**TRACE**

**Software Requirements Specification**

**Version 1.1**

**1/24/2025**

**Document Control**

**Approval**

The Guidance Team and the customer will approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 0.1 |
| Current Release: | 1.1 |
| Indicator of Last Page in Document: | Ω |
| Date of Last Review: | 1/24/25 |
| Date of Next Review: | N/A |
| Target Date for Next Update: | N/A |

**Distribution List**

The following list of people will receive a copy of this document every time a new version of this document becomes available:

**Guidance Team Members:**  
Dr. Oscar A Mondragon – Manager  
Juan Puebla – TA  
Sofia Holguin – TA  
Zachary Wittmann – IA

**Customer:**Sebastian Quinones  
Diana Ramirez  
Timothy Hannan  
Lucia Rodriguez-Melgoza  
Miriam Olague

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 0.1 |  | Guidance team | Create main section and section 1 content |
| 0.2 |  | Guidance team | Create section 2 and UC diagram |
| 0.3 |  | Guidance team | Real World Requirements and Class diagram |
| 0.4 |  | Guidance team | User interfaces |
| 0.5 |  | Guidance team | Behavioral Requirements and STD |
| 0.6 | 7/18/2024 | Guidance team | Data Flow Diagram and Data Dictionary |
| 0.7 | 7/20/2024 | Guidance team | Consistency checking among models |
| 0.8 | 1/14/2025 | Guidance team | Added FilterOptions to data dictionary and fixed Related Real-world Objects names to match class diagram names |
| 1.0 | 1/21/2025 | DAC | Added Additional requirements for the Artificial Intelligence functionality. |
| 1.1 | 1/24/2025 | DAC | Added Specific version numbers for General Constraints 3,4,5 (Python and Svelte/Sveltekit, Neo4J |

**Table of Contents**

[1. Introduction 5](#_Toc187709455)

[1.1. Purpose and Intended Audience 5](#_Toc187709456)

[1.2. Scope of the Product 5](#_Toc187709457)

[1.3. Definitions, Acronyms, and Abbreviations 6](#_Toc187709458)

[1.3.1. Definitions 6](#_Toc187709459)

[1.3.2. Acronyms 6](#_Toc187709460)

[1.3.3. Abbreviations 7](#_Toc187709461)

[1.4. Overview 7](#_Toc187709462)

[1.5. References 8](#_Toc187709463)

[2. General Description 9](#_Toc187709464)

[2.1. Product Perspective 9](#_Toc187709465)

[2.2. Product Features 9](#_Toc187709466)

[2.2.1. List of Actors with Descriptions 9](#_Toc187709467)

[2.2.2. List of Use Cases with Descriptions 10](#_Toc187709468)

[2.2.2.1. Level 1 Use Cases 10](#_Toc187709469)

[2.2.2.2. Level 2 Use Case 10](#_Toc187709470)

[2.2.3. Use Case Diagrams 12](#_Toc187709471)

[2.3. User Characteristics 15](#_Toc187709472)

[2.4. General Constraints 15](#_Toc187709473)

[2.5. Assumptions and Dependencies 16](#_Toc187709474)

[2.5.1. Assumptions: 16](#_Toc187709475)

[2.5.2. Dependencies: 16](#_Toc187709476)

[3. Specific Requirements 17](#_Toc187709477)

[3.1. External Interface Requirements 17](#_Toc187709478)

[3.1.1. User Interfaces 17](#_Toc187709479)

[3.1.2. Hardware Interfaces 59](#_Toc187709480)

[3.1.3. Software Interfaces 59](#_Toc187709481)

[3.1.4. Communications Interfaces 59](#_Toc187709482)

[3.2. Behavioral Requirements 60](#_Toc187709483)

[3.2.1. Same Class of User 60](#_Toc187709484)

[3.2.2. Related Real-world Objects 60](#_Toc187709485)

[3.2.2.1. Logger Class 63](#_Toc187709486)

[3.2.2.2. Machine Service Manager Class 64](#_Toc187709487)

[3.2.2.3. Machine Manager Class 65](#_Toc187709488)

[3.2.2.4. Project Manager Class 66](#_Toc187709489)

[3.2.2.5. Project Class 67](#_Toc187709490)

[3.2.2.6. HTTP Client Class 68](#_Toc187709491)

[3.2.2.7. Proxy Server Class 68](#_Toc187709492)

[3.2.2.8. DB Enumerator Class 69](#_Toc187709493)

[3.2.2.9. AI Wordlist Manager Class 70](#_Toc187709494)

[3.2.2.10. Directory Brute Forcer Class 71](#_Toc187709495)

[3.2.2.11. Crawler Class 72](#_Toc187709496)

[3.2.2.12. Directory Tree Creator Class 73](#_Toc187709497)

[3.2.2.13. Authentication Attacker Class 73](#_Toc187709498)

[3.2.2.14. Request Manager Class 75](#_Toc187709499)

[3.2.2.15. SQL Injection Manager Class 75](#_Toc187709500)

[3.2.3. Stimulus 76](#_Toc187709501)

[3.2.3.1. User Interfaces 78](#_Toc187709502)

[3.2.4. State Transitions 86](#_Toc187709503)

[3.3. Non-behavioral Requirements / Quality Attributes 90](#_Toc187709504)

[3.3.1. Performance Requirements 90](#_Toc187709505)

[3.3.2. Availability 91](#_Toc187709506)

[3.3.3. Security 91](#_Toc187709507)

[3.3.4. Maintainability 92](#_Toc187709508)

[3.3.5. Portability 92](#_Toc187709509)

[3.3.6. Design and Implementation Constraints 92](#_Toc187709510)

[3.4. Other Requirements 93](#_Toc187709511)

[3.4.1. Data Flow 93](#_Toc187709512)

[3.4.1.1. Context Diagram (AKA DFD Level 0) 93](#_Toc187709513)

[3.4.1.2. Data flow diagrams 95](#_Toc187709514)

[3.4.1.3. Data Dictionary 101](#_Toc187709515)

[3.4.2. Operations 111](#_Toc187709516)

[3.4.3. CAT 1 STIG Vulnerabilities 111](#_Toc187709517)

# **Introduction**

# **Purpose and Intended Audience**

The Software Requirements Specification (SRS) aims to provide the analyst with a detailed description of how to use the software created for the Targeted Reconnaissance for Advanced Content Exploitation (TRACE) system. The Data Analysis Center’s (DAC) Analyst (the analyst from now on) is the main user of the TRACE system. The software system allows the analyst to gather information from various enterprise testing tools and use shallow learning search algorithms to select brute force lists and generate username and password lists that are more likely to succeed within a given application. The analyst can select the tool they need and use it on a specified list of IP addresses to generate a tree graph representation of the system. TRACE facilitates the execution of penetration testing for DAC Analysts. This SRS is intended for analysts who seek to benefit from using our tool in the field when conducting penetration tests, the designers working on this tool's creation, and all other project stakeholders.

# **Scope of the Product**

The Department of Defense (DoD) must ensure that the technological systems they are delivering have secure and resilient capabilities. To ensure these capabilities, and the survivability and lethality of the combat systems used, it is critical for them to perform security testing of these systems. During this phase of testing, the analyst, acting as a penetration tester, is responsible for the discovery of vulnerabilities and realizing the exploitation tied to the vulnerabilities. Then, analysts generate a report describing the risks of each vulnerability. The product being developed, named Targeted Reconnaissance for Advanced Content Exploitation or TRACE, will be used to facilitate and expedite penetration testing for the analysts.

