Pochet CSD370

Owner: Julio Pochet Edmead
Reviewer:
Contributors: Darren Osier
Date Generated: Thu Jun 19 2025

Executive Summary

High level system description

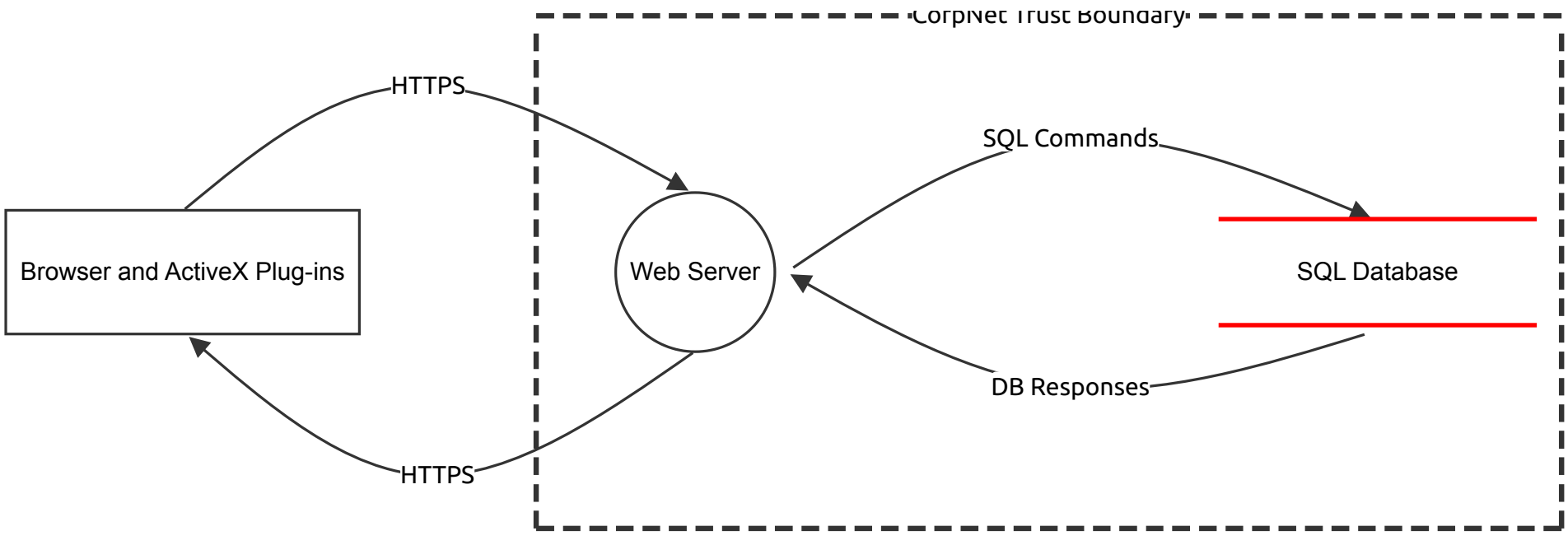
Not provided

Summary

Total Threats	3
Total Mitigated	1
Not Mitigated	2
Open / High Priority	1
Open / Medium Priority	1
Open / Low Priority	0
Open / Unknown Priority	0

CSD370

STRIDE model showing potential threats in a client-server web application involving a browser, web server, and SQL database.



CSD370

Web Server (Process)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations

HTTPS (Data Flow)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations

SQL Commands (Data Flow)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations

HTTPS (Data Flow)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations

DB Responses (Data Flow)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations

SQL Database (Store)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Unauthorized SQL Command Execution	Tampering	Low	Mitigated		An attacker could attempt to manipulate SQL commands sent from the Web Server to the SQL Database. If input validation and access control are not properly enforced, this could result in unauthorized data modification.	Implement strict role-based access controls (RBAC), input validation, and parameterized queries to prevent unauthorized data manipulation.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
3	Missing Logging of Database Transactions	Repudiation	Medium	Open		Lack of detailed logging for database operations could make it difficult to trace or verify actions taken by users or attackers. This could hinder forensic investigations or compliance audits.	Implement logging mechanisms that record user actions, query types, and timestamps to ensure traceability. Logs should be protected against tampering.
4	SQL Injection Vulnerability	Information disclosure	High	Open		User input passed to the SQL Database without sanitization could allow attackers to inject malicious SQL queries. This may lead to exposure of sensitive data such as user credentials or financial records.	Input sanitization and the use of prepared statements with bound parameters are recommended to prevent SQL injection attacks.

Browser and ActiveX Plug-ins (Actor)

Description:

Number	Title	Type	Priority	Status	Score	Description	Mitigations