Contents

[Website Security and Report Framework 3](#_Toc115441492)

[Cross Site Request Forgery 4](#_Toc115441493)

[SQL Injection 8](#_Toc115441494)

[Poor Firewall Configuration 15](#_Toc115441495)

[Evaluation 23](#_Toc115441496)

[References 24](#_Toc115441497)

[Figures 25](#_Toc115441498)

[Appendix 26](#_Toc115441499)

# Website Security and Report Framework

Breif

This is a technical report introducing types of cyber-attacks and vulnerabilities. This report will systematically discuss the challenges and issues of Website Security.

This report will follow a framework inspired by Adam Shostack’s ‘Threat Modelling’ four step framework. This Framework Is designed to expand on his framework of Model, Find, Address and Validate to ensure a comprehensive look into each vulnerability.

Diagram

Description automatically generated

Figure Four-Step Framework

The Threat Model testing type used will be a STRIDE per Interaction. This follows the CIA Model as a strong basis for finding vulnerabilities. These will be represented with the Microsoft Threat Modelling tool using data flow diagrams.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Threat | Violation | Definition |
| S | Spoofing | Authentication | Impersonating to gain access to unauthorised data |
| T | Tampering | Integrity | Modification of held data |
| R | Repudiation | Non Repudiation | Claim deniability for an action |
| I | Information Disclosure | Confidentiality | Transfer of information to unauthorised personnel |
| D | Denial of Service | Availability | Obstructing access to service |
| E | Elevation of Privileges | Authorization | Unauthorized access to data falsely acting as a higher form of authorisation. |

(Shostack, 2014 p29-29)

All Threat Modelling analysis description is extracted from the Microsoft STRIDE threat modelling tool report. See Appendix for screenshots of the original report.

# Cross Site Request Forgery

The goal of CSRF attacks is to deceive the target into sending a malicious request. It takes on the victim's identity and privileges in order to carry out an undesirable action on the victim's behalf. This can expose data or lead to accidental code execution and can be done via URL, image load, or HTTP Request.

Threat Modelling and Vulnerability IdentificationDiagram

Description automatically generated

Figure CSRF Basic Model

Diagram

Description automatically generated

Figure CSRF Threat Model

(Shostack, 2014 pp 53-58)

|  |  |  |
| --- | --- | --- |
| Interaction  Money | Risk:  High | Graphical user interface  Description automatically generated  Figure CSRF Money Interaction |
| External Entity Human User Potentially Denies Receiving Data | Repudiation | “Human User claims that it did not receive data from a process on the other side of the trust boundary. Consider logging or auditing to record the source time and summary of data. ( There is an audit log within the system)” |
| Data Flow Money Is Potentially Interrupted | Denial of Service | “An external agent interrupts data flowing across a trust boundary in either direction” |

|  |  |  |
| --- | --- | --- |
| Interaction  Request Identification | Risk:  High | Diagram  Description automatically generated with low confidence  Figure CSRF Identification Interaction |
| External Entity Process Transaction Potentially Denies Receiving Data | Repudiation | “Process Transaction claims that it did not receive data from a process on the other side of the trust boundary. Consider logging or auditing to record the source time and summary of data. ( There is An audit log within the system)” |
| Data Flow Request Identification Is Potentially Interrupted | Denial of Service | “An external agent interrupts data flowing across a trust boundary in either direction” |

Diagram

Description automatically generated

Figure 6 CSRF Threat Tree

When assessing the scope of consequence of this attack, it is important to consider all the potential access to damage a session.

When categorising the effect on confidentiality, integrity and availability, CSRF has the ability to influence each of these pillars of vulnerability identification. The only limit being the targets own privileges. The types of potential attacks:

* Read and Modify Application Data (Confidentiality, Integrity)
* Denial Of Service ( Availability)
* Gain/ Assume Privileges ( Confidentiality)
* Abuse of privilege to attack other targets (Integrity)

When considering STRIDE, Spoofing, Tampering Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges are all affected evident through the threat modelling and vulnerability identification.

