Certified Secure Software Lifecycle Professional

Appendix C

Threat Modeling

In order to explain the threat modeling process, we will take a more practical approach of defining, modeling, and measuring the threats of a web store for a fictitious company named Zion, Inc., that has the following requirements:

Zion, Inc. is in the business of selling and renting Zii game consoles, games, and accessories. Lately, it has been losing market share to online competitors who are providing a better customer experience than Zion's brick and mortar establishments. Zion, Inc,. wants to secure its #1 market leader position for gaming products and services. The company plans to provide a secure, uninterrupted, and enhanced user experience to its existing and prospective customers. Zion, Inc., has contracted your organization to perform a threat modeling exercise for its online strategy. You are summoned to provide assistance and are given the following requirements:

- Customers should be able to search for products and place their orders using the web store or by calling the sales office.
- Prior to a customer's placing an order, a customer account needs to be created.
- Customer must pay with a credit card or debit card.
- Customers must be logged in before they are allowed to personalize their preferences.
- Customers should be able to write reviews of only the products they purchase.
- Sales agents are allowed to give discounts to customers.
- Administrators can modify and delete customer and product information.

Your request for pertinent documentation yields the following statements and requirements:

- The web store will need to be accessible from the Intranet as well as the Internet.
- The web store will need to be designed with a distributed architecture for scalability reasons.
- User will need authenticate to the web store with the user account credentials which in turn will authenticate to the backend database (deployed internally) via a web services interface.
- User account information and product information will need to be maintained in a relational database for improved transactional processing.
- Credit card processing will be outsourced to a third-party processor.
- User interactions with the web store will need to be tracked.
- The database will need to backed up periodically to a third-party location for disaster recovery (DR) purposes.
- ASP.Net using C# and the backend database can be either Oracle or Microsoft SQL Server.

We will start threat modeling Zion, Inc.'s web store by first defining the threat model. This includes identifying the assets and security objectives and creating an overview of the application.

Before we dive into the process of threat modeling, we must first identify the security objectives (vision).

Identify security objectives (vision)

For Zion, Inc.'s web store, from the requirements, we can glean the following objectives:

- ***Objective 1:*** Secure #1 market leader position for gaming products and services.
- ***Objective 2:*** Provide secure service to existing and prospective customers.
- ***Objective 3:*** Provide uninterrupted service to existing and prospective customers.
- ***Objective 4:*** Provide an enhanced user experience to existing and prospective customers.

Objective 1 and Objective 4 are both more business objectives than they are security objectives, so while they are noted, we don't really address them as part of the threat model. However, Objective 2 and Objective 3 are directly

related to security. To provide a secure service (Objective 2), the web store must take into account the confidentiality, integrity, and availability of data, ensure that authentication, authorization, and auditing are in place and that sessions, exceptions, and configurations are properly managed. To provide uninterrupted services (Objective 3), the web store will have high availability requirements defined in the needs statement and SLA, which will be assured through monitoring, load balancing, replication, disaster recovery, and business continuity and recoverable backups.

Although the loss of customer data and downtime can cause detrimental and irrecoverable damage to the brand name of Zion, Inc., for this threat model, we will focus primarily on tangible assets, which include customer data, product data, and the application and database servers.

Once the security objectives are identified and understood, we threat model the software. This includes the following phases with specific activities inside each phase as shown in *Figure C.1*.

1. Model Application Architecture
2. Identify Threats
3. Identify, Prioritize and Implement Controls
4. Document and Validate