The Cyber Experimentation & Analysis Division (CEAD) needs a tool to aid the analysts in the penetration tests by automating steps that would normally be done by hand. To aid in this, the System of Interest (SOI), TRACE, provides the following functionality:

1. Select the desired tool, the desired tool may be changed later according to the needs of the analyst.
2. Scan allowed IPs.
3. Use the AI component of the SOI to generate usernames and passwords from the data found within the allowed IPs.
4. Allow the analyst to select a default wordlist
5. Display the current progress of the penetration test to the analyst.
6. Display the system as a tree for easier visualization as the system is being scanned.
7. Display to the analyst the remaining IPs that need to be scanned.
8. Generate a file to export the compiled results of the penetration test.

TRACE does not do the following:

1. Perform the penetration testing

TRACE benefits include:

1. Provide the analysts with the required tools to generate possible lists of usernames and passwords.
2. Provides a report of the usernames and passwords found.

# **Definitions, Acronyms, and Abbreviations**

# **Definitions**

**Table 1.3.1-1: Definitions**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| Machines | The list of machines that will undergo analysis by the analyst with the use of the TRACE tool. |

# **Acronyms**

This section lists the acronyms used in this document and their associated definitions.

**Table 1.3.2-1: Acronyms**

|  |  |
| --- | --- |
| TRACE | Targeted Reconnaissance for Advanced Content Exploitation |
| SRS | Software Requirements Specification |
| SOI | System of Interest |
| DAC | Data Analysis Center |
| CEAD | Cyber Experimentation & Analysis Division |
| STD | State Transition Diagram |
| DFD | Data Flow Dictionary |
| DD | Data Dictionary |
| DEVCOM | U.S. Army Combat Capabilities Development Command |
| DoD | Department of Defense |
| IP | Internet Protocol |
| Recon | Reconnaissance |
| CVPA | Cooperative Vulnerability and Penetration Assessment |
| STIG | Security Technical Implementation Guide |

# **Abbreviations**

**Table 1.3.3-1: Abbreviations**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| e.g. | For example |
| i.e. | That is |
| UI | User interface - The part of the system with which the user interacts with directly |
| DB | Database - An organized collection of data |

# **Overview**

The Software Requirements Specification (SRS) document provides the analysts and any other stakeholders with a detailed description of the TRACE system, particularly the system’s intended functionality, and the mechanisms of user interaction. The SRS document has a predefined outline. A brief synopsis of what each section covers follows (please note that the numbering corresponds to the section number in the SRS):

1. **Introduction:** Briefly introduce the system to be developed, including its purpose and scope. The main terms, acronyms, and abbreviations of the SRS are further defined.
   1. **Purpose and Intended Audience**: Highlights the purpose and audience of the SRS.
   2. **Scope of Product**: Describes the purpose, goals, benefits, and background of the system.
   3. **Definitions, Acronyms, and Abbreviations**: Defines terms, acronyms, and abbreviations used in the report.
2. **General Description**: Provides an overall view of the system.
   1. The **Product Perspective** section describes how the system fits into the company’s overall structure and its relationship with other systems.
   2. The **Product Features** provides a brief description of all the system’s main features. This section contains the Use Case Diagram, a description of Actors, and Use Cases that define the user's interactions with the system.
   3. The **User Characteristics** section describes the intended users of the system and any characteristics that may influence the system design.
   4. The **General Constraints** section identifies any limitations that may impact the development of the software.
   5. The **Assumptions and Dependencies** section states any assumptions made during the development of the requirements and identifies external factors that may affect the system.
3. **Specific Requirements**: Delves into the detailed technical requirements, such as external interfaces, behavioral, and non-behavioral that allow the design of the system to satisfy all the conditions described in the Requirements Definition Document (RDD).
   1. **External Interface Requirements**: Describes the interface between the system and external entities, such as users, hardware, or other systems.
   2. **Behavioral Requirements**: Specifies the expected behavior of the system under various circumstances.
   3. **Non-Behavioral Requirements**: Covers requirements for the quality attributes of the system, such as performance, usability, and security requirements.
   4. **Other Requirements**: Describes requirements mostly pertaining to data that flows through the system and how it is transformed.

# **References**

“Application Security and Development Security Technical Implementation Guide,” STIG Viewer | Unified Compliance Framework. [Online]. Available: <https://www.stigviewer.com/stig/application_security_and_development/>. [Accessed May 31, 2024].

“Kali docs: Kali Linux Documentation,” *Kali Linux*. [Online]. <https://www.kali.org/docs/> [Accessed May. 30, 2024].

# **General Description**

# **Product Perspective**

TRACE is a cybersecurity tool designed to analyze the data within specified IP addresses to determine any possible usernames and passwords that could be used to access the system using an AI shallow learning algorithm. TRACE is an independent, self-contained software system designed to aid analysts when performing penetration testing. It shall offer a user-friendly interface to help the analyst generate the required usernames and passwords, as well as improve the visualization of the system. The system shall leverage Shallow Learning Artificial Intelligence algorithms to generate the usernames and passwords. TRACE will aid the analysts in expediting necessary steps and will be an essential tool to add to the company’s toolset.

# **Product Features**

The main features of this SOI include the following functionalities: Managing a project, configuring the services to be utilized, performing analysis on machine(s), and logging the actions of the system. These features and more are shown in the Use Case Diagram – Level 1 see figure 2.2.3-1. The functionalities shown within Figure 2.2.3-1 will be expanded upon by describing them in two Level 2 Use Case Diagrams.

# **List of Actors with Descriptions**

The main actors of the TRACE system are the following:

|  |  |
| --- | --- |
| Actor | Description |
| **Analyst** | A cyber analyst working for the Data Analysis Center’s Cyber Experimentation & Analysis Division (DAC CEAD), who operates the TRACE system. The analyst can use the TRACE system to import a project, perform the scan, display the scan progress, select the machines to be analyzed, and export the results. Each analyst will have full access to the system. |
| **Lead Analyst** | An Analyst that is assigned to lead a project. The lead analyst should be able to use the TRACE system to create, delete, and lock a project in addition to all the Analyst’s functionality. |
| **DEVCOM Analyst** | An abstract actor represents when both users, the lead analyst and the analyst, have the ability to use a system feature. |
| **Machines** | The list of IP addresses with defined port number that TRACE will be able to analyze. |

# **List of Use Cases with Descriptions**

# **Level 1 Use Cases**

Use cases are grouped by levels to cope with the complexity of the TRACE system. The Use Case Level 1 includes abstract use cases identified by separations of concerns. The following two Use Cases belong to Level 1 and are further decomposed into a Level 2 Diagram.

|  |  |
| --- | --- |
| Use Case | Description |
| **Manage Project** | The lead analyst can configure a project by creating, deleting, and locking a project. The lead analyst and analyst are TRACE users and able to import, open, and export a project and input Nmap information. |
| **Perform Analysis** | The analyst can select a service to use on the specified machines, view the progress of the analysis, view the log, and filter the results |

# **Level 2 Use Case**

Level 2 Use Cases for TRACE are organized by separation of concerns identified in level 1.