(Shostack, 2014, PP 44-52)

Intrusion Techniques

In order to carry out an attack, the first step is to create a legitimate malicious request that the victim may comply with. See the following situation: Using a CSRF-vulnerable bankcash.co.uk, an online application, ‘Z’ wants to send ‘X’ £255. Attacker ‘C’ intends to deceive ‘Z’ into giving the money to ‘C’ instead of ‘X’. The following steps make up the attack:

1. Creating a script or URL for an attack
2. Tricking Z into executing through coercion or some type of social engineering

There are two types of attack scenarios to consider: The GET Scenario and the POST Scenario.

**GET Scenario:**

A GET Scenario will use requests to execute transfers with an address looking like this:

http://bankcash.co.uk/transfer.do?acct=X&amount=255

This address’s vital components in relation to currency and recipient are open and vulnerable to tampering. The ‘acct=X’ being the recipient is now changed to ‘acct=C’ for the attacker. The amount from £255 is now changed to £2000.

http://bankcash.co.uk/transfer.do?acct=C&amount=2000

With the first step of the attack done, now the exploited link needs to be deployed by the target. This is where the coercion needs to take place. These can be done through email or planted in commonly visited sites by the target. These can also be disguised using different masking techniques.

Exploited address imbedded into an image link

<img src="http://bankcash.co.uk/transfer.do?acct=C&amount=2000"width="8" height="8" border="8">

Exploited address disgusted as a legitimate link

href="http://bankcash.co.uk/transfer.do?acct=C&amount=2000">!Click Me!

**Post Scenario**

The Post scenario is very much the same at Get with the only difference being how the attack is executed by the target. A Post address looks like this:

http://bankcash.co.uk/transfer.do acct=X&amount=2000

This cannot be deployed using common IMG tags, but by using FORM tags

Text, letter

Description automatically generated

Figure Post Scenario Tagging

When considering other attack scenarios such as PUT and DELETE web application security models, the Same Origin Policy prevents this kind of attack unless explicitly told to open requests.

(OWSAP S, K. 2020).

Mitigation

Before finding methods for mitigation, it is important to find what doesn’t work as a preventative to intrusion.

* URL Rewriting

Initially the session ID of the victim is not known to the attacker. However, the URL includes the user's session ID which is exposed.

* Using a Secret Cookie

No matter if the end user was duped into sending the request or not, all authentication tokens will be sent. The application container merely uses session identifiers to link the request to a particular session object.

Token based mitigation

The most popular and most effective way of mitigating CSRF, is using CSRF tokens. Their characteristics being unique, secret and unpredictable.

Tokens for CSRF should be produced on the server-side. They may be created once for each request or once for each user session. As there is less time for an attacker to use the stolen tokens, per-request tokens are more secure than per-session tokens. When a client requests a token, the token issued should be verified to match the session that is in use. If the validation fails either due to it missing or false values, it will be terminated.

Deploying CSRF tokens is preferably done through the request header in HTTP via JavaScript than a hidden field. Seen in the GET requests in Intrusion Techniques, there are multiple points of potential leaks that an attacker can expose. Browser history, network logs and file logs.

A picture containing text

Description automatically generated

Figure Token Deploying

CSRF Token Storing in DOM (Document Object Model)

As seen here, a CSRF token can be added to the tag. The CSRF token can be extracted from this tag for all subsequent requests made on the page. Whilst it can be stored in a JavaScript variable it is not recommended to store locally or as cookies.



Figure CSRF Storing

**Setting up a custom header**

Text

Description automatically generatedXMLHttpRequest open() can be set to anti-csrf-token and a filter safe and unsafe HTTP’s will be defined by the csrfSafeMethod().

Figure Token Filtering

# SQL Injection

A SQL injection attack involves inserting a SQL query through the applications input into the system. An effective SQL injection attack can read sensitive data from the database, change database data (Insert/Update/Delete), and carry out database administration tasks.

Threat Modelling and VulERABILITY Identification

A screenshot of a computer

Description automatically generated with low confidence

Figure SQL Basic Model

Diagram

Description automatically generated

Figure SQL Threat Model

|  |  |  |
| --- | --- | --- |
| Interaction:  JDBC | Risk:  High | A picture containing graphical user interface  Description automatically generated  Figure JDBC Interaction |
| Spoofing of Destination Data Store SQL Database | Spoofing | “SQL Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of SQL Database. Consider using a standard authentication mechanism to identify the destination data store”. |
| Potential SQL Injection Vulnerability for SQL Database | Tampering | “SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.” |
| Potential Excessive Resource Consumption for Web Application or SQL Database | Denial Of Service | “Does Web Application or SQL Database take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.” |

