product Managers**
* **Inventory Managers**
* **Order Managers**
* **Customers** (Not yet functional)

Admins oversee user accounts, products, inventory, orders and returns, while also accessing activity logs and having all possible permissions. Product Managers handle products and category/subcategory management, ensuring information accuracy. Inventory Managers maintain stock levels, update warehouse details, and analyze inventory reports. Order Managers focus on processing orders, managing statuses, and handling returns. Customers can browse products, place orders, and manage their order history.

**Document Owners:** Sarjoun Radiyeh, Hadi El Nawfal, Mohamad Shaker

**Reviewer:** Khaled El Dassouki

## **Web Application External Dependencies**

|  |  |
| --- | --- |
| **ID** | **Description** |
| 1 | The GCS web server will ideally run on a Windows server configured with Flask for the backend. All dependencies and libraries will be regularly updated to ensure security. A self-signed certificate is used to enable HTTPS for secure communication during development and testing. The server will feature robust performance capabilities, scalable storage, and advanced security measures, including the latest operating system updates and application security patches, ensuring reliability and protection against vulnerabilities. |
| 2 | The website’s frontend will ideally run on a different Windows server, using React for the frontend. All dependencies and libraries will be regularly updated to ensure security. A self-signed certificate is used to enable HTTPS for secure communication during development and testing. The server will feature robust performance capabilities, scalable storage, and advanced security measures, including the latest operating system updates and application security patches, ensuring reliability and protection against vulnerabilities. |
| 3 | The GCS website implements role-based access controls (RBAC) via a separate Flask instance on a server. HTTPS is enabled using a self-signed certificate to simulate secure connections. Security practices such as regular library updates and strong authentication mechanisms are applied. The server will feature robust performance capabilities, scalable storage, and advanced security measures, including the latest operating system updates and application security patches, ensuring reliability and protection against vulnerabilities. |
| 4 | The database server will be MySQL and will run as a separate service. The MySQL instance is configured with strong passwords and limited access to prevent unauthorized interactions. Communication with the database occurs over the network. The server will feature robust performance capabilities, scalable storage, and advanced security measures, including the latest operating system updates and application security patches, ensuring reliability and protection against vulnerabilities. |
| 5 | The connection between the Flask backend, RBAC Flask instance, React frontend, and the MySQL database will occur over a private network. TLS is configured for all components using self-signed certificates to ensure secure communication during development and testing. |

## **Trust Levels**

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| 1 | Website Administrator | The admin can access all pages, perform user management, and oversee all activity. |
| 2 | Inventory Manager | The inventory manager can manage stock levels, update inventory, and track low-stock items. |
| 3 | Order Manager | The order manager can view and manage customer orders, generate invoices, and process returns. |
| 4 | Product Manager | The product manager can manage the product catalog, including adding, editing, and removing products. |
| 5 | Database Server Administrator | Has read and write access to the database used by the application. |
| 6 | Web Server User Process | The process or user that the web server executes code as and authenticates itself against the database server. |
| 7 | Database Read User | The database user account used to access the database for read-only purposes. |
| 8 | Database Read/Write User | The database user account used to access the database for both read and write purposes. |
| 9 | Anonymous Web User | A user who has connected to the GCS website but has not provided valid credentials. |
| 10 | User with Valid Login Credentials | A user who has connected to the GCS website and has logged in using valid login credentials. |
| 11 | User with Invalid Login Credentials | A user who has connected to the GCS website and is attempting to log in using invalid login credentials. |

