Design Document

Architecture and Component Interaction

The system is developed as three major components in asset scanner, user interface through cli, and centralized server with access to the NVD database. The main components explained:

1. **Asset Scanner**:

- Scans user's device to gather a list of installed applications along with their versions.
- **Functionality**:
    - **System Scan Command**: Users can initiate an on-demand scan from the CLI, which will allow the system to discover assets, gather pertinent information, and send that information to the **NVD Server** so that it can fetch matching vulnerabilities.

2. **CLI Interface**:

- **Objective**: This is a terminal command line application by which the final user interacts with the application instead of a graphical user interface. Users tend to start scans; view threats detected and select patches through a set of commands and options.
- **Key Functionalities**:
    - **Scan Command**: The command initiates the scanning of your assets, prompting the **Asset Scanner** to identify all installed software and packages.
    - **Display Vulnerabilities**: The CLI displays detected vulnerabilities after scanning. Each vulnerability is shown in a list format, which includes:
        - Severity (via CVSS scores)
        - Description and prerequisites for patching
        - Any dependencies

- **Identify Vulnerabilities to Patch**: Users can browse vulnerabilities in the list using commands specific to select specific, individual ones that shall be patched.

3. **NVD Server**:
   - **Role**: A backend server that acts like a repository of vulnerability data, where detected assets are matched with the vulnerabilities of the National Vulnerability Database.
   - **Components**:
     - **Vulnerability Database**: All details for each vulnerability, such as affected versions, CVSS scores, and recommended fixes are included in this.
       -
       Query Interface: This is an API that processes requests from the CLI to fetch vulnerability data for assets identified by the Asset Scanner.
     - **Automatic Daily Updates Provision**: The process is automation-based and keeps on synchronizing the database with the latest data that exists in NVD, thus making the system up-to-date regarding the latest vulnerability information.

Data Flow

1. The **Asset Scanner** can relay a list of detected assets to the **CLI** when the scan is complete.
2. The **CLI** makes a query to the **NVD Server** to retrieve information on vulnerabilities for each of the assets.
3. The **NVD Server** responds back with the information pertaining to vulnerability that is then printed out on the CLI and is viewed by the users.
4. The user must choose the vulnerable applications for which they would like to apply the patches.
5. The CLI then directs the user to the page(s) where the patch can be downloaded.

- **NVD Server to Vulnerability Database**: Intra-node database query to send back data regarding the presence of the existence of vulnerabilities, updated daily from the NVD.
- **Asset Scanner to CLI**: Realtime communication/data sharing where the CLI shows the asset information scanned by the **Asset Scanner**.

Design Rationale

Why CLI Interface?

The transition into a CLI is very light and flexible, requiring no graphical resources or a web server for its installation. Designing a CLI is suitable for **end users**, who, like IT administrators and technically skilled users, feel comfortable in command-line environments. Installation and usage on different operating systems are simplified, thus fostering scalability and flexibility across different environments.

Security and Privacy

By their nature, CLI-based applications minimize potential attack vectors against web interfaces, because this application is not exposed to the network to as large an extent as usual. All communication between CLI and **NVD Server** may be handled by HTTPS; vulnerability data will be locally stored only for a short time before being deleted when the CLI session is terminated.

Update Mechanism

It is essential to maintain the latest version of the **Vulnerability Database** in this system. The auto-update mechanism at the **NVD Server** assists in lessening the maintenance task, so the users are scanning against the latest possible data without any kind of intervention from the user's end.

User-Centric Design Choices

The CLI is structured in such a clear manner that commands like `scan`, `list-vulnerabilities`, and `patch-select` are intuitive, as follows:

- The process of the **Scan Command** is made easy and in no way needs actual user input. The whole procedure goes in well with that of IT security

workflows.

- The **Display Vulnerabilities** feature provides detailed exposure of the vulnerabilities. Then, it will be allowed to analyse the risk created by the vulnerability according to its severity and dependencies.
- The **Patch Selection** process is interactive order wise and very simple that users can confirm or amend their choices before applying those patches.

Scalability and Maintenance

This design is scalable due to the modular nature of the CLI and its interaction with the **NVD Server**. If the database of vulnerabilities grows in size, there is no negative impact on the CLI experience since the **NVD Server** can handle it. The command-line environment also requires very few resources so is easy to maintain and operate even on resource-constrained systems.

Patch Retrieval Mechanism

Currently, the utility would direct the users to webpages or sites for patch downloads, instead of performing the patch download by itself.

This is mainly due to i**nconsistency in patch hosting -** different vendors and software publishers maintain unique approaches to hosting patches and updates. Some may provide centralized download portals, while others rely on individual product pages, making it challenging to determine the exact patch location programmatically.

Due to this, identifying a standardized mechanism for patch download proved much harder than anticipated.

Manually directing users to the official pages was seen as **the most viable and practical** solution to this issue.

Applying Patches to Vulnerable Applications

Applying patches consistently across diverse applications presents a significant challenge due to the varied update mechanisms employed by different types of software. Some applications, particularly CLI tools or system utilities, may support direct command-line patch installations, while others require manual downloads of update files from vendor websites. Additionally, certain applications only support

in-app updates, necessitating the program to be opened and updated interactively, often requiring user authentication or specific permissions. This diversity complicates the process of building a single automated solution for patch application within the system, as it must accommodate multiple update paths and potentially user-specific settings.

In designing the patch management feature, this complexity must be considered carefully. A flexible approach would likely be needed, allowing the system to guide users on specific actions for each type of application. Alternatively, leveraging patch scripts for CLI-compatible applications and offering guidance for those requiring in-app updates can provide a balance, though this approach requires more user involvement and maintenance. We have settled on providing the user guidance on updating the application for each patch through appropriate websites rather than applying it all ourselves, since it not always possible.