**Title of the project : "Integrated Cybersecurity Analytics** **”**

Threat Intelligence, Incident Response, SEIM, Nessus, CWE Vulnerabilities, and Web Application Testing

**Overview :**

This project aims to provide comprehensive analytics on various cybersecurity domains, including Threat Intelligence, Incident Response strategies, Security Information and Event Management (SIEM) principles using Qradar as a focal tool, exploration of Nessus vulnerability scanning, understanding of Common Weakness Enumeration (CWE) vulnerabilities, and practical Web Application Testing techniques. This project offers hands-on experience, allowing participants to delve into real-world scenarios, tools, and methodologies used in securing digital environments and combating cyber threats effectively.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **name** | **collage** | **contact** |
| **1** | **Renu Bahuguna** | **Coer University**  **Roorkee** | **9917312870** |

|  |  |  |
| --- | --- | --- |
| **S.no** | **Vulnerability Name** | **CWE - No** |
| **1** | Identification and Authentication Failure | CWE-287:  Improper Authentication (4.13) |
| **2** | Software and Data integrity failure | CWE-353:  Missing Support for Integrity Check |
| **3** | Broken Access Control | CWE-284:  Improper Access Control |
| **4** | Cryptographic failure | CWE-327:  Use of a Broken or Risky Cryptographic Algorithm |
| **5** | Injection | CWE-94:  Improper Control of Generation of Code ('Code Injection') |
| **6** | Insecure Design | CWE-657:  Violation of Secure Design Principles |
| **7** | Security Misconfiguration | CWE-15:  External Control of System or Configuration Setting |
| **8** | Software login and monitoring failure | CWE-778:  Insufficient Logging |
| **9** | Server Side Request Forgery | CWE-918:  Server-Side Request Forgery (SSRF) |
| **10** | Vulnerable And Outdated Components | CWE-1104:  Use of Unmaintained Third Party Components |

1. Vulnerability Name: Identification and Authentication Failure

CWE-287: Improper Authentication OWASP Category: OWASP Top Ten 2007

Description: When an actor claims to have a given identity, the product does not prove or insufficiently proves that the claim is correct.

Business Impact: CWE-287's Improper Authentication can lead to data breaches, financial losses from fraudulent activities, compliance violations resulting in fines, reputational damage eroding customer trust, and operational disruptions due to security incidents. Businesses risk lawsuits and legal consequences, impacting their finances and reputation. Implementing robust authentication measures and regular security assessments is crucial to mitigate these risks and safeguard against unauthorized access.

1. Vulnerability Name: Software and Data integrity failure CWE-353: Missing Support for Integrity Check

OWASP Category: A08:2021

Description: Product uses a transmission protocol that does not include a mechanism for verifying the integrity of the data during transmission, such as a checksum.

Business Impact: It lead to compromised data integrity, enabling unauthorized modifications or corruption. Business impacts include loss of trust, potential legal liabilities, operational disruptions, and compromised system reliability, undermining the organization's reputation and leading to financial losses. Resolving this involves implementing integrity checks to safeguard against unauthorized alterations and ensuring data remains intact and trustworthy.

1. Vulnerability Name: Broken Access Control

CWE-284: Improper Access Control OWASP Category:A01:2021

Description : The product does not restrict or incorrectly restricts access to a resource from an unauthorized actor.

Business Impact: Software and Data Integrity Failure (CWE-353) poses a grave risk to businesses, potentially resulting in corrupted data, unauthorized alterations, compromised system reliability, data breaches, leading to regulatory non-compliance, reputational harm, financial losses, and legal consequences. Implementing robust integrity checks and data validation mechanisms is imperative to mitigate these risks and maintain the trustworthiness and security of business-critical information.

1. Vulnerability Name: Cryptographic failure

CWE-327: Use of a Broken or Risky Cryptographic Algorithm

OWASP Category: A02:2021

Description: product uses a broken or risky cryptographic algorithm or protocol.

Business Impact: The business impact of Cryptographic Failure (CWE-327) is profound, as it can lead to severe security breaches, compromised data confidentiality, unauthorized access to sensitive information, financial losses due to fraud or theft, regulatory non-compliance, damaged reputation, legal liabilities, and erosion of customer trust. Implementing strong, vetted cryptographic algorithms and regularly updating cryptographic protocols are imperative to mitigate these risks and ensure robust protection of sensitive data and systems.

1. Vulnerability Name: Injection

CWE-94: Improper Control of Generation of Code ('Code Injection')

OWASP Category:A03:2021

Description: Product constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment.