**Manage Project:**

|  |  |
| --- | --- |
| Use Case | Description |
| **Create Project** | The lead analyst can create a project by adding the necessary information and generating project folders to store the project information. |
| **Import Project** | The analyst can import an existing project to continue its progress. Upon importing a project, the analyst becomes the lead analyst. |
| **Delete Project** | The lead analyst can delete a project and all the information relating to the project if the project is not locked. |
| **Lock Project** | The analyst can lock a project to prevent the project’s information from being deleted. |
| **Import Nmap Results** | The analyst can import Nmap information to identify the machines on which the services are executed. |
| **Open Project** | The analyst can open an existing project to continue the working on the project. |
| **Export Project** | The analyst can export a project. |

**Perform Analysis:**

|  |  |
| --- | --- |
| Use Case | Description |
| **Manage Directory Brute Force** | *The analyst can select a word list before executing directory brute force service on given http root. The results are displayed to the analyst.* |
| **Manage Crawling** | *The analyst provides a HTTP address and executes the web crawler service to discover web pages and resources. The results are displayed to the analyst.* |
| **Build Tree**  <<include>> | *This service builds a tree**using the information from the crawling and directory brute-force processes. The tree provides a visual structure of the websites and directories being analyzed. Build tree is an include use case used by manage crawling and manage directory brute-force.* |
| **Change HTTP Requests** | *The analyst can edit previous HTTP requests before they are sent again to a machine.* |
| **Resend Requests** | *The analyst can resend selected HTTP requests previously sent to a machine.* |
| **Create HTTP Client** | *The analyst can set up an HTTP client to support managing HTTP requests and resend HTTP requests to the machines being analyzed.* |
| **Establish Proxy Server** <<include>> | *After creating the HTTP client, TRACE should establish a proxy server to handle the requests and packets from the Change Request, Resend Request, and SQL injection processes that are sent to and received from the machines. This service is an include use case used by the Create HTTP Client UC.* |
| **Perform SQL Injection** | *The analyst identifies injection vulnerability points and provides payloads for SQL queries sent to the target machine.* |
| **DB Enumeration** | *The analyst can execute this service to retrieve information about the database structure(s) found in the targeted machine.* |
| **AI Generate Usernames and Passwords** | *The analyst can select TRACE services to be included before executing this service which makes use of the information gathered from the selected services executed on the machines. Note: this service makes use of a component provided by the customer that makes use of artificial intelligence technology.* |
| **Display Machines Analysis Progress** | *The analyst can visualize the progress of any TRACE service which can be Idle, in progress, completed. Note: the AI names and password service can only be executed if the crawler service has been executed.* |
| **Display Machine Analysis Progress (Services Progress):** | *The analyst shall be able to view the progress of an individual service as it is being performed.* |
| **View Log** | *The analyst shall view the logs of the results and actions taken during the analysis of the machines.* |
| **Filter Results** | *The analyst shall have the option to filter the logs based on criteria such as response codes or content length.* |
| **Select Machine** | *The analyst shall be able to select a machine to run the services on.* |
| **Pause** | *The analyst shall be able to pause an ongoing service.* |
| **Resume** | *The analyst shall be able to resume a paused service.* |
| **Cancel** | *The analyst shall be able to cancel a service.* |

# **Use Case Diagrams**

The following section details the use case diagrams for TRACE. Table 2.2.3-1 provides a legend for the use case diagrams in this section.

Table 2.2.3-1: Use Case Diagram Legend

|  |  |  |
| --- | --- | --- |
| **NAME** | **NOTATION** | **DESCRIPTION** |
| System |  | The rectangle represents the boundary of the system. It separates the system from its external environment. |
| Use Case |  | Service provided by the system. |
| Actor | A stick figure with a circle on it  Description automatically generated | An external entity that interacts with the system via the use cases. Role of the users of the system. |
| Association | A diagram of a person using a device  Description automatically generated | The actor participates in the Use Case. |
| Include Relationship | A diagram of a diagram  Description automatically generated | The **Base** use case includes the **Included** use case. The **Base** use case requires the behavior of the **Included** use case to provide its functionality. |
| Extend Relationship | A close-up of a circle  Description automatically generated | The **Base** use case extends the **Extending** use case. The **Base** use case can optionally use the behavior of the **Extending** use case. |

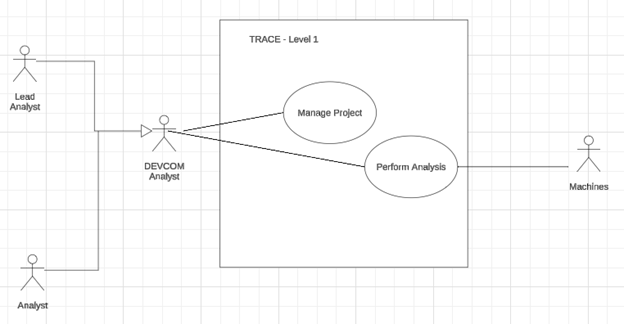


Figure 2.2.3-1. TRACE: Level 1 Use Case Diagram

A diagram of a project

Description automatically generated

Figure 2.2.3-2 Level 2 Configure Project

A diagram of a diagram

Description automatically generated with medium confidence

Figure 2.2.3-3. Level 2 Perform Analysis

# **User Characteristics**

This section identifies the users of the system, the nature of their use of the system, and their characteristics.

UC1. Users shall be authorized members of the DEVCOM Analytic Center who are granted access to the system via login credentials.  
UC2. The users may have disabilities that prevent them from using the system   
UC3. There shall be accommodations such as color blindness to aid the analyst in using the tool better.  
UC4. The users shall be expected to have some experience with penetration testing.

# **General Constraints**

GC1. The system shall be a web-based application.  
GC2. The system shall be capable of running smoothly on a machine with the specifications provided on the RDD: 8GB Ram, 4-Core 2.5Ghz, and 10GB of available storage.  
GC3. The backend of the system shall be developed using Python 3.14.  
GC4. The frontend of the system shall be developed using SvelteKit v.2.16.1 and must be compatible with Svelt v5.19.2. GUI Components shall leverage the library: shadcn-svelte 0.14.0

GC5. The database of the system shall be developed using the Neo4j 5.26.1.  
GC6. The system shall not utilize docker.

# **Assumptions and Dependencies**

# **Assumptions:**

A1. The analyst, the user of the TRACE, is expected to have experience using similar tools, executing TRACE services, and interpreting the results for their gain.

A2. TRACE should have access to the intranet to reach the intended machines.

# **Dependencies:**

D1. On or Before 01/22/2025: DAC will provide a Git Repository in the form of a zip file that will contain the following:  
 a. A README that provides information related to the AI algorithm leveraged in trace  
 b. Source code of the AI algorithm  
 c. Requirements.txt file for dependency installation  
 d. A sample wordlist that will be leveraged during the development in SW2.  
 e. A site\_list.csv that is intended to be an example (although it can be overwritten) of how the algorithm will scrape data from webpages   
 f. a folder labelled "llm\_pass\_eval" which provides a script that leverages Gemini in order to obtain feedback regarding the results of your code and instructions on how to connect.

# **Specific Requirements**

# **External Interface Requirements**

This section contains the specification of requirements for interfaces among different components and their external capabilities, including all its users, both human and other systems. The characteristics of interfaces to the SOI are included. Each interface may represent a bi-directional flow of orientation.

# **User Interfaces**

This section addresses the main user interfaces for TRACE and their corresponding GUI components.

**NOTE: The following user interfaces are only potential options that may be considered for development, and these user interfaces are not required.**

1. **TRACE Landing Page**

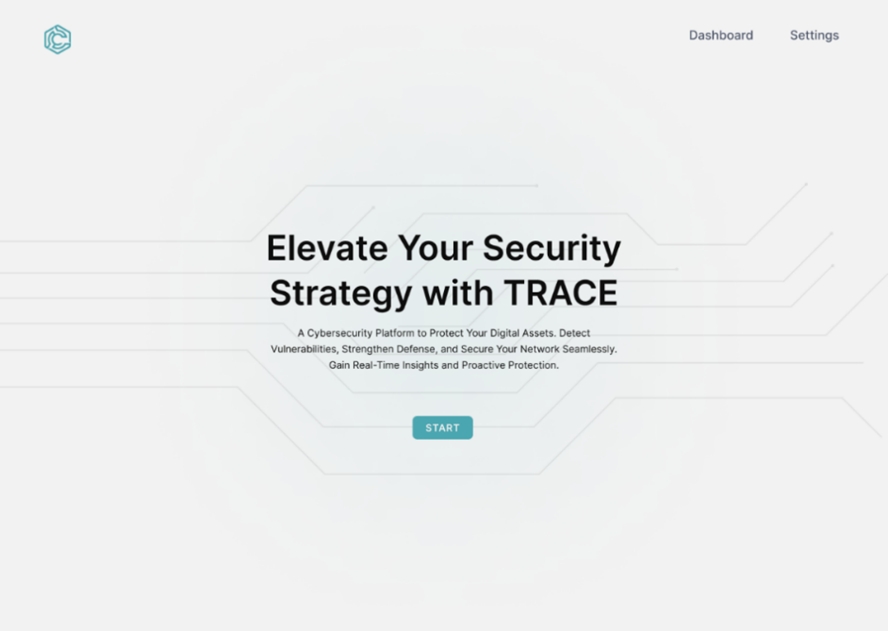


Figure 3.1.1-1 TRACE Landing Page

**Logical Characteristic**

**[SRS 1]** The Landing Page shall provide an introductory interface for TRACE, welcoming users to the system. It shall include the following components (as shown in Figure 3.1.1-1):

1. A TRACE logo displayed in the upper-left corner of the screen.
2. A navigation bar in the upper-right corner with the following options:
   1. A button labeled "Dashboard."
   2. A button labeled "Settings."
3. A header: “Elevate Your Security Strategy with TRACE”.
4. Supporting text under the header: "A Cybersecurity Platform to Protect Your Digital Assets. Detect Vulnerabilities, Strengthen Defense, and Secure Your Network Seamlessly. Gain Real-Time Insights and Proactive Protection."
5. A button labeled "Start" is positioned in the center of the screen underneath the supporting text.

**Optimized Aspects**

**[SRS 1.1]** The system shall display the "Start" button at the center of the screen with sufficient size and contrast to ensure visibility and guide user interaction.

**[SRS 1.2]** The system shall provide hover text for all interactive elements in sentence case and ensure it is readable and informative ("Click to start a session").

**[SRS 1.3]** The system shall ensure the TRACE logo remains clickable and redirects to the dashboard. If a project session is active, the system shall prompt the user to confirm leaving the session with a message.

1. **TRACE Project Dashboard – Side Bar**



Figure 3.1.1-2 Project Dashboard Side Bar

**Logical Characteristic**

**[SRS 2]** The Sidebar shall provide a navigation menu for users to quickly access key sections of the TRACE application. It shall include the following components (as shown in figure 3.1.1-2):

1. A TRACE logo button positioned at the top.
2. A folder icon button called “Project Selection”.
3. A folder directory button called “Project Folders”.
4. A trash bin icon button called “Deleted Projects”.
5. A gear icon button called “Settings”.

**Optimized Aspects**

**[SRS 2.1]** The system shall provide descriptive hover text for each navigation icon in sentence case, such as "Access project folders" or "Open archived projects."

**[SRS 2.2]** The system shall visually highlight the currently selected icon with a distinct color (blue) or border to clearly indicate the active section.

**[SRS 2.3]** The system shall maintain consistent spacing between icons to ensure uniformity and ease of navigation.

1. **TRACE Project Dashboard – My Projects**

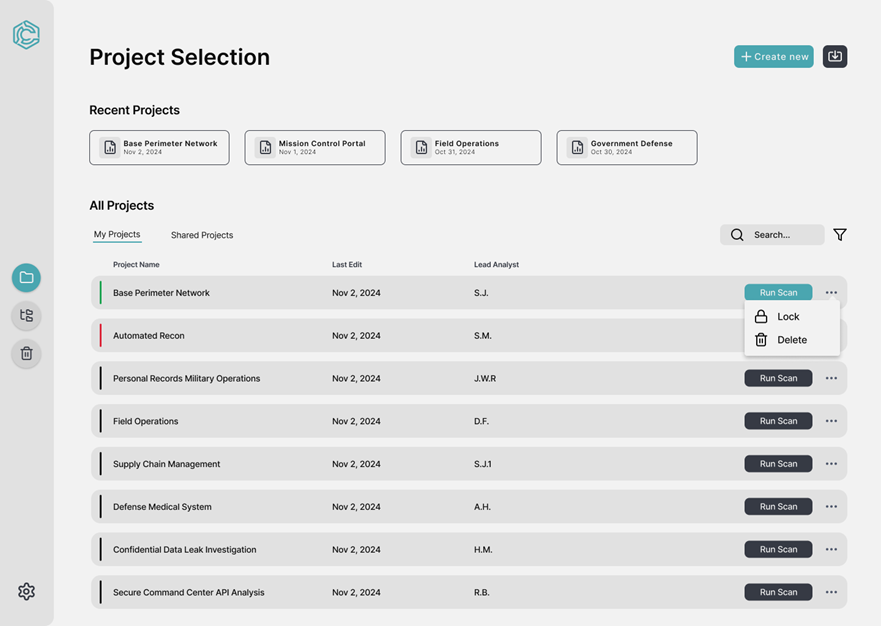


Figure 3.1.1-3 My Projects

**Logical Characteristic**

**[SRS 3]** The Project Dashboard shall provide users with an overview of their projects and access to tools for managing them. It shall include the following components (as shown in Figure 3.1.1-3):

1. Section title labeled as “Project Selection”.
2. Section subtitle labeled as "Recent Projects” displaying project cards. Each card shall include:
   1. Project Name and Last Edit Date.
3. Section subtitle labeled as “All Projects”.
4. A tab interface with two tabs labeled “My Projects” and “Shared Projects”. Where “My Projects” displays project cards with the following information:
   1. Project Name, Last Edit Date, and Lead Analyst Initials.
   2. A button labeled “Run Scan”.
   3. An ellipsis with additional options including two buttons labeled “Lock” and “Restore”.
5. A search and filter bar at the top right corner of the project list.
6. A button labeled “Create New” at the top right corner of the page.
7. A button with an import logo to the left of the “Create New” button.
8. A side bar with options (as shown in Figure 3.1.1-2).

**Optimized Aspects**

**[SRS 3.1]** The system shall display project statuses using intuitive color indicators (green for active, red for errors, and gray for inactive).

**[SRS 3.2]** The system shall provide a clear tabbed interface for users to switch seamlessly between "My Projects" and "Shared Projects."

**[SRS 3.3]** The system shall display dynamic updates for search and filter results as the user types or adjusts filters.

**[SRS 3.4]** The system shall gray out the 'Run Scan' button for projects that are inactive or have errors, ensuring users cannot initiate scans on unsuitable projects.

1. **TRACE Project Dashboard – Shared Projects**

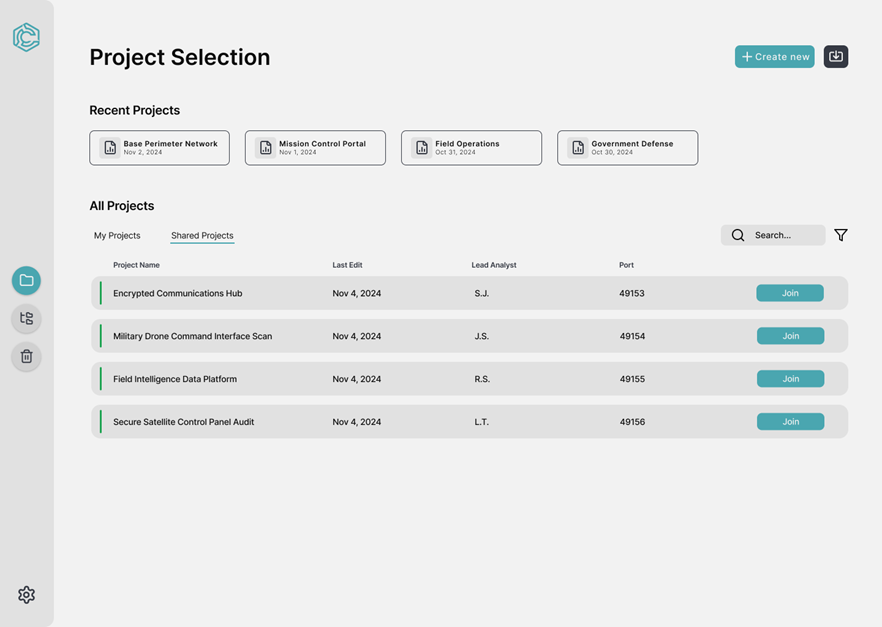


Figure 3.1.1-4 Shared Projects

**Logical Characteristic**

**[SRS 4]** The Shared Projects Page shall provide a view for analysts to join active project sessions hosted on the network. It shall include the following components (as shown in figure 3.1.1-4):

1. Section title labeled as “Project Selection”.
2. Section subtitle labeled as "Recent Projects” displaying project cards. Each card shall include:
   1. Project Name
   2. Last Edit Date
3. Section subtitle labeled as “All Projects”.
4. A tab interface with two tabs labeled “My Projects” and “Shared Projects”. Where “Shared Projects” displays project cards with the following information:
   1. Project Name
   2. Last Edit Date
   3. Lead Analyst Initials
   4. Port Number
   5. A button labeled “Join”
5. A search and filter bar at the top right corner of the project list.
6. A button labeled “Create New” at the top right corner of the page.
7. A button with an import logo to the left of the “Create New” button.
8. A side bar with options (as shown in Figure 3.1.1-2).

**Optimized Aspects**

**[SRS 4.1]** The system shall ensure the "Join" button is visible and active only for projects the user can access.

**[SRS 4.2]** The system shall display hover text for the "Join" button with the message "Join this project session."

1. **TRACE Project Dashboard – Project Folders**

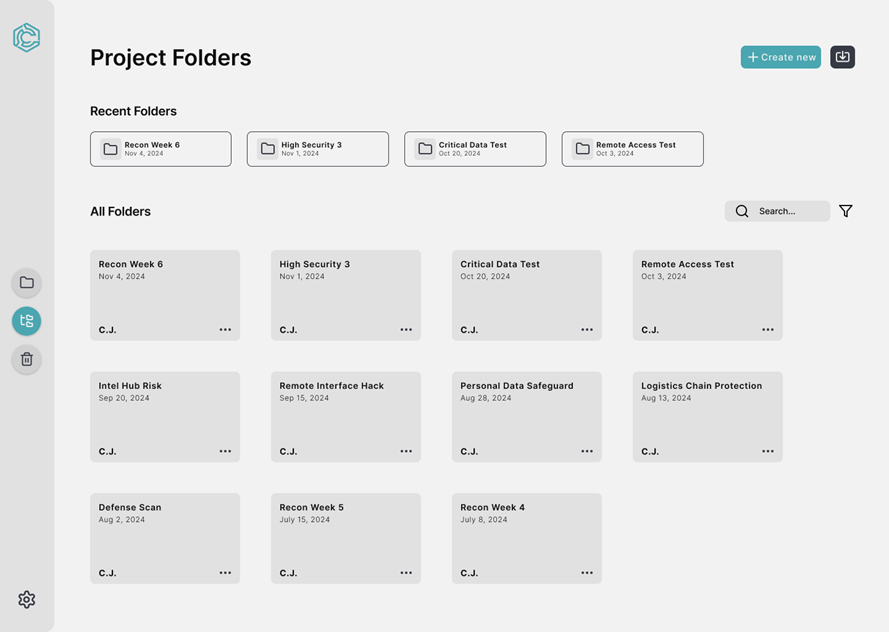


Figure 3.1.1-5 Project Folders

**Logical Characteristic**

**[SRS 5]** The Project Folders Page shall provide a view for users to manage and access their project folders. It shall include the following components (as shown in figure 3.1.1-5):

1. Section title labeled as “Project Folders”.
2. Section subtitle labeled as "Recent Folders” displaying folder cards. Each card shall include:
   1. Folder Name
   2. Created Date
3. Section subtitle labeled as “All Folders”.
4. A search and filter bar at the top right corner of the project list.
5. A grid of folder cards with the following information:
   1. Project Name
   2. Created Date
   3. Lead Analyst Initials
   4. An ellipsis menu for additional options.
6. A button labeled “Create New” at the top right corner of the page.
7. A button with an import logo to the left of the “Create New” button.
8. A side bar with options (as shown in Figure 3.1.1-2).

**Optimized Aspects**

**[SRS 5.1]** The system shall allow users to organize the "All Folders" grid either alphabetically or by the last edited date, based on user preferences.

**[SRS 5.2]** The system shall display hover text for the "More Options" button, providing a clear description of available actions, such as "Rename," "Move," or "Delete."

1. **TRACE Project Dashboard – Deleted Projects**

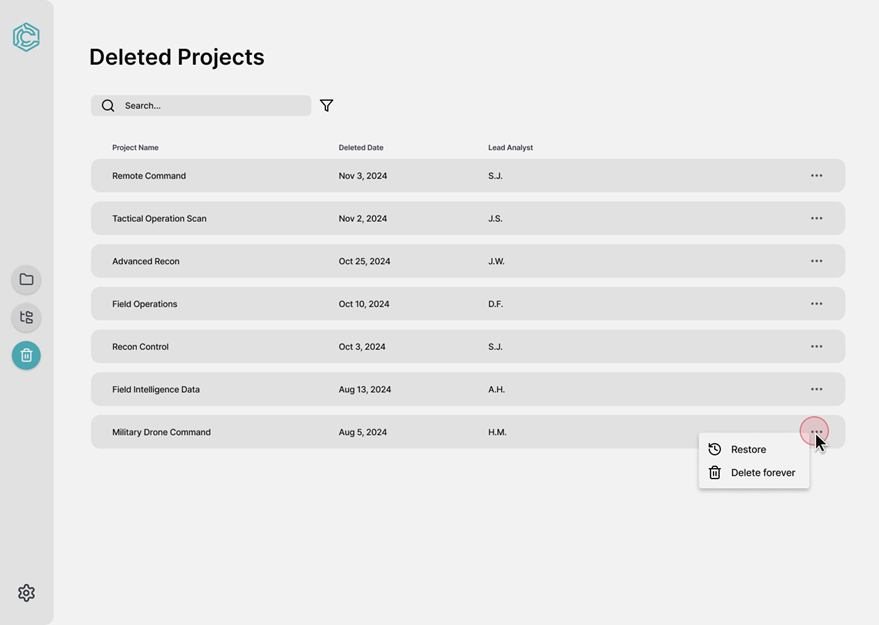


Figure 3.1.1-6 Deleted Projects

**Logical Characteristic**

**[SRS 6]** The Deleted Projects Page shall provide users with a view to manage projects that have been deleted. It shall include the following components (as shown in figure 3.1.1-6):

1. Section title labeled as “Deleted Projects”.
2. A search and filter bar at the top left corner of the project list.
3. Project cards with the following information:
   1. Project Name
   2. Deleted Date
   3. Lead Analyst Initials
   4. An ellipsis with additional options including two buttons labeled “Restore” and “Delete Forever”.
4. A side bar with options (as shown in Figure 3.1.1-2).

**Optimized Aspects**

**[SRS 6.1]** The system shall show hover text for all actions, such as Restore and Delete Forever, to clarify their functionality.

**[SRS 6.2]** The system shall allow sorting of the table columns by Project Name, Deleted Date, or Lead Analyst.

1. **TRACE Project Dashboard – Create Project**

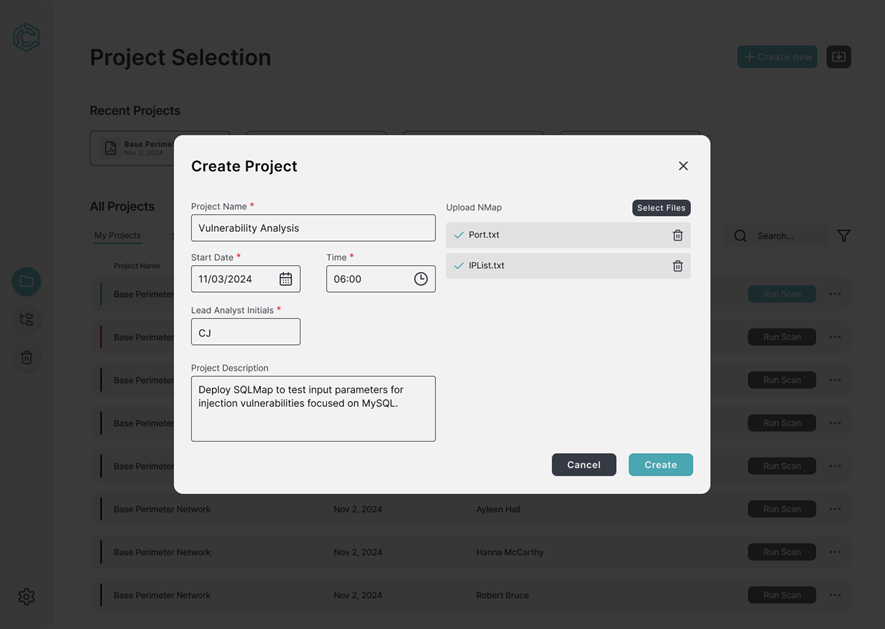


Figure 3.1.1-7 Create Project

**Logical Characteristic**

**[SRS 7]** The Create Project Page shall enable users to configure and initiate a new project. It shall include the following components (as shown in figure 3.1.1-7):

1. A modal pop up with a section title labeled “Create Project”.
2. Input fields for project configuration:
   1. Required:
      1. Project Name
      2. Start Date
      3. Time
      4. Lead Analyst Initials
   2. Optional:
      1. Project Description
      2. File Upload
3. File Upload section with a button labeled “Select Files” and a button labeled as a trach icon next to each uploaded file.
4. Action buttons labeled “Cancel” and “Create”.
5. A close icon at the top right of the modal.

**Optimized Aspects**

**[SRS 7.1]** The system shall provide clear hover text to guide users through the project configuration process.

1. **TRACE Project Dashboard – Import Project**

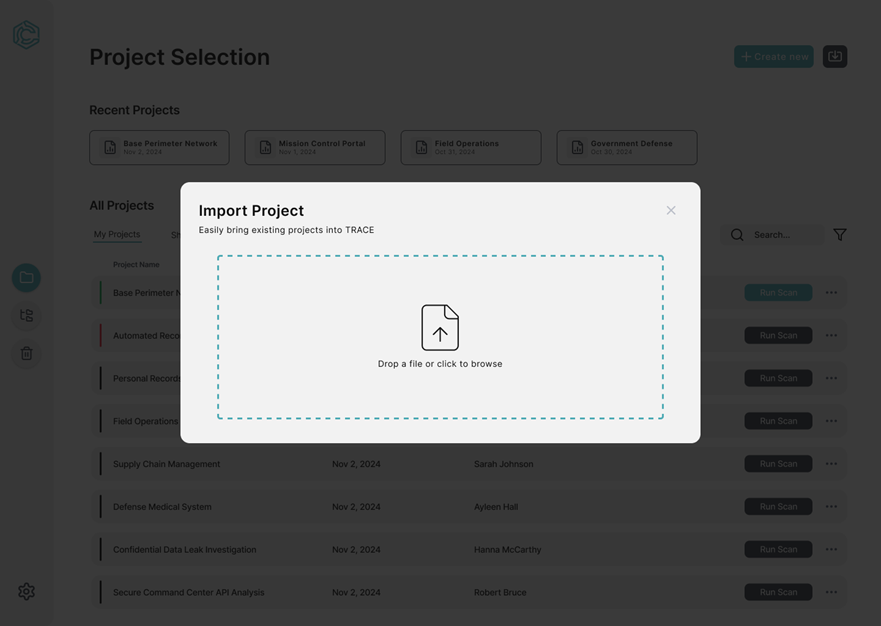


Figure 3.1.1-8 Import Project

**Logical Characteristic**

**[SRS 8]** The Import Project Page shall provide an interface for users to upload existing project files into TRACE. It shall include the following components (as shown in figure 3.1.1-8):

1. A modal pop up with a section title labeled “Import Project”.
2. Description text labeled "Bring existing projects into TRACE."
3. A dotted border box with a file upload icon in the middle. Supporting text labeled “Drop a file or click to browse” is located below the icon.
4. A close icon at the top right of the modal.

**Optimized Aspects**

**[SRS 8.1]** The system shall display hover text or tooltips when users hover over the drag-and-drop area, providing guidance like "Upload your project files here."

1. **TRACE Tool Dashboard – Side Bar**



Figure 3.1.1-9 Tool Dashboard Side Bar

**Logical Characteristic**

**[SRS 9]** The Side Navigation Bar shall provide quick access to key TRACE functionalities. It shall include the following components (as shown in figure 3.1.1-9):

1. A TRACE logo button positioned at the top.
2. A hammer icon button called “Tools Dashboard”.
3. A tree icon button called “Tree Graph”.
4. A file with checkmark icon button called “Results”.
5. A brain icon button called “AI Credential Generator”.
6. A gear icon button called “Settings”.

**Optimized Aspects**

**[SRS 9.1]** The system shall highlight the currently active icon with a distinct color or border to indicate the selected page.

**[SRS 9.2]** The system shall provide hover text with a short description for each icon ("Tools Dashboard" for the hammer icon).

**[SRS 9.3]** The system shall allow users to click on the TRACE logo to navigate back to the Project Dashboard but must prompt the user to confirm before ending the current project session.

1. **TRACE Tool Dashboard - Tools**

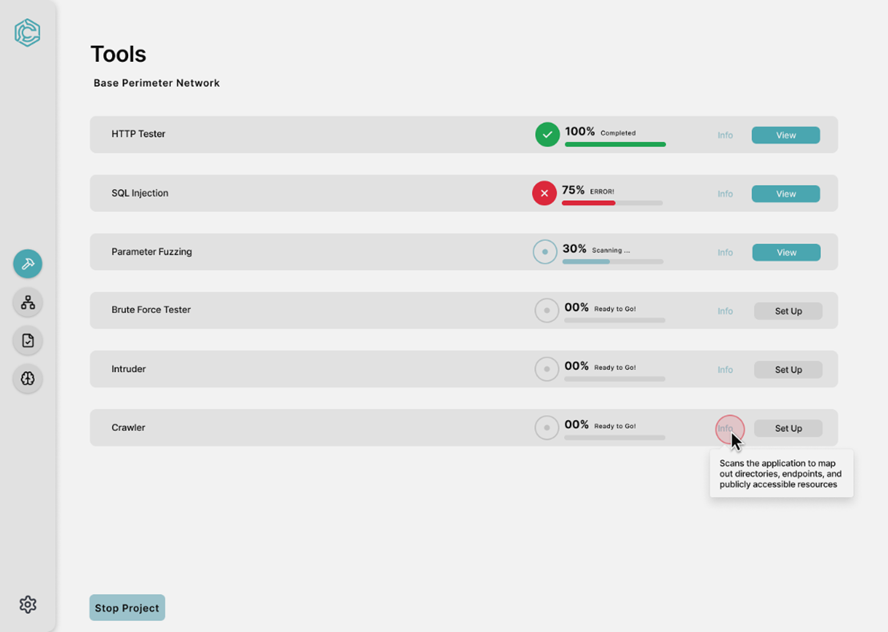


Figure 3.1.1-10 Tool Dashboard

**Logical Characteristic**

**[SRS 10]** The Tools Dashboard Page shall provide an overview of the tools being utilized for the active project session. It shall include the following components (as shown in figure 3.1.1-10):

1. Section title labeled as “Tools”.
2. Project name displayed under the page title, "Base Perimeter Network".
3. A list of tools each represented in a row with the following information:
   1. Tool Name
   2. Progress Bar with a percentage and color corresponding to its status (Scanning, Error, and Complete).
   3. An info button displays a tooltip with a description of the tool “Scans the application to map out directories, endpoints, and publicly accessible resources”.
   4. Action buttons labeled “View” for complete scans and “Set Up” for tools requiring configuration.
4. A button labeled “Stop Project”.
5. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 10.1]** The system shall display the progress bar dynamically, updating in real time to reflect the tool's progress.

**[SRS 10.2]** The system shall provide descriptive tooltips for each tool when the user clicks "Info", explaining the purpose and functionality of the tool.

**[SRS 10.3]** The system shall only enable "View" for tools that have completed or are currently running.

**[SRS 10.4]** The system shall clearly label tools that require setup with "Set Up".

1. **TRACE Tool Dashboard – HTTP Configuration**

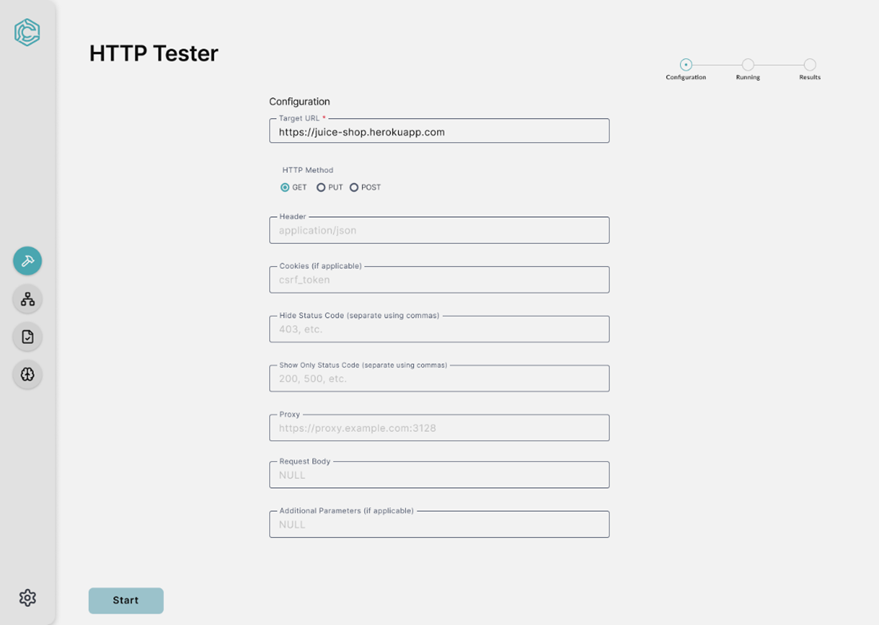


Figure 3.1.1-11 HTTP Configuration

**Logical Characteristic**

**[SRS 11]** The HTTP Tester Configuration Page shall allow users to set up and execute HTTP requests to analyze server responses. It shall include the following components (as shown in figure 3.1.1-11):

1. Section title labeled as “HTTP Tester”.
2. A sub text labeled “Configuration”.
3. A list of input boxes labeled (Target URL, Header, Cookies, Hide Status Code, Proxy, Request Body, and Additional Parameters). Each input box has a placeholder text.
4. Radio buttons labeled “GET”, “PUT”, and “POST” underneath a sub text labeled “HTTP Method”.
5. A visual progress bar with three stages: “Configuration”, “Running”, and “Results” with the current stage highlighted.
6. A button labeled “Start” at the bottom.
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 11.1]** The system shall validate the input boxes before allowing the user to proceed and display an error if an input is incorrect.

**[SRS 11.2]** The system shall provide real-time hover text for each field, explaining its purpose.

**[SRS 11.3]** The system shall highlight incomplete or invalid fields in red to prompt user correction.

1. **TRACE Tool Dashboard – HTTP Running**

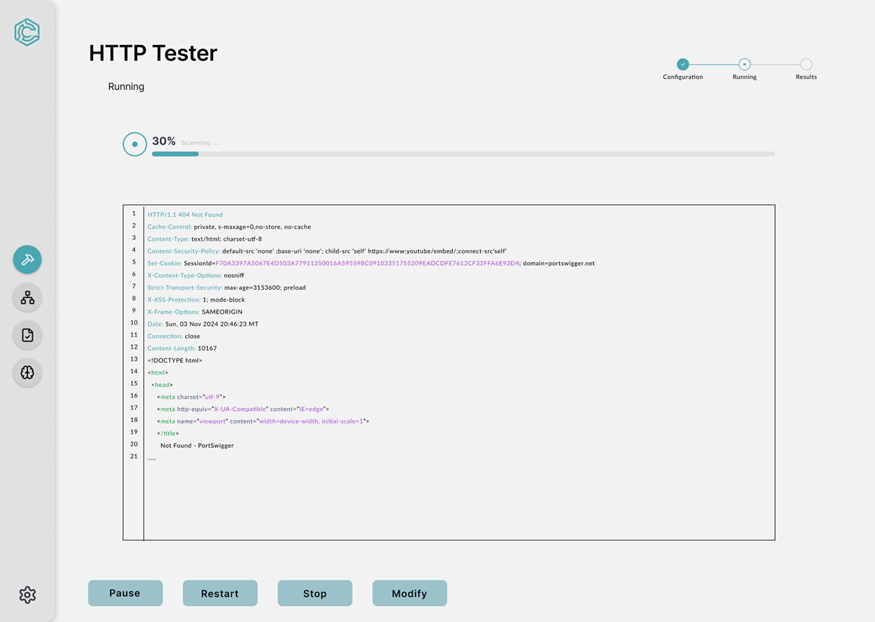


Figure 3.1.1-12 HTTP Running

**Logical Characteristic**

**[SRS 12]** The HTTP Tester Running Phase shall display real-time feedback on the progress and response of HTTP requests during execution. It shall include the following components (as shown in figure 3.1.1-12):

1. Section title labeled as “HTTP Tester”.
2. A sub text labeled “Running”.
3. A visual progress indicator showing the percentage completion of the scan (30% Scanning…).
4. A scrollable output box displaying the server response to the current HTTP request.
5. A visual progress bar with three stages: “Configuration”, “Running”, and “Results” with the current stage highlighted.
6. Buttons located at the bottom of the page including:
   1. A button labeled “Pause”.
   2. A button labeled “Restart”.
   3. A button labeled “Stop”.
   4. A button labeled “Modify”.
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 12.1]** The system shall display real-time updates for the scan progress and server responses in a readable, line-by-line format.

**[SRS 12.2]** The system shall highlight errors (404 Not Found) in red to differentiate them from successful responses.

**[SRS 12.3]** The system shall automatically display the latest server response during active scans.

1. **TRACE Tool Dashboard – HTTP Tester**

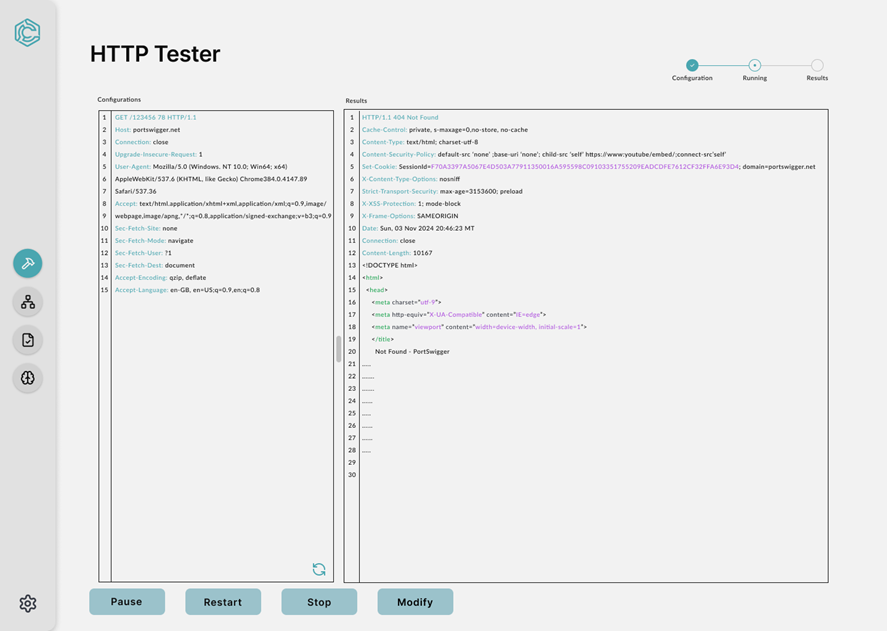


Figure 3.1.1-13 HTTP Tester

**Logical Characteristic**

**[SRS 13]** The HTTP Tester (Repeater) Phase shall allow users to view and compare HTTP request configurations with their corresponding server responses in real time. It shall include the following components (as shown in figure 3.1.1-13):

1. Section title labeled as “HTTP Tester”.
2. Two panes labeled “Configuration” and “Results”.
   1. The left pane displays the current HTTP request details with a refresh button located on the bottom right.
   2. The right pane displays the server’s response.
3. A visual progress bar with three stages: “Configuration”, “Running”, and “Results” with the current stage highlighted.
4. Buttons located at the bottom of the page including:
   1. A button labeled “Pause”.
   2. A button labeled “Restart”.
   3. A button labeled “Stop”.
   4. A button labeled “Modify”.
5. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 13.1]** The system shall display HTTP requests and responses in an aligned, readable, and scrollable format for easy comparison

**[SRS 13.2]** The system shall highlight errors (404 Not Found) in red to differentiate them from successful responses.

1. **TRACE Tool Dashboard – HTTP Results**

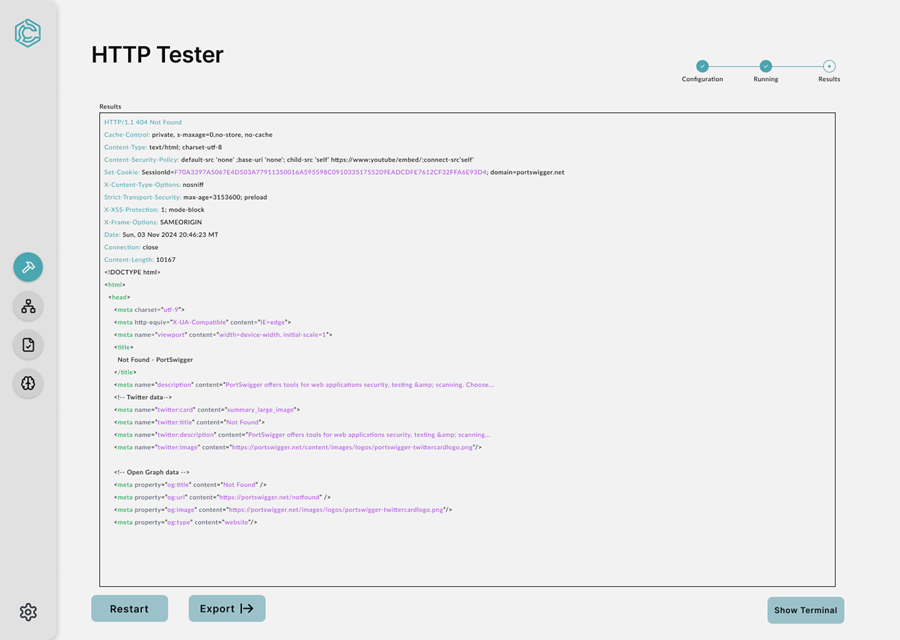


Figure 3.1.1-14 HTTP Results

**Logical Characteristic**

**[SRS 14]** The HTTP Tester Results Phase shall provide the final outcome of the HTTP request operation, displaying the server's full response and allowing the user to take follow-up actions. It shall include the following components (as shown in figure 3.1.1-14):

1. Section title labeled as “Results”.
2. A scrollable text area showing the complete server response.
3. A visual progress bar with three stages: “Configuration”, “Running”, and “Results” with the current stage highlighted.
4. Buttons located at the bottom of the page including:
   1. A button labeled “Restart”.
   2. A button labeled “Export”.
   3. A button labeled “Show Terminal”.
5. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 14.1]** The system shall format the response content clearly, using syntax highlighting for key elements such as headers and HTML tags.

**[SRS 14.2]** The system shall ensure the full response is viewable via scrolling.

1. **TRACE Tool Dashboard – Edit Request Configuration**