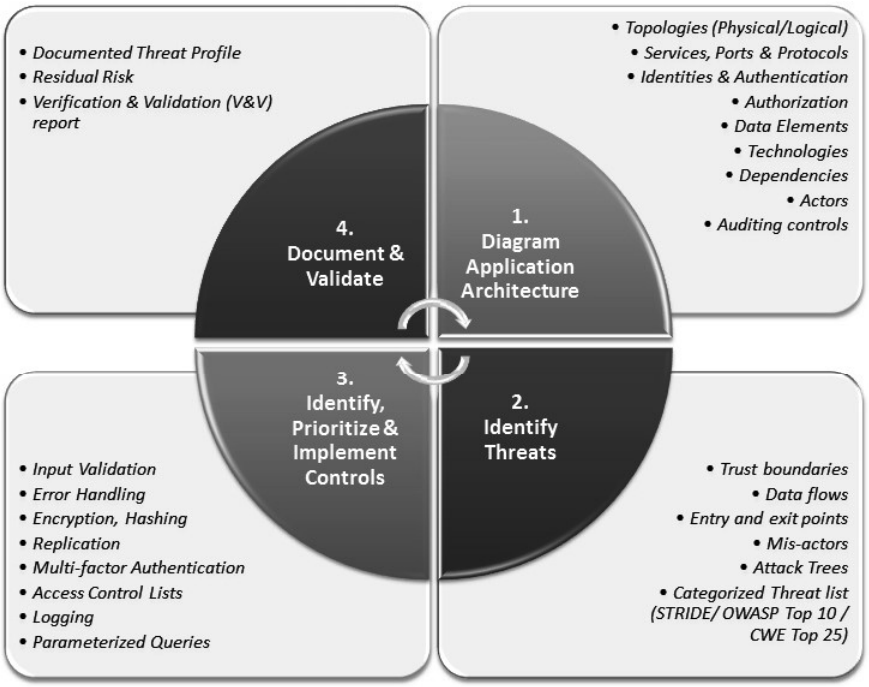

Figure C.1 – **Threat Modeling Process (Phases and Activities)**

Phase 1 – *Model Application Architecture*

This phase includes the diagramming of the application attributes and it includes the following activities.

Identify the physical topology.

Zion, Inc's web store will be deployed as an Internet-facing application in the DMZ with access for both internal and external users. Physically, the application will be entirely hosted on an application server hosted in the DMZ, with access to a database server that will be present internally as depicted in *Figure C.2*.

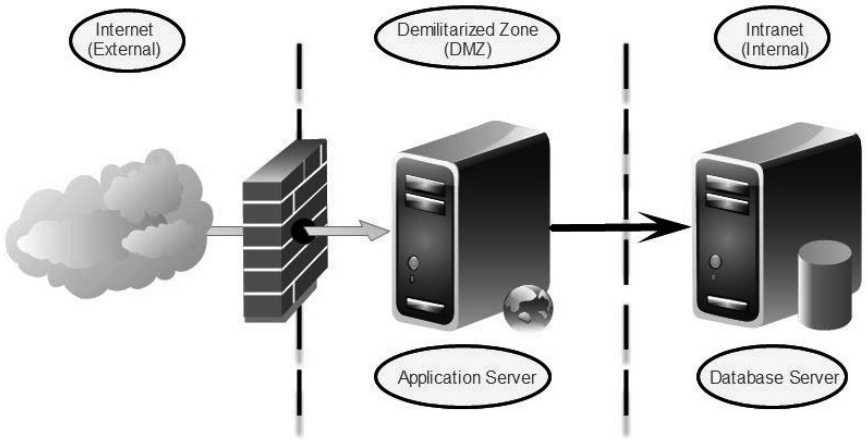

Figure C.2 – **Physical topology**

Identify the logical topology.

Zion, Inc's web store will be logically designed as a distributed client/server application with distinct presentation, business, data, and service tiers as depicted in *Figure C.3*. Clients will access the application using their web browsers on their desktops, laptops, and mobile devices.