|  |  |  |
| --- | --- | --- |
| Interaction: HTTPS | Risk:  High | Graphical user interface, application  Description automatically generated  Figure HTTPS Interaction |
| Elevation by Changing the Execution Flow in Web Application | Elevation of Privilege | “An attacker may pass data into web application in order to change the flow of program execution within web application to the attacker's choosing.” |
| Potential Process Crash or Stop for Web Application | Denial of Service | “Web application crashes, halts, stops or runs slowly; in all cases violating an availability metric.” |
| Potential Data Repudiation by Web Application | Repudiation | “Web Application claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.” |
| Web Service Process Memory Tampered | Tampering | “If Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Web Application executes (for example, passing back a function pointer.), then Web Service can tamper with Web Application. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.” |

Diagram

Description automatically generated

Figure Injection Attack Tree

When assessing the scope of consequence of this attack, it is important to consider all the potential access to damage a databases data.

When categorising the effect on confidentiality, integrity and availability, SQL Injections have the ability to influence each of these pillars of vulnerability identification. The only limit being the capability of the attacker. The types of potential attacks:

There are two main factors that attract SQL attacks:

1. Ubiquity of SQL systems with shared vulnerabilities
2. Data value contained within the database

Access could open up access such as:

* A successful exploitation of a SQL injection vulnerability can make it possible to alter permission information stored in a SQL database. (Access Control)
* It might be feasible to log in as another user without having previously known the password if weak SQL commands are used to check user names and passwords. (Access Control)
* SQL injection vulnerabilities cause loss of confidentiality because SQL databases frequently contain sensitive data.
* With a SQL injection attack, it is possible to modify or even destroy important information in addition to reading it.

When considering STRIDE, Spoofing, Tampering Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges are all affected evident through the threat modelling and vulnerability identification.

(CWE-89 n.d.)

Intrusion Techniques

There is an increasingly widespread problem with websites that use databases is SQL Injection. Because the weakness is so simple to find and exploit, any website or software program with even a small userbase is likely to see an attempt at this kind of assault.

The attack is carried out by inserting a character into the data input to introduce previously unestablished SQL commands onto the control space. This problem results from SQL's lack of a clear separation between the control and data spaces.

This first example is in Java. This section would enable a hacker to insert code into the database query that would be run. An attacker can insert whatever SQL code they want thanks to the query's simple add of the invalidated "customerName" field. This kind of approach is very typical.

Text

Description automatically generated

Figure Java Code Exposure

This second example in C# produces and executes a SQL query that searches for items with a given name. The search restricts the objects displayed to those whose owners that have user names that match the presently authenticated user.

Text, letter

Description automatically generated

Figure C# Query Search

The following is the request that this code aims to run:



Figure Code to Run

The query, however, only functions properly if (itemName) does not contain a single-quote character because it is built dynamically by joining a user input string and a constant base query string. If an attacker with the user name ‘Greg’ enters the string "name' OR 'a'='a" for itemName, then the query becomes the following:

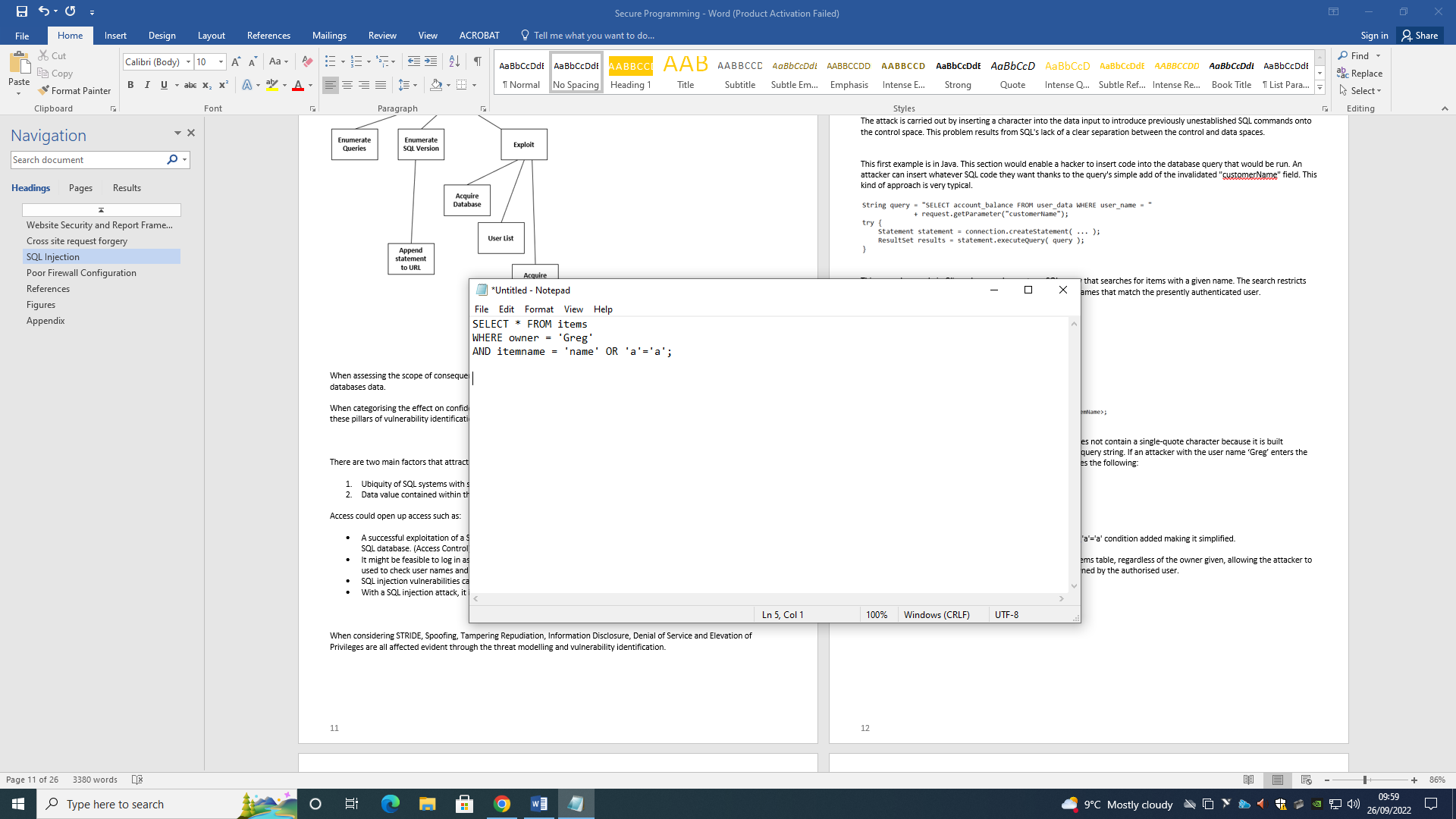


Figure Item Query Example

The where clause always evaluates to true because of the OR 'a'='a' condition added making it simplified.

The query is now simpler and returns all items stored in the items table, regardless of the owner given, allowing the attacker to get around the restriction that the query only return items owned by the authorised user.

Mitigation

Presenting three practical mitigations: Parameterized Queries, Stored Procedures and Allow-list Input Validation.

**Parameterized Queries**

The developer must first define all the SQL code for parameterized queries before passing each parameter to the query. No matter what user input is provided, the database can tell the difference between code and data thanks to this coding approach.

The parameterized query in the safe example below would not be vulnerable and would instead search for a username that literally matched the complete string Greg' or '1'='1 if an attacker were to enter the userID of Greg' or '1'='1.

**Java example**

Java uses PreparedStatement() with bind variablesText, letter

Description automatically generated. It is parameterized query, to execute the same database query. Not just Java support parameterized queries. HQL, ASP and many others support this method.

Figure Java Parameterized Queries

**Stored Procedures**

When done securely, which is the standard for most stored procedure languages, a number of common programming expressions for stored procedures have the same impact as using parameterized queries. Prepared statements and stored procedures differ in that the SQL code for a stored procedure is created, saved, and called from the application rather than the database.

**Java example**

The identical database query is carried out by CallableStatement, Java's implementation of the stored procedure interface, in the following code example. The sp\_getAccountBalance stored procedure would need to have the same capability as the aforementioned query and be predefined in the database.

Graphical user interface, text, application

Description automatically generated

Figure Java Stored Procedures

**Allow-list Input Validation**

Although those values come from the code and not from user parameters, input validation or query redesign is the best defence for table or column names.