Business Impact: SQL injection attacks enable malicious insertion of SQL queries via client input, jeopardizing database security. Successful exploits can extract sensitive data, manipulate database content, execute admin tasks (like DBMS shutdown), access DBMS file system files, and, in some instances, issue commands to the underlying operating system. These attacks manipulate data inputs to execute SQL commands, demanding robust security measures like input validation, prepared statements, and regular audits to prevent exploitation and safeguard systems.

1. Vulnerability Name: Insecure Design

CWE-657: Violation of Secure Design Principles OWASP Category: A04:2021

Description:The product violates well-established principles for secure design.

Business Impact: The business impact of Insecure Design (CWE-657) is substantial, potentially resulting in compromised systems, increased susceptibility to cyber attacks, data breaches, reputational damage, financial losses due to exploitation, regulatory penalties for non-compliance, and prolonged remediation efforts. Addressing violations of secure design principles demands significant resources, including redesigning systems and applications, to mitigate risks and enhance overall security posture, safeguarding critical assets and customer trust.

1. Vulnerability Name: Security Misconfiguration

CWE-15: External Control of System or Configuration Setting

OWASP Category: A05:2021

Description : One or more system settings or configuration elements can be externally controlled by a user.

Business Impact: Security Misconfigurations (CWE-15) pose significant business risks, potentially leading to unauthorized access, data breaches, service disruptions, compliance violations, and reputational damage. Exploitation of misconfigurations enables attackers to compromise systems, access sensitive information, disrupt operations, and exploit weaknesses in the infrastructure. Mitigating these vulnerabilities demands regular audits, proper configuration management, and adherence to security best practices to fortify defenses, ensuring robust protection of critical assets and maintaining customer trust.

1. Vulnerability Name: software login and monitoring failure

CWE-778: Insufficient Logging

OWASP Category:A09:2021

Description: When a security-critical event occurs, the product either does not record the event or omits important details about the event when logging it.

Business impact: The business impact of "Software Login and Monitoring Failure" or CWE-778 (Insufficient Logging) is substantial. Insufficient logging and monitoring can lead to delayed detection of security incidents, hindering the ability to identify and respond promptly to threats. This vulnerability can result in prolonged unauthorized access, unnoticed malicious activities, difficulty in forensic investigations, compliance breaches, reputational damage, and increased vulnerability to cyber threats. Mitigating this issue requires robust logging mechanisms, adequate monitoring, and timely analysis of logs to enhance threat detection and response capabilities, minimizing the impact of security incidents on business operations and data integrity.

1. Vulnerability Name: Server Side Request Forgery

CWE-918: Server-Side Request Forgery (SSRF)

OWASP Category A10:2021

Description : The web server receives a URL or similar request from an upstream component and retrieves the contents of this URL, but it does not sufficiently ensure that the request is being sent to the expected destination.

Business Impact: The business impact of Server-Side Request Forgery (SSRF), categorized under CWE-918 and OWASP A10:2021, can be severe. SSRF vulnerabilities enable attackers to manipulate server requests, potentially leading to unauthorized access to internal systems, sensitive data exposure, and exploitation of internal resources. This vulnerability may facilitate attacks like data theft, bypassing security controls, service disruption, and in some cases, accessing sensitive information from within the network. Exploitation of SSRF can result in reputational damage, regulatory penalties, financial losses due to data breaches, and disruption of services. Mitigation involves robust input validation, network segregation, and secure coding practices to prevent SSRF, safeguarding against unauthorized access and data exposure.

1. Vulnerability Name: Vulnerable And Outdated Components

CWE-1104: Use of Unmaintained Third Party Components

OWASP Category: A06:2021

Description: The product relies on third-party components that are not actively supported or maintained by the original developer or a trusted proxy for the original developer.

Business Impact: Relying on outdated or unmaintained third-party components exposes systems to known vulnerabilities, increasing the risk of exploitation by attackers. This vulnerability may lead to severe consequences such as data breaches, loss of sensitive information, regulatory non-compliance penalties, reputational damage, service disruption, and financial losses due to exploitation or system compromise. Mitigating this risk requires proactive monitoring of third-party libraries, regular updates, patch management, and adopting secure development practices to reduce reliance on vulnerable components, ensuring a more resilient and secure application infrastructure.

