SECURIN ASSESSMENT

GARIMI REDDY SHYASHYANKHA REDDY

BL.EN.U4CSE20018

AMRITA SCHOOL OF ENGINEERING

NVD-CVE-API

STEP- BY-STEP EXAPLANATION:

1. **Creating the Index.html page:**

This HTML code is creating a redirection page. When someone accesses this HTML file, the browser will immediately redirect them to another page located at cves/list.html. The <meta> tag inside the <head> section specifies the refresh behaviour, where content="0; URL=cves/list.html" indicates that the redirection should occur instantaneously. The <h1> tag inside the <body> section displays the text "Redirecting..." to inform the user about the redirection process.

1. **Creating the list.html page:**

This HTML file provides the structure for a web page displaying a list of Common Vulnerabilities and Exposures (CVEs). It includes a container with a title, a table wrapper for displaying CVE data, and option buttons for controlling the display, such as selecting the number of results per page, sorting by last modified date or published date, and navigating between pages. The table itself is initially empty and will be populated dynamically with CVE data using JavaScript. This structure allows users to easily view and interact with CVE information, making it a user-friendly interface for accessing and managing vulnerability data.

1. **Styling the page using style.css:**

The .container class ensures the content remains visually centered and maintains a readable text alignment, while the table element is centered within its container. Interactive feedback is introduced through the table tr td:hover rule, which changes the cursor to a pointer when hovering over table cells, indicating their clickable nature. Key elements, such as the header row and table cells, are styled for clarity and readability, with the header row distinguished by a heavier font weight and a background color. Lastly, the .optionButtons class arranges the pagination and sorting buttons in a visually consistent manner, promoting ease of use and navigation within the CVE list interface. Together, these CSS rules contribute to a visually appealing and user-friendly experience for exploring CVE data.

1. **Creating a JavaScript page to render the data into table:**

This JavaScript code fetches data from the National Vulnerability Database (NVD) API and renders it into an HTML table. The fetchData function sends a request to the API with parameters for pagination, retrieving a subset of results based on the startIndex and resultsPerPage variables. Upon receiving the data, it calls the renderTable function, which populates the table with CVE information sorted by the last modified date. Event listeners are set up for pagination buttons and dropdown menu for changing results per page, triggering a new data fetch when interacted with. Additionally, event listeners are attached to the table headers for sorting the table by last modified date and published date. When a table row is clicked, it redirects the user to a new page with more details about the selected vulnerability. Overall, this code provides dynamic interaction with CVE data, allowing users to navigate through the dataset efficiently and access detailed information when needed.

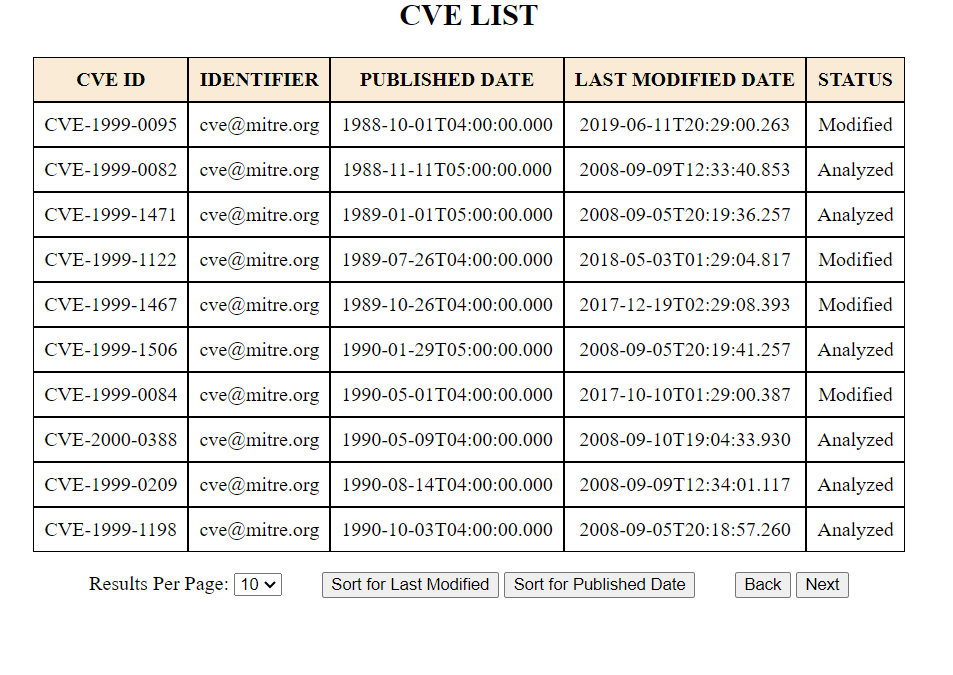
1. **Creating ID.html to show the CVE information:**

This HTML document establishes a structured framework for presenting information pertaining to Common Vulnerabilities and Exposures (CVE). It comprises sections delineated by headings such as "Description," "CVSS V2 Metrics," "Scores," and "CPE," each aimed at encapsulating distinct facets of a CVE entry. Within these sections, placeholders denoted by element IDs await dynamic data injection, likely facilitated by JavaScript, enabling the depiction of crucial CVE details such as severity, scores, and vulnerability criteria. The layout appears conducive to facilitating comprehensive understanding and navigation of CVE-related information, offering a systematic approach to convey pertinent details to users.

1. **Using JavaScript to parse the CVE ID:**

This JavaScript code parses the CVE ID from the URL, fetches corresponding data from the National Vulnerability Database (NVD) API, and renders it dynamically into the HTML elements on the page. The renderData function populates various elements such as the CVE ID, description, severity, score, vector string, CVSS V2 metrics table, exploitability score, impact score, and Common Platform Enumeration (CPE) table with relevant data extracted from the API response. The fetchData function initiates the data retrieval process by making a request to the NVD API using the extracted CVE ID. Once the data is received, it invokes the renderData function to update the HTML content accordingly. This setup enables the display of detailed CVE information based on the provided CVE ID.

Output:



A screenshot of a computer

Description automatically generated