To ensure that invalid user input does not show up in the query, parameter values that are used to target different table names and column names should be translated to the legal or expected table or column names.

Example of table name validation:

Graphical user interface, text, application

Description automatically generated

Figure List input Validation

Since it is now known to be one of the permitted and expected values for a table name in this query, the tableName can then be included directly to the SQL statement.

It would be ideal to transform the user-supplied data to a Boolean before using that Boolean to choose the safe value to append to the query for the sort order.

Text

Description automatically generated with medium confidence

Figure String Query

This guarantees it is safe to do so each time user input can be transformed into a non-String, such as a date, numeric, Boolean, enumerated type, etc. before it is attached to a query or used to pick a value to append to the query.

# Poor Firewall Configuration

In order to protect users entities from cyber-attacks, firewalls must be correctly configured and serve a key role in network security.

The firewall can be kept safe by configuring domain names and Internet Protocol (IP) addresses. Based on network type, such as public or private, firewall policy configuration can be set up with security rules that restrict or allow access to thwart a potential hacker or viral attacks.

Chart, diagram

Description automatically generatedThreat Modelling And Vulnerability Identification

Figure Firewall Basic Model

Diagram

Description automatically generated

Figure Firewall Threat Model

|  |  |  |
| --- | --- | --- |
| Interaction:  Cache | Risk:  High | Figure Cache Interaction |
| Spoofing of Destination Data Store Cache | Spoofing | “Cache may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Cache. Consider using a standard authentication mechanism to identify the destination data store”. |
| Potential Excessive Resource Consumption for Browser Client or Cache | Denial of Service | “Does Browser Client or Cache take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout”. |
| Spoofing of Source Data Store Cache | Spoofing | “Cache may be spoofed by an attacker and this may lead to incorrect data delivered to Browser Client. Consider using a standard authentication mechanism to identify the source data store.” |
| Weak Access Control for a Resource | Information Disclosure | “Improper data protection of Cache can allow an attacker to read information not intended for disclosure. Review authorization settings.” |

|  |  |  |
| --- | --- | --- |
| Interaction:  Firewall | Risk:  High | Figure Firewall Interaction |
| Spoofing the Firewall External Entity | Spoofing | “Firewall may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Firewall. Consider using a standard authentication mechanism to identify the external entity.” |
| External Entity Firewall Potentially Denies Receiving Data | Repudiation | “Firewall claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.” |
| Browser Client May be Subject to Elevation of Privilege Using Remote Code Execution | Elevation of privilege | “Firewall may be able to remotely execute code for browser client.” |

Diagram

Description automatically generated

Figure 28 Firewall Threat Tree

When assessing the scope of consequence of an attack, it is important to consider all the potential access to damage a system

When categorising the effect on confidentiality, integrity and availability, poor firewall implementation have the ability to influence each of these pillars of vulnerability identification. The only limit being the capability of the attacker. The types of potential attacks:

* Insider attacks help divide up the many resources on a network so that attackers must exert more effort to travel from one machine to another. Lack of a well-rounded firewall security can cause confidentiality and integrity issues.
* Lack of rigorous packet checking by a firewall at the layer 7 can cause the higher chance of infiltration more likely.
* DDOS attacks can be mitigated however firewalls are still vulnerable to protocol attacks. Since firewalls are an integral defence in a security architecture poor configuration can proliferate attacks.

When considering STRIDE, Spoofing, Tampering Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges are all affected just as CIA evident through the threat modelling and vulnerability identification.

(Dave Burton 2020)

Intrusion Techniques

Despite the fact that Firewalls and IDS can stop malicious packets from entering a network, an attacker can send altered packets to the target in order for it to get past the IDS/Firewall. These evasion methods available in a multitude.

**Packet Fragmentation**

The attack technique known as packet fragmentation involves breaking up the probing packets into smaller pieces and transmitting them to the target network. The fragmentation results in too many packets, which uses up more CPU and network resources. A tool that can be used is SYN/FIN scanning through NMAP commands do display these kind of intrusions.

Text

Description automatically generated

Figure Packet Fragmentation

**Source Port Manipulation**

Source port manipulation is a method for getting around firewalls and IDS that involves replacing the real port numbers with generic ones in order to get over their restrictions.