**SANS Top 20 Security Vulnerabilities In Software Applications**

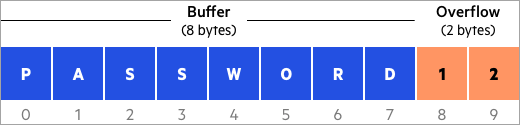
**List Of SANS Top 20 Critical Vulnerabilities In Software**

|  |  |  |
| --- | --- | --- |
| **S\_No** | **CWE No.** | **Vulnerabilities Name** |
|  | **CWE-119**: | Memory Buffer Error |
|  | **CWE-79**: | Cross-site Scripting |
|  | **CWE-20** | Unvalidated Input Error |
|  | **CWE-200**: | Sensitive Information Exposure Error |
|  | **CWE-125**: | Out-of-bounds Read Error |
|  | **CWE-89**: | SQL Injection |
|  | **CWE-416**: | Free Memory Error |
|  | **CWE-190**: | Integer Overflow Error |
|  | **CWE-352**: | Cross-Site Request Forgery |
|  | **CWE-22**: | Traversal |
|  | **CWE-78**: | Command Injection |
|  | **CWE-787**: | Out-of-bounds Write Error |
|  | **CWE-287**: | Improper Authentication Error |
|  | **CWE-476**: | Dereferencing NULL Pointer |
|  | **CWE-732**: | Incorrect Permission Assignment |
|  | **CWE-434**: | Unrestricted File Upload |
|  | **CWE-611**: | Information Exposure through XML Entities |
|  | **CWE-94**: | Code Injection |
|  | **CWE-798**: | Hard-coded Access Key |
|  | **CWE-400**: | Uncontrolled Resource Consumption |

**#1) CWE-119: Memory Buffer Error**

This flaw is usually introduced during Architecture and Design, Implementation, Operation stages of the SDLC.This buffer overflow happens when an application process tries to store more data than it can hold in the memory. Since the buffers can only store some level of data and when that level is reached and exceeded, the data flows to another memory location which can corrupt the data already contained in that buffer.

The example below shows a buffer allocated with 8bytes storage. But it overflowed by 2bytes because of more data was sent for execution.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/Application-Sec-1.png)

**#2) CWE-79: Cross-site Scripting**

Cross-site Scripting (XSS) is an injection attack that usually happens when a malicious actor or an attacker injects malicious or harmful script into a web application which can be executed through the web browsers. Once the malicious script finds its way into the compromised system, it can be used to perform different malicious activities.

**Cross-site scripting occurrence:**

* When un-validated and un-trusted data are inputted into a web application through the web form request.
* When the web application instantly output a web page that contains this malicious data.
* During the process of generating a page, the software fail to validate against the data, which house the content that can be executed by a web browser, like HTML and JavaScript.
* The victim unknowingly visits the page that was generated through a web browser, that house the malicious script that was injected through the use of the untrusted data.
* The malicious script comes from a page that was sent by the attacker’s web server, the compromised system web browser then goes ahead to process the malicious script.
* This action violates the web browser’s policy about same origin, which stipulates that scripts coming from one domain should not have access to resources or execute code in another different domain except its own domain.

**#3) CWE-20: Unvalidated Input Error**

The application receives input, but fails to validate the input, whether it has all necessary details needed for it to be accepted into the system for processing.

When there is input sanitization, this can be used to check any potentially dangerous inputs in order to ensure that the inputs are safe to be processed with the source code or when it’s an input that is needed to communicate with other components.

The below images show that a good application should not accept script or command as an input. If such inputs are not properly sanitized, the application will process it thinking it’s a valid request.

**#4) CWE-200: Sensitive Information Exposure Error**

This happens when the application knowingly and unknowingly exposes information that is confidential and sensitive to an attacker who does not have the authorization to access these information.

**Below are some sensitive information that could be exposed:**

* Personal information like personal messages, financial data, health status records, geographic location, or contact details
* System configuration details and environment, **for example,** the operating system and installed packages
* Business Record and intellectual property
* Network configuration details
* Internal application state
* Metadata like the message headers

**#5) CWE-125: Out-of-bounds Read Error**

This occurs when the application reads data past the normal level, either to the end or before the beginning of the buffer. This gives unprivileged access to an attacker to read sensitive information from other memory locations, which can as well leads to a system or application crash.

If you now check the below example, you will see that the IF statement needs to be modified to include a minimum range validation.

**#6) CWE-89: SQL Injection**

[SQL injection](https://www.softwaretestinghelp.com/sql-injection-how-to-test-application-for-sql-injection-attacks/) is a form of security vulnerability whereby the attacker injects a Structured Query Language (SQL) code to the Webform input box in order to gain access to resources or change data that is not authorized to access.

This vulnerability can be introduced to the application during the design, implementation, and operation stages.

If the input values are correct, the user is granted access to the application or request, but if the values are incorrect, access will be denied.

**#7) CWE-416: Previously Freed Memory**

This issue is caused by the referencing of memory after it has been released, which can seriously lead to a program crash. When you use a previously freed memory, this can have adverse consequences, like corrupting of valid data, arbitrary code execution which is dependent on the flaw timing.

**Two common causes are:**

* Error conditions within the software and in some other exceptional cases.
* No explanation as to which part of the program caused the free memory.