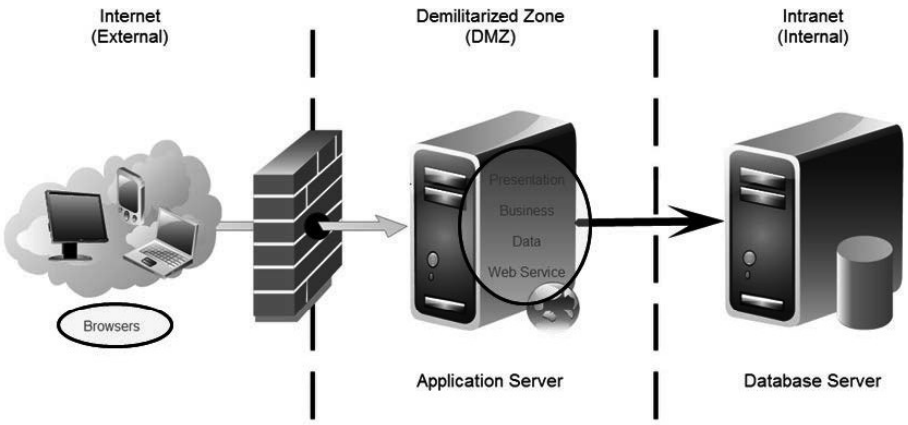

Figure C.3 – **Logical topology**

Determine components, services, protocols, and ports that need to be defined, developed, and configured for the application.

Users will connect to the web application over port 80 (using http) or over port 443 (using https). The web application will connect to the SQL server database over port 1433 (using TCP/IP). When the users use a secure transport channel protocol such as https over 443, the SSL certificate is also deemed a component and will need to be protected from spoofing threats. *Figure C.4* illustrates the components, services, protocols, and ports for Zion, Inc's web store.

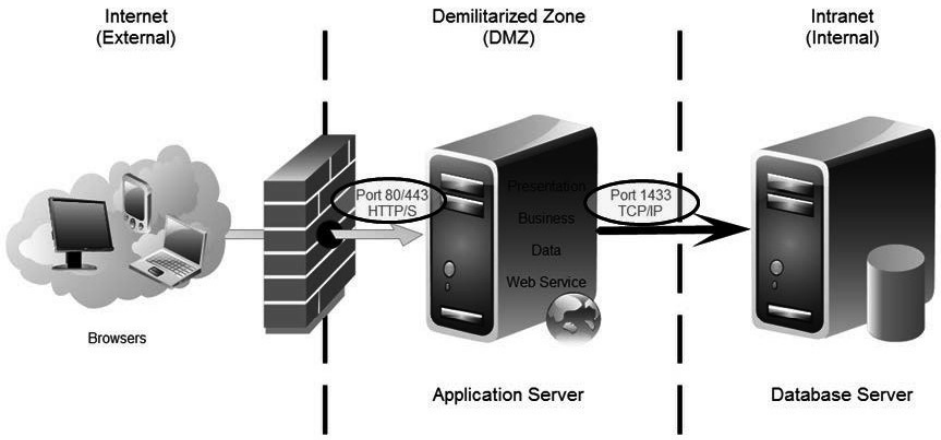

Figure C.4 – **Components, Services, Protocols and Ports**

Identify the identities that will be used in the application and determine how authentication will be designed in the application.

User will authenticate to the web application using forms authentication (user name and password) which in turn will authenticate to the SQL Server 2008 database (deployed internally) via a web services application using a web application identity as depicted in *Figure C.5*.

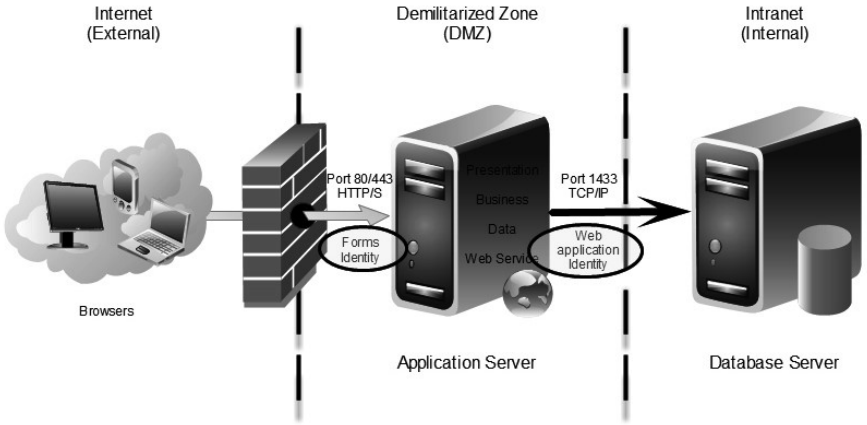

Figure C.5 – **Identities**

Identify human and non-human actors of the system.