The source port manipulation in NMAP is done with the -g option. Below is an illustration of a source code manipulation command.

A screenshot of a computer

Description automatically generated with medium confidence

Figure Source Port Manipulation

**IP Address Decoy**

Decoys' IP addresses are manually specified to avoid IDS and Firewalls and give the impression that both the host and the target are network scanning. Multiple IP addresses are generated by NMAP scan, making it challenging for the target security frameworks to determine the source from the registered logs.

Text

Description automatically generated

Figure IP Address Decoy

**IP Spoofing**

Finding a computer's IP address allows an attacker to change the packet headers before sending request packets to the target device while posing as a legitimate host. The packets are actually sent from the attacker's machine, despite the fact that they also seem to be coming from a reliable source.

Hping3 can be used to perform the IP spoofing.

Text, chat or text message

Description automatically generated

Figure 32 IP Spoofing

**Backdoor Infiltration in Python**

|  |  |
| --- | --- |
| Code | Explanation |
|  | The following Python modules are required to assemble our victim-side script. The  client-side methods are provided through socket. It will be necessary for subprocess to run a terminal command. Only the transformation of a Python list-string into a list will require ast. |
|  | Write an initializer method for the Victim class that accepts the server's IP address and port as parameters. |
|  | Create a TCP connection and connect to the server. |
|  | Online:Instantly run any command and return the standard output, or stdout, when interacting online.  Offline: Executes a series of commands and then gives the attacker the complete results.  To reduce the amount of interactions that enhance the likelihood of being seen by an intrusion detection system, it is preferable to operate offline (ids)  It should be recognised that the primary technique utilised to execute a command is subprocess Popen(). |
|  | Build a Victim class object, use the connect to server() function to establish a connection with the server, and select between the online and offline methods by using the method. |

Figure 33 Victim Backdoor Code

Mitigation

Poor network configuration mitigation starts from the very beginning of a development lifecycle. Since Firewalls are part of an entire infrastructure of network and system defence, firewall configuration is not the only security component to focus on. It is useful to take into account the industry standards that ensure a security level that allows for at least common attacks to be mitigated. Specifically referencing ISO/IEC 27033.

There are six steps in this management system that allow for a comprehensive look at how network security should be.