In this instance, the memory is allocated to another pointer immediately after it has been freed. The previous pointer to the freed memory is used again and now points to somewhere around the new allocation. By the time the data is changed, this can corrupt the used memory and could make the application behave in an undefined way.

**#8) CWE-190: Integer Overflow Error**

When a calculation is processed by an application and there is a logical assumption that the resulting value will be greater than the exact value, integer overflow happens. Here, an integer value increases to a value that cannot be stored in a location.

This issue can trigger buffer overflows, which can be used to execute arbitrary code by an attacker. This integer overflow error is usually introduced into the system during the Design and Implementation stages of the SDLC.

**#9) CWE-352: Cross-Site Request Forgery**

This is when a web application does not sufficiently verify the HTTP request, whether the request was actually coming from the right user or not. The webservers are designed to accept all requests and to give a response to them.

The below image shows an attacker inducing a user to perform actions that they do not intend to perform.

**#10) CWE-22: Directory Traversal**

Directory traversal or file path traversal is a web security vulnerability that allows an attacker to read arbitrary files on the server that is currently running an application.

**#11) CWE-78: OS Command Injection**

It is about the improper sanitization of special elements that may lead to the modification of the intended OS command that is sent to a downstream component. An attacker can execute these malicious commands on a target operating system and can access an environment to which they were not supposed to read or modify.

**#12) CWE-787: Out-of-bounds Write Error**

This happens when the application writes data past the end, or before the beginning of the designated buffer.When this happens, the end result is usually data corruption, system, or application crash. What the application does is some sort of pointer arithmetic that is used in referencing a memory location outside the buffer boundaries.

**#13) CWE-287: Improper Authentication Error**

This is when an attacker claims to have a valid identity but the software failed to verify or proves that the claim is correct.

A software validates a user’s login information wrongly and as a result, an attacker could gain certain privileges within the application or disclose sensitive information that allows them to access sensitive data and execute arbitrary code.

**#14) CWE-476: Dereferencing A NULL Pointer**

Dereferencing a null pointer is when the application dereferences a pointer that was supposed to return a valid result instead returns NULL and this leads to a crash. Dereferencing a null pointer can happen through many flaws like race conditions and some programming error.

The processes that are performed with the help of the NULL pointer usually lead to failure, and the possibility of carrying out the process is very slim. This helps attackers to execute malicious code.

**#15) CWE-732: Incorrect Permission Assignment**

This vulnerability happens when an application assigns permissions to a very important and critical resource in such a manner that exposed the resource to be accessed by a malicious user.

**16) CWE-434: Unrestricted File Upload**

This vulnerability occurs when the application does not validate the file types before uploading files to the application. This vulnerability is language independent but usually occurs in applications written in ASP and PHP language.

A dangerous type of file is a file that can be automatically processed within the application environment.

**#17) CWE-611: Information Exposure Through XML Entities**

When an XML document is uploaded into an application for processing and this document contains XML entities with uniform resource identifier that resolves to another document in another location different from the intended location. This anomaly can make the application to attach incorrect documents into its output.

**#18) CWE-94: Code Injection**

The existence of code syntax in the user’s data increases the attacker’s possibility to change the planned control behavior and execute arbitrary code. This vulnerability is referred to as “injection weaknesses” and this weakness could make a data control become user-controlled.

**#19) CWE-798: Hard-coded Access Key**

This is when the password and access key is hard coded into the application directly for inbound authentication purpose and outbound communication to some external components and for encryption of internal data. Hard-coded login details usually cause vulnerability that paves the way for an attacker to bypass the authentication that has been configured by the software administrator.

The system administrator will always find it very hard to detect this vulnerability and fix it.

**There are two main streams to this weakness:**

* **Inbound**: The application contains an authentication system that validates the input credentials against the hard-coded details.
* **Outbound**: The application connects to another system and details for connecting to the other system are hardcoded into the system.

In the Inbound stream, there is always a default administrator’s account that is created, and the credentials to access it will be hard-coded into the application and associated with that default admin account.

**#20) CWE-400: Uncontrolled Resource Consumption**

This vulnerability happens when the application does not control the allocation properly and maintenance of a limited resource, this allows an attacker to be able to influence the amount of resources consumed, which will eventually lead to the exhaustion of available resources.

Part of the limited resources includes memory, file system storage, database connection pool entries, and CPU.

**The three different instances which can lead to resource exhaustion are:**

* Shortage of throttling for the number of allocated resources
* Losing out all references to a resource before reaching the shutdown stage
* Failure to close/returning a resource after processing

**The issue of resource exhaustion is usually as a result of incorrect implementation of the following scenarios:**

* Error conditions and other exceptional circumstances.
* There is mixed reaction over which part of the program releases the resource.