The requirements state that:

- Customers should be able to search for products and place their orders using the web store or by calling the sales office.
- Sales agents are allowed to give discounts to customers
- Administrators can modify and delete customer and product information.
- The database will need to backed up periodically to a third-party location for disaster recovery (DR) purposes.

This helps us identify three human actors of the system: customers, sales agents and administrators as depicted in *Figure C.6*. Non-human actors (not shown in the figure) can include batch processes that back up data periodically to the third-party DR location.

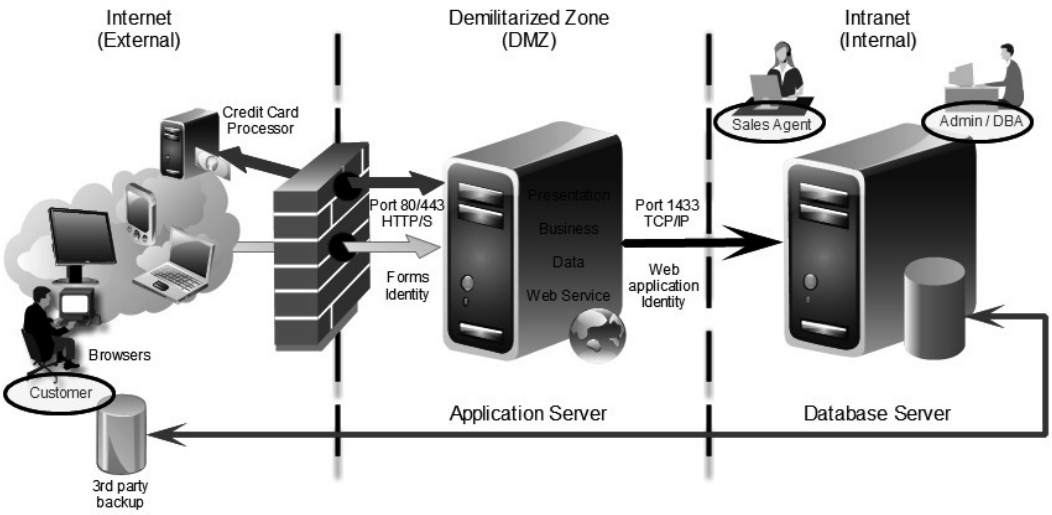

Figure C.6 – **Actors**

Identify data elements.

Some of Zion, Inc's web store data elements that need to be modeled for threats of disclosure, alteration, and destruction include customer information (account information, billing address, shipping address, etc.), product information (product data, catalog of items, product pricing, etc.), order information (data of order, bill of materials, shipping date, etc.), and credit card information (credit card number, verification code, expiration month and year, etc.). Since the web store will be processing credit card information, customer card data information will need to be protected according to the PCI DSS requirements.

Generate a data access control matrix.

The data access control matrix gives insight into the rights and privileges (Create (C), Read (R), Update (U) or Delete (D)) that the actors will have on the identified data elements as depicted in *Figure C.7*. The same should be performed for any service roles in the application.

		User Roles		
		Administrator	**Customer**	**Sales Agent**
Data	**Customer Data**	C, R, U, D	C, R, U, D	C, R, U
	Product Data	C, R, U, D	R	R, U
	Order Data		C, R, U, D	C, R, U, D
	Credit Card Data		C, R, U, D	R, U

Figure C.7 – **Data Access Control Matrix**

Identify the technologies that will be used in building the application.

Customer requirements stated that the web application will need to be in ASP.Net using C# while there was a choice of database technology between Oracle and Microsoft SQL Server. *Figure C.8* depicts the choosing of the Internet Information Server as the web server to support ASP.Net technology and the choosing of the SQL Server as the backend database. Whether ASP.Net will use the .Net 3.5 or .Net 4.0 framework, and if the SQL server will be the latest version or a prior version are important determinations to make at this point to leverage security features within these frameworks or products.

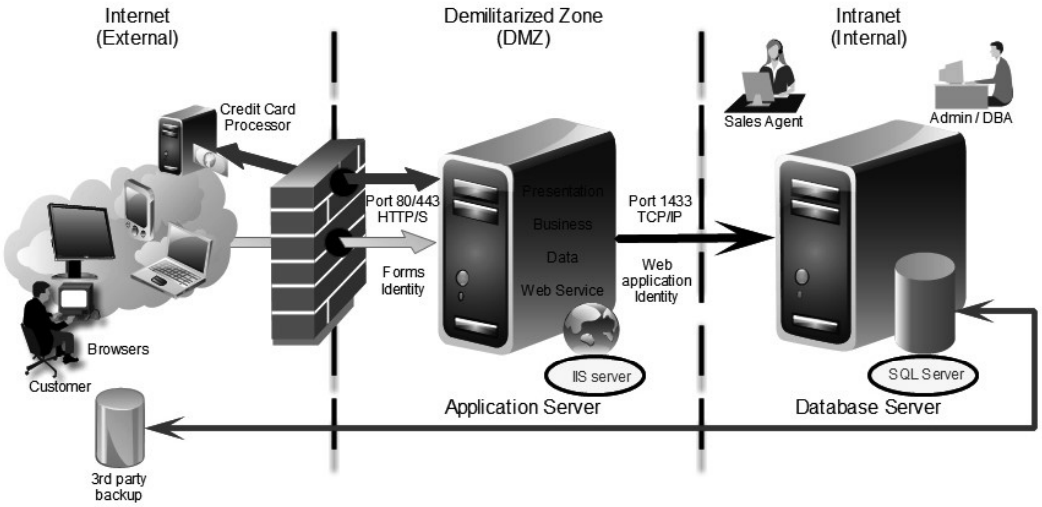

Figure C.8 – **Technologies**

Identify external dependencies.

External dependencies include the credit card processor and the third-party backup service provider as depicted in *Figure C.9*. The output of this activity is the architectural makeup of the application.

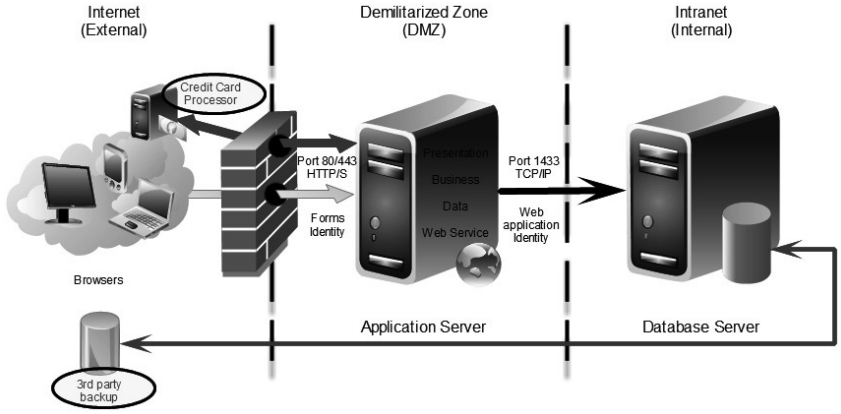

Figure C.9 – **Dependencies**

Phase 2 – *Identify Threats*

In order to identify potential applicable threats, we will conduct the following activities on the Zion, Inc. application.

Identify trust boundaries.

A trust boundary is the point at which the trust level or privilege changes. For the Zion, Inc's web store, trust boundaries exist between the external (Internet) and the DMZ and between the DMZ and the internal (Intranet) zones.

Identify entry points.

Entry points are those items that take in user input. Each entry point can be a potential threat source and so each must be explicitly identified and safeguarded. Entry points in a web application could include any page that takes in user input. Some of the entry points identified in the Zion, Inc's web store include the following:

- Port 80 / 443
- Logon Page
- User Preferences Page
- Product Admin Page

Identify exit points.

Exit points are those items that display information from within the system. Exit points also include processes that take data out of the system. Exit points can be the source of information leakage and need to be equally protected. Some of the exit points identified in the Zion, Inc's web store include the following:

- Product Catalog Page
- Search Results Page
- Credit card verification processes
- Backup processes

Identify data flows.

Data flow diagrams (DFDs) and sequence diagrams assist in understanding how the application will accept, process, and handle data as it is marshaled across different trust boundaries. Some of the data flows identified in Zion, Inc's web store include the following:

- Anonymous user browses product catalog page →Adds to Cart → Creates Account →Submits Order
- User Logs In → Updates Preferences → User Logs Out
- Administrator Logs In → Updates Product Information

Identify privileged functionality.

Code that allows elevation of privilege or the execution of privileged operations is identified. All administrator functions and critical business transactions are identified.

Introduce Mis-Actors

For Zion, Inc's threat model, both internal and external threat agents are introduced. Some applicable mis-actors include rogue DBA, uneducated users, external hacker, and any batch processes that make updates automatically.

Determine potential and applicable threats.

Although an attack-tree methodology could have been applied to determine potential and applicable threats, it was determined that using a categorized threat list would be more comprehensive. The STRIDE threat list was used for this exercise and the results tabulated as shown in *Table C.1*.

STRIDE List	*Identified Threats*
Spoofing	- Cookie Replay - Session Hijacking - CSRF
Tampering	- Cross Site Scripting (XSS) - SQL Injection
Repudiation	- Audit Log Deletion - Insecure Backup
Information Disclosure	- Eavesdropping Verbose Exception - Output Caching
Denial of Service	- Website Defacement
Elevation of Privilege	- Logic Flaw

Table C.1 – **Threat identification using STRIDE threat list**

Phase 3 – *Identify, Prioritize and Implement Controls*

The three common ways to rank threats are

- Delphi ranking
- Average ranking
- Probability x Impact (P x I) ranking

Both average ranking and P x I ranking methodologies to rank threats were followed and the results tabulated for Zion, Inc's. The Delphi ranking exercise was

Threat	D	R	E	A	DI	Average Rank (D+R+E+A+DI)/5
SQL Injection	3	3	2	3	2	2.6 (High)
XSS	3	3	3	3	3	3.0 (High)
Cookie Replay	3	2	2	1	2	2.0 (Medium)
Session Hijacking	2	2	2	1	3	2.0 (Medium)
CSRF	3	1	1	1	1	1.4 (Medium)
Verbose Exception	2	1	2	3	1	1.8 (Medium)
Brute Forcing	2	1	1	3	2	1.8 (Medium)
Eavesdropping	2	1	2	3	2	2.0 (Medium)
Insecure Backup	1	1	2	1	2	1.4 (Medium)
Audit Log Deletion	1	0	0	1	3	1.0 (Low)
Output Caching	3	3	2	3	3	2.8 (High)
Website Defacement	3	2	1	3	2	2.2 (High)
Logic Flaws	1	1	1	2	1	1.2 (Low)

Table C.2 – **Average ranking**

	Probability of Occurrence (P)			Business Impact (I)		P	I	Risk
Threat	R	E	DI	D	A	(R+E+DI)	(D + A)	P x I
SQL Injection	3	2	2	3	3	7	6	42
XSS	3	3	3	3	3	9	6	54
Cookie Replay	2	2	2	3	1	6	4	24
Session Hijacking	2	2	3	2	1	7	3	21
CSRF	1	1	1	3	1	3	4	12
Verbose Exception	1	2	1	2	3	4	5	20
Brute Forcing	1	1	2	2	3	4	5	20
Eavesdropping	1	2	2	2	3	5	5	25
Insecure Backup	1	2	2	1	1	5	2	10
Audit Log Deletion	0	0	3	1	1	3	2	06
Output Caching	3	2	3	3	3	8	6	48
Website Defacement	2	1	2	3	3	5	6	30
Logic Flaws	1	1	1	1	2	3	3	06

High: 41 to 60; Medium: 21 to 40; Low: 0 to 20

Table C.3 – **P x I ranking**

conducted but because of its non-scientific approach to risk, the findings were not deemed useful. *Table C.2* shows the threat ranks using the average ranking methodology. *Table C.3* shows the risk rank based on P x I ranking methodology.

After the threats are prioritized, your findings and the threat model are submitted to the organization. Based on this threat model, appropriate controls are identified for implementation to bring the security risk of Zion, Inc's web store within acceptable thresholds, as defined by the business. (See *Table C.4*)

Phase 4 – *Document and Validate*

Threats and controls can be documented diagrammatically or in textual format. Zion, Inc's threats are documented diagrammatically as depicted in *Figure C.10*. An example of textually documenting the SQL injection threat is tabulated in *Table C.5*.

Upon documentation of threats and controls, and the residual risk, we would validate the Zion, Inc. threat model to ensure that:

- The application architecture that is modeled (diagrammed) is accurate and contextually current (up-to-date).
- Threats are identified across each trust boundary and for data element.

Threat (P x I rank)	Controls
XSS (54)	Encode output; Validate request; Validate input; Disallow script tags; Disable active scripting
Output Caching (48)	Don't cache credentials; Complete mediation
SQL Injection (42)	Use parameterized queries; Validate input; Don't allow dynamic construction of SQL
Website Defacement (30)	Load balancing and DR; Disallow URL redirection
Eavesdropping (25)	Data encryption; Sniffers detection; Disallow rogue systems;
Cookie Replay (24)	Cookieless authentication; Encrypt cookies to avoid tampering
Session Hijacking (21)	Use random and non-sequential Session identifiers; Abandon sessions explicitly; Auto Log off on browser shutdown
Verbose Exception (20)	Use non-verbose error message; Trap, record and handle errors; Fail secure
Brute Forcing (20)	Don't allow weak passwords; Balance psychological acceptability with strong passwords
CSRF (12)	Use unique session token; Use referrer origin checks; Complete mediation
Insecure Backup (10)	Data encryption; SSL (transport) or IPSec (network) in-transit protection; ACLs
Audit Log Deletion (06)	Don't allow direct access to the database; Implement Access Triple security model; Separation of privilege
Logic Flaws (06)	Design reviews

Table C.4 – **Control Identification**

- Each threat has been explicitly considered and controls for mitigation, acceptance or avoidance have been identified and mapped to the threats they address.
- The residual risk of that threat is determined and formally accepted by the business owner, if the decision to accept the risk is made.

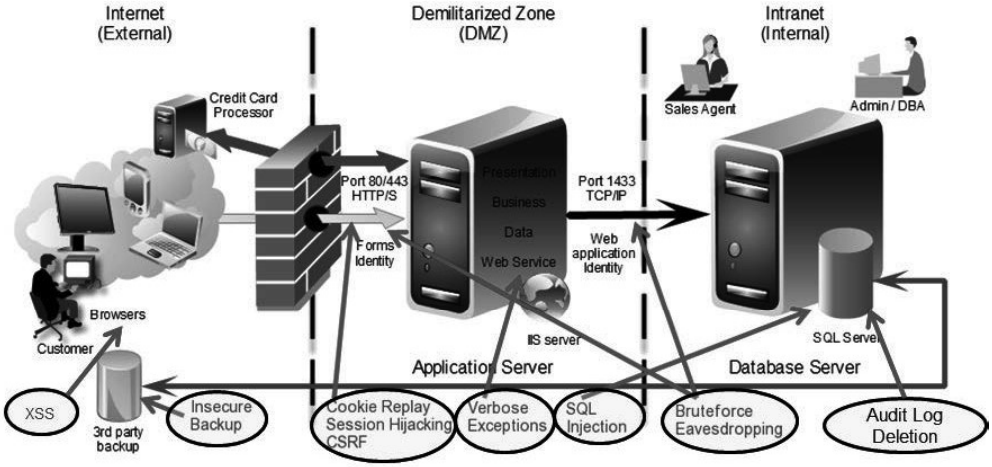

Figure C.10 – **Diagrammatically documents threats**

It is also important to revisit the threat model and revalidate it, should the scope and attributes of the Zion Inc's web store (application) change.

Threat Description	***Injection of SQL commands***
Threat targets	- Data access component - Backend database.
Attack techniques	- Attacker appends SQL commands to user name, which is used to form an SQL query.
Security Impact	- Information Disclosure. - Alteration. - Destruction (Drop table, procedures, delete data etc.). - Authentication bypass.
Safeguard controls to implement	- Use a regular expression to validate the user name. - Disallow dynamic construction of queries using user supplied input without validation. - Use parameterized queries.
Risk	- High

Table C.5 – **Textual documentation of a SQL Injection threat**