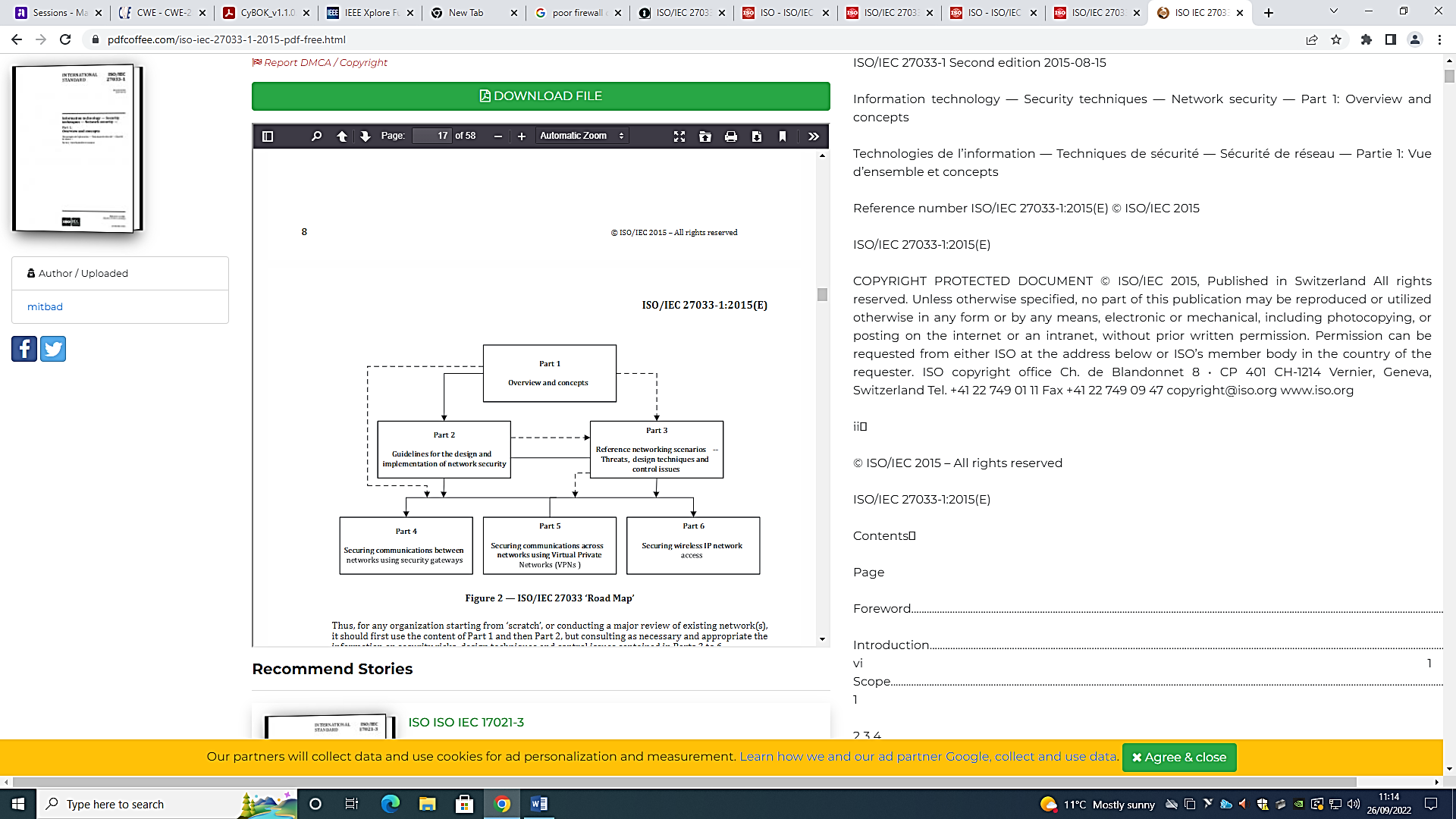


Figure ISO Lifecycle

Finding and assessing potential area vulnerabilities within a network model. Using the CIA concept will need to be a template for finding risks and ensuring it upholds within the network if there was an attack. Consider:

* Coding and data secrecy in networks and systems connected to networks.
* Coding and data integrity in networks and systems connected to networks
* Information accessibility, network services, and systems linked to networks.
* Non-Repudiation of network transactions (commitments),
* Accountability of network transactions,
* Information authenticity as well as that of network users and administrators
* Information and code reliability in networks and systems connected to networks.
* Control over unauthorised use and exploitation of network resources, taking organisation policies and responsibilities in connection to law and regulation into account.

(ISO/IEC n.d.)

Diagram

Description automatically generated

Figure CIA Within a Network

**Third Party Anti Malware Implementation**

Anti-malicious code software's job is to examine data and programmes for suspicious patterns linked to malware. Signatures are a collection of patterns that must be examined for; they should be updated periodically or whenever new signatures are made available for high-risk malware alerts.

Anti-malicious code software should be installed on both the central system servers, and the remote computers when using remote access. Administrators have to know that using unauthorised communication to outside parties exposes the system to higher than usual dangers from malicious malware.

Diagram

Description automatically generated

Figure Anti Malware Process

# Evaluation

In regards to evaluating in the context of web security, it is important to establish potential assets that a web service may hold. This usually accumulates data involving people whether staff or user data such as customers. According to a Cisco 2017 Annual Cybersecurity Report, organisations lose more than 20% of their consumer base just from online breaches.

Steps to ensure a secure web security foundation:

Managed Web Host Services and Servers

Many provide server security capabilities that can secure website data better. Examine a number of web hosting providers based on a number of criteria, including uptime, response time during downtime, the causes of downtime, the calibre of the customer support, and the advantages offered as part of the package, such as managed services, SSL certificates, data storage and scalability options, backups, and supported web applications. The website components may be updated and maintained without any downtime using hosts that provide several environments for development.

Implementation of a Web Application Firewall

Programmed bots are available that continuously check websites, especially new ones, for vulnerabilities. One approach to protect the site from these automated threats is by adding a web application firewall (WAF).

Connection Encryption

These connections must be encrypted for any type of transaction. Establish a secure handshake between the cooperating website and the devices of the clients using Secure Sockets Layer (SSL) certificates.

Competent Penetration Testing

To guarantee that no vulnerabilities are overlooked, be sure to hire a qualified engineer and a penetration tester who can perform rigorous testing. Any one of these weaknesses could lead to a data breach, which could irreparably damage the organization’s reputation. Database queries should be protected from SQL injections. Create it so users can’t change it,

(CloudSecureTech 2017)

# References

Shostack, A. (2014). The Four-Step Framework. In Threat modelling: Designing for security (pp. 29–29). essay, Wiley.

Shostack, A. (2014). Types of Diagrams. In Threat modelling: Designing for security (pp. 44–52). essay, Wiley.

Shostack, A. (2014). STRIDE Threat Trees. In Threat modelling: Designing for security (pp. 53–58). essay, Wiley.

CWE-89: Improper Neutralization of Special Elements used in an SQL Command (‘SQL Injection’) (3.4.1). [online] Available at: https://cwe.mitre.org/data/definitions/89.html. [22 Sep. 2022].

OWSAP S, K. (2020). Cross Site Scripting (XSS) | OWASP. [online] Owasp.org. [online] Available at: https://owasp.org/www-community/attacks/xss/.[23 Sep. 2022].

Microsoft. (n.d.). Microsoft Security Development Lifecycle Practices. [online] Available at: https://www.microsoft.com/en-us/securityengineering/sdl/practices#practice4 [24 Sep. 2022].

ISO/IEC (n.d.) ISO/IEC 27033 IT network security standard. [online] Available at: https://www.iso27001security.com/html/27033.html[24 Sep. 2022].

Dave Burton (2020) The Dangers of Firewall Misconfigurations and How to Avoid Them [online] Available at: https://www.akamai.com/blog/security/the-dangers-of-firewall-misconfigurations-and-how-to-avoid-them[26 Sep. 2022].

CloudSecureTech (2017). Importance Of Web Security In The Web Development Stage [online] Available at: https://www.cloudsecuretech.com/web-security-no-longer-overlooked-web-development-stage/[29 Sep. 2022].

# Figures

[Figure 1 Four-Step Framework 3](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441364)

[Figure 2 CSRF Basic Model 4](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441365)

[Figure 3 CSRF Threat Model 4](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441366)

[Figure 4 CSRF Money Interaction 5](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441367)

[Figure 5 CSRF Identification Interaction 5](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441368)

[Figure 6 CSRF Threat Tree 6](#_Toc115441369)

[Figure 7 Post Scenario Tagging 7](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441370)

[Figure 8 Token Deploying 8](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441371)

[Figure 9 CSRF Storing 8](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441372)

[Figure 10 Token Filtering 8](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441373)

[Figure 11 SQL Basic Model 9](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441374)

[Figure 12 SQL Threat Model 9](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441375)

[Figure 13 JDBC Interaction 10](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441376)

[Figure 14 HTTPS Interaction 10](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441377)

[Figure 15 Injection Attack Tree 11](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441378)

[Figure 16 Java Code Exposure 12](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441379)

[Figure 17 C# Query Search 12](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441380)

[Figure 18 Code to Run 12](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441381)

[Figure 19 Item Query Example 12](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441382)

[Figure 20 Java Parameterized Queries 13](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441383)

[Figure 21 Java Stored Procedures 13](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441384)

[Figure 22 List input Validation 14](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441385)

[Figure 23 String Query 14](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441386)

[Figure 24 Firewall Basic Model 15](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441387)

[Figure 25 Firewall Threat Model 15](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441388)

[Figure 26 Cache Interaction 16](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441389)

[Figure 27 Firewall Interaction 16](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441390)

[Figure 28 Firewall Threat Tree 17](#_Toc115441391)

[Figure 29 Packet Fragmentation 18](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441392)

[Figure 30 Source Port Manipulation 18](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441393)

[Figure 31 IP Address Decoy 18](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441394)

[Figure 32 IP Spoofing 19](#_Toc115441395)

[Figure 33 Victim Backdoor Code 20](#_Toc115441396)

[Figure 34 ISO Lifecycle 21](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441397)

[Figure 35 CIA Within a Network 22](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441398)

[Figure 36 Anti Malware Process 22](file:///C:\Users\Abdur-rahman\Documents\Bsc\Secure%20Programming\Secure%20Programming.docx#_Toc115441399)

# Appendix

Original Threat Model Report

Graphical user interface

Description automatically generated

Diagram

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated