THREAT ID	THREAT TITLE	THREAT AGENT	THREAT DESCRIPTION	THREAT TARGET	ATTACK SURFACE	ATTACK TECHNIQUES	LIKELIHOOD	IMPACT	MITIGATION	CONTROL
TA1	SQL Injection (SQLi)	External - Lone Individual	As the name implies, this is when SQL code is injected into the system. From this the attacker is able to view and/or delete data from the database.	Customer information such as passwords, payment information and address.	Through data entry forms, such as the login page or directly through requests to the website.	There are several ways a SQLi can be performed but the simplest way is to add an apostrophe before the SQL code you would like to input and a SQL comment '--' after the code.	High: Although this has decreased in popularity it is still in third position in the OWASP top 10 [18]. As there is a wide availability of automated tools which perform SQLi [25][26] increases the likelihood.	High: There would be damage to the reputation of the business if information is leaked or service may be interrupted if the payment data or addresses is deleted.	Change the default configuration of MySQL [1] Ensure that hashed passwords are salted Employ the principle of least privilege [17] Move towards a framework which employs ORM (Object Relational Mapping), such as Django	Use stored procedures [17] Use parameterised statements [12][28] Regular expressions can be used to filter input [23]
TA2	Cross-Site Scripting (XSS)	External - Lone Individual	Cross-Site Scripting (XSS) is an injection attack that involves the injection of JavaScript code. The attacker's injected code is executed within the browser, providing a significant amount of access to the application. There are three main types of XSS: Persistent/Stored XSS, Non-Persistent/Reflected XSS, and DOM-Based XSS [7].	Customer information, a user's session and the site itself.	Through data entry forms, and HTTP requests when a Reflected XSS is performed.	Techniques include inserting script tags into data entry forms [12] and manipulating HTTP requests.	High: Simple to carry out, wide availability of scanner tools such as Burp Suite and XSStrike [20] and the existence of readily available scripts.	High: Ranges from site defacement to session hijacking [21].	Implementing a content security policy within the HTTP responses to disallow inline JavaScript [12][15]	Use entity encoding [12] for special characters To use PHP character escaping functions such as htmlspecialchars() [24] To switch to the React framework as this automatically escapes user input
TA3	Distributed Denial Of Service (DDos)	External - Criminal Organisation	A DDoS attack is when a server is overwhelmed by flooding it with traffic from multiple sources, making it unavailable to legitimate users.	To deny service to the site.	The web server hosting the application.	A botnet is most commonly used to deploy a DDos attack and the following some of the methods used to overwhelm the server are TCP SYN flood, ICMP flood and UDP flood [29].	Medium: As a large number of resources are needed to carry this out and access to a botnet is not specifically required although this increases the chances of success. However, the likelihood is increased due to the availability of DDos attack tools [3]	High: Lost revenue, as the site will be down, and damage to the reputation of the business.	Graceful degradation [16] to reduce the effect upon ongoing processes Incorporating IPS (Intrusion Prevention System) and IDS (Intrusion Detection System) [6]	Incorporating a load balancer to manage traffic By using a cloud provider, such as AWS and Azure, their in-built DDos protections will assist in mitigation
TA4	HTTP Parameter Pollution (HPP)	External - Lone Individual	Duplicated parameters in a HTTP request can result in unexpected behaviour of an application and may be exploited by attackers [10].	Business processes.	Endpoints of the system.	PHP and Apache use the last occurrence of a given parameter [4]. This means that additional parameters can be added to the end of the input of a parameter by using the '&' character to denote the end of an argument and the start of another parameter. For example, a user can add new arguments by inputting 'param1=arg1&param2=arg2&param1=maliciousarg'.	Low: As error messages from endpoints are less verbose [28] the vulnerability is harder to detect, and extensive knowledge of the system is required to pass the correct arguments. This is also compounded by the fact that this is not a widely known attack [2]	Medium: Allows malicious actors to discount their meal by changing the 'total' parameter to zero and therefore damage the business through lost revenue.	Encrypting the arguments, prevents a user from entering plain text and therefore is required to first discover the encryption technique used to send valid data	By blacklisting the '&' and '?' character for user input a malicious actor is unable to add more parameters The 'entity encoding' technique discussed in the 'control' section for threat TA2 can also apply here
TA5	HTML Injection	External - Lone Individual	A HTML injection attack involves an attacker injecting malicious HTML code into a web page [9], with the intention of manipulating its content to trick users into revealing sensitive information or taking malicious actions.	Customer information.	Through data entry forms, and HTTP requests.	Typically, a similar technique to XSS is used for HTML injection although the difference lies within how the injected code is typically used. A distinguishing factor of this attack over XSS is how the end user is targeted. A malicious actor will most likely perform content spoofing of the login page.	Low: Although not difficult to perform, it is more difficult to deceive users and therefore other attacks may be chosen instead	Medium: Tricked users will reveal their information to an attacker	By educating both customers and staff about the possibility of content spoofing this will decrease those 'tricked' by a malicious actor	As with the threats (TA1, TA2 and TA4) input sanitisation techniques, such as blacklisting, entity encoding and escaping characters are the best defence against this attack
TA6	Cross-Site Request Forgery (XSRF)	External - Criminal Organisation/Lone Individual	XSRF is a social engineering technique that utilises the authenticated user's credentials to perform unauthorised actions on a web application. This attack can be initiated by visiting a malicious website or through email-based links.	Possible user actions.	Endpoints of the system.	The attack is typically executed through social engineering methods, which deceive users into clicking on a malicious link, often in the form of a GET request. These links can be delivered through emails [19] or embedded images [22].	Medium: Although technically simple, this requires social engineering tactics and knowledge of the system.	High: Any action a valid user performs through HTTP requests, the attacker can as well	Requiring user interaction, such as through a CAPTCHA [27] Utilise POST requests instead of GET requests [12] By using hidden fields within a html form, it is possible to force a user to use your form [22]	Including the session ID directly in the request, therefore an attacker requires the session ID of the user to form the malicious request [8] Using the shared token technique by implementing a hidden token field, which is generated by the session ID [8] A proxy between the web server and the target application, which holds a token table which maps tokens to session IDs can be used to implement the above [8]

THREAT ID	SPOOFING IDENTITY	TAMPERING	REPUDIATION	INFORMATION DISCLOSURE	DENIAL OF SERVICE	ELEVATION OF PRIVILAGE
TA1	☑	☑	☑	☑	☑	☑
TA2	☑	☐	☑	☐	☐	☑
TA3	☐	☐	☐	☐	☑	☐
TA4	☐	☑	☐	☑	☐	☐
TA5	☐	☑	☐	☑	☐	☐
TA6	☑	☑	☐	☑	☐	☐

THREAT ID	Damage	Reproducibility	Exploitability	Affected users	Discoverability	Total	Rating
TA1	3	3	2	3	3	14	High
TA2	3	2	2	3	3	13	High
TA3	3	1	1	3	2	10	Medium
TA4	2	3	1	1	0	7	Medium
TA5	2	2	2	1	3	10	Medium
TA6	2	3	2	1	1	9	Medium

THREAT ID	ASSET	ATTACK	ATTACK SURFACE	ATTACK GOAL	IMPACT	CONTRROL	MITIGATION
TA1	A1, A2, A4	SQLi	Login page	Login as another user, delete data or reveal passwords	High: Passwords revealed or logged in as admin	Stored Procedures Parameterised Statements Filter input	Change default configurations Salt hashed passwords Switch to a ORM framework
TA2	A3, A4	XSS	Data entry forms and HTTP requests	To gain control of a user's session	High: Site defacement to session hijacking	Escape characters Switch to React framework	Implement a content security policy
TA3	A3	DDos	The server	To prevent access to the site	High: Site cannot be accessed	Load balancer Migrate to cloud	Graceful degradation IPS/IDS
TA4	A6	HPP	HTTP requests	Change arguments of HTTP requests to desired values	Medium: Individuals may discount meals	Blacklist Use entity encoding	Encrypting the arguments
TA5	A4	HTML Injection	Login page	Trick users into revealing passwords	Medium: Reveals customer information	Character encoding Escape characters Blacklist	Educate users on security
TA6	A1	XSRF	HTTP requests	Trick users into performing unwanted actions	High: Attacker can perform any action through HTTP requests	Require user interaction Use hidden fields within the html form	Include the session ID directly within the request Proxy server