## **Entry Points**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Trust Levels |
| 1 | HTTPS Port | The web application is only accessible via TLS. All pages are layered on this secure entry point. | (1)Website Administrator (2) Inventory Manager (3) Order Manager (4) Product Manager  (9) Anonymous Web User  (10) User with Valid Login Credentials  (11) User with Invalid Login Credentials |
| 1.1 | Login Page | Allows all roles to log in to access their respective features. | (1)Website Administrator (2) Inventory Manager (3) Order Manager (4) Product Manager  (9) Anonymous Web User  (10) User with Valid Login Credentials  (11) User with Invalid Login Credentials |
| 1.1.1 | Login Function | The login function accepts user supplied credentials and compares them with those in the database. | (1)Website Administrator  (2) Inventory Manager  (3) Order Manager  (4) Product Manager  (10) User with Valid Login Credentials  (11) User with Invalid Login Credentials |
| 1.2 | Dashboard | A central hub showing all metrics and key performance indicators (KPIs). | (1)Website Administrator |
| 1.3 | Warehouse Management Page | Handles warehouse-related operations such as creation, updates, and viewing warehouse data. | (1)Website Administrator (2) Inventory Manager |
| 1.3.1 | Warehouse Creation Modal | Allows for creating a new warehouse by providing necessary details like Manager ID and Location. | (1)Website Administrator  (2) Inventory Manager |
| 1.3.2 | Warehouse Update Modal | Enables updating existing warehouse details such as Manager ID and Location. | (1)Website Administrator (2) Inventory Manager |
| 1.4 | Inventory Management Page | Manages inventory, tracks stock levels, and provides alerts for low-stock items. | (1)Website Administrator (2) Inventory Manager |
| 1.4.1 | Inventory Update Modal | Allows updating the stock level of a product and assigning it to a warehouse. | (1)Website Administrator (2) Inventory Manager |
| 1.5 | Order Management Page | Displays and manages orders, including generating invoices and adding items to orders. | (1)Website Administrator (3) Order Manager |
| 1.5.1 | Invoice Generation Page | Provides a detailed summary of an order, including billing and pricing information. | (1)Website Administrator (3) Order Manager |
| 1.5.2 | Add Item to Order Modal | Allows adding new items to a specific order from a dropdown menu of available products. | (1)Website Administrator (3) Order Manager |
| 1.6 | Product Management Page | Manages the product catalog, including adding, editing, or removing products. | (1)Website Administrator (4) Product Manager |
| 1.6.1 | Add New Product Modal | Allows adding new products by entering product details like name, price, and category. | (1)Website Administrator (4) Product Manager |
| 1.6.2 | Remove Product Modal | Allows removing products by selecting the product from the table | (1)Website Administrator (4) Product Manager |
| 1.6.3 | Modify Product Modal | Allows modifying products by selecting the product from the table | (1)Website Administrator (4) Product Manager |
| 1.6.4 | Bulk Upload Products Modal | Allows uploading multiple products at once via a structured file (e.g., CSV). | (1)Website Administrator (4) Product Manager |
| 1.7 | Returns Management Page | Handles customer returns, including processing refunds and updating return statuses. | (1)Website Administrator (3) Order Manager |
| 1.7.1 | Modify Returns Modal | Allows modifying returns by selecting the return from the table | (1)Website Administrator  (3) Order Manager |
| 1.7.2 | Remove Returns Modal | Allows removing returns by selecting the return from the table | (1)Website Administrator  (3) Order Manager |

## **Exit Points**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Potential Threats** |
| 1.1 | Login API | Displays error messages when login fails. | **Account Harvesting** (via specific error messages), **SQL Injection** (if error traces are displayed). |
| 1.2 | Viewing API | Displays table summary to staff. | **Data Disclosure** (unauthorized access to table details). |
| 1.3 | Adding API | Adding data to tables. | **Data Tampering** (unauthorized addition of data before confirmation). |
| 1.4 | Removing API | Removing data from tables. | **Data Tampering** (unauthorized removal of data before confirmation). |
| 1.5 | Modifying API | Modifying data on tables. | **Data Tampering** (unauthorized manipulation of data before confirmation). |
| 1.6 | Uploading Files API | Processes CSV upload. | **Data Tampering** (unauthorized addition of data before confirmation). |
| 1.7 | Logout API | Ends the user's session. | **Session Hijacking** (if tokens are not invalidated or reused). |
| 1.8 | Password Reset API | Sends reset link or token to user email. | **Token Replay** (if token reuse is not mitigated). |
| 1.9 | Reporting API | Generates and downloads reports. | **Information Disclosure** (exposure of sensitive data in reports). |
| 1.10 | |  | | --- | | Admin Dashboard API |  |  | | --- | |  | | View admin-level details about table. | **Information Disclosure** (exposure of sensitive data in diagrams). |

## **Web Server Assets**

### **Website Users and Admin**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Asset | Description | Roles/Permissions Involved |
| 1.1 | User Login Details | The login credentials that users (customers/staff) use to log into the GCS Management Page (Different for each role) or Storefront (Customer Page). | (1) Website Administrator  (2) Inventory Manager  (3) Order Manager  (4) Product Manager  (5) Database Server Administrator  (6) Web Server User Process  (7) Database Read User  (8) Database Read/Write User  (10) User with Valid Login Credentials |
| 1.2 | Staff Login Details | The login credentials for staff and admins to manage the backend inventory, orders, and products, returns, warehouses (respectively). | (1) Website Administrator  (2) Inventory Manager  (3) Order Manager  (4) Product Manager  (5) Database Server Administrator  (6) Web Server User Process  (7) Database Read User  (8) Database Read/Write User |
| 1.3 | Personal Data | The GCS system stores personal user information like names, email addresses, and phone numbers. | (1) Website Administrator  (2) Inventory Manager  (3) Order Manager  (4) Product Manager  (5) Database Server Administrator  (6) Web Server User Process  (7) Database Read User  (8) Database Read/Write User  (10) User with Valid Login Credentials |

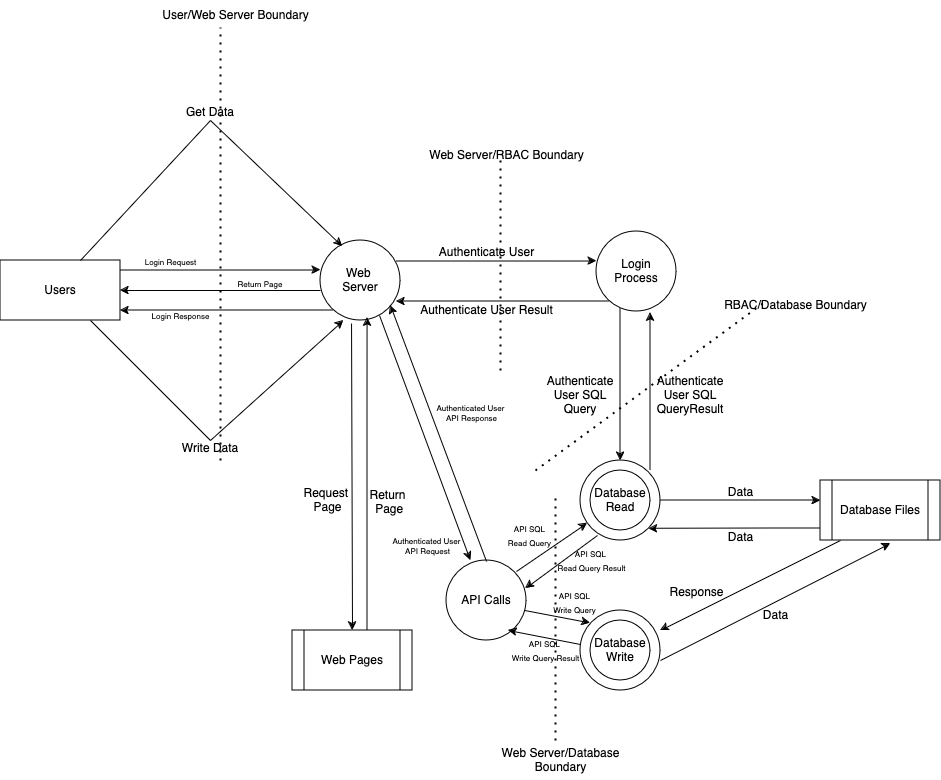
### **System**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Asset | Description | Roles/Permissions Involved |
| 2.1 | Availability of GCS Website | The GCS website should be accessible 24/7 for staff to manage and customers to shop. | (1) Website Administrator  (2) Inventory Manager  (3) Order Manager  (4) Product Manager  (5) Database Server Administrator |
| 2.2 | Ability to Execute Code as a Web Server User | This is the ability to execute source code on the web  server as a web server user. | (1) Website Administrator  (6) Web Server User Process |
| 2.3 | Ability to Execute SQL as a Database Read User | The ability to execute SQL select queries on the MySQL database to retrieve data. | (5) Database Server Administrator  (7) Database Read User  (8) Database Read/Write User |
| 2.4 | Ability to Execute SQL as a Database Read/Write User | This is the ability to execute SQL. Select, insert, and update queries on the database and thus have read and write access to any information stored within the GCS Database. | (5) Database Server Administrator  (8) Database Read/Write User |

### **Website**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Asset | Description | Roles/Permissions Involved | |
| 3.1 | Login Session | This is the login session of a user to the GCS website. This user could be an admin, inventory manager, order manager, product manager, or customer. | (1) Website Administrator  (2) Inventory Manager  (3) Order Manager  (4) Product Manager  (10) User with Valid Login Credentials |
| 3.2 | Access to the Database Server | Access to the database server allows you to administer the database, giving full access to database users and all data contained within the database. | (5) Database Server Administrator |
| 3.3 | Ability to Create Users | The ability to create users would allow an individual to create new users in the system. These could be admins, inventory managers, order managers, products managers, and customers. | (5) Database Server Administrator |
| 3.4 | Access to Audit Data | The audit data shows all auditable events that occurred within the GCS application by admins, inventory managers, order managers, products managers, and customers. | (5) Database Server Administrator |

## **Data Flow Diagram**



# **Step 2: Determining Threats**

## **STRIDE**

|  |  |  |
| --- | --- | --- |
| Threat | Description | Security Control |
| Spoofing | Threat action aimed at accessing and use of another user’s credentials, such as username and password. | Authentication |
| Tampering | Threat action intending to maliciously change or modify persistent data, such as records in a database, and the alteration of data in transit between two computers over an open network, such as the Internet. | Integrity |
| Repudiation | Threat action aimed at performing prohibited operations in a  system that lacks the ability to trace the operations. | Non-Repudiation |
| Information disclosure | Threat action intending to read a file that one was not granted access to, or to read data in transit. | Confidentiality |
| Denial of service | Threat action attempting to deny access to valid users, such as by making a web server temporarily unavailable or unusable. | Availability |
| Elevation of privilege | Threat action intending to gain privileged access to resources in order to gain unauthorized access to information or to compromise a system. | Authorization |

### **Spoofing**

**Threats:**

* Credential stuffing (using stolen credentials).
* Brute force login attacks.
* Session hijacking (stealing cookies).
* Cross-Site Request Forgery (CSRF).
* Broken access control (via improper implementation or flawed business logic).
* Man-in-the-middle (MITM) attacks.

**Countermeasures:**

* Implement Multi-Factor Authentication (MFA).
* Use secure cookies (HTTPOnly, Secure flags).
* Set up session timeouts and regenerate session IDs after login.
* Use CSRF tokens for form submissions.
* Enforce TLS/SSL for all communications; use HSTS headers.
* Apply strict access controls and enforce business logic validations.

### **Tampering**

**Threats:**

* Cross-Site Scripting (XSS) attacks.
* SQL Injection attacks.
* File tampering (malicious alterations in file storage).

**Countermeasures:**

* Use XSS prevention libraries (e.g., sanitizing inputs with OWASP).
* Implement parameterized SQL queries or ORM frameworks.
* Use file integrity monitoring and apply strict file permissions.

### **Repudiation**

**Threats:**

* Log tampering to erase traces of malicious activity.
* False claims from users (denying actions they performed).

**Countermeasures:**

* Use append-only log mechanisms and secure write-once media.
* Regularly back up logs.
* Log all critical actions and ensure logs are tamper-resistant.
* Combine logging with CSRF and session integrity measures.

### **Information Disclosure**

**Threats:**

* Unsecured APIs exposing sensitive data.
* SQL injection leaks.
* File system vulnerabilities allowing unauthorized access.
* Insufficient encryption for data in transit or at rest.

**Countermeasures:**

* Secure APIs with token-based authentication (e.g., JWT).
* Apply encryption for data at rest and in transit.
* Use parameterized queries to prevent SQL injections.
* Set up data masking for sensitive information in logs or error messages.
* Implement directory browsing restrictions.

### **Denial of Service (DoS)**

**Threats:**

* Distributed Denial of Service (DDoS) attacks flooding the server.
* Resource exhaustion from unoptimized queries or excessive API calls.

**Countermeasures:**

* Optimize database queries and APIs to handle high traffic efficiently.
* Implement account locking.

### **Elevation of Privilege**

**Threats:**

* Exploiting flaws in access control mechanisms.
* Privilege escalation by authorized users performing unauthorized actions.

**Countermeasures:**

* Implement granular role-based access controls (RBAC).
* Use privileged access monitoring and session auditing.
* Validate user actions against assigned privileges.
* Regularly review and update access control policies.
* Implement principle of least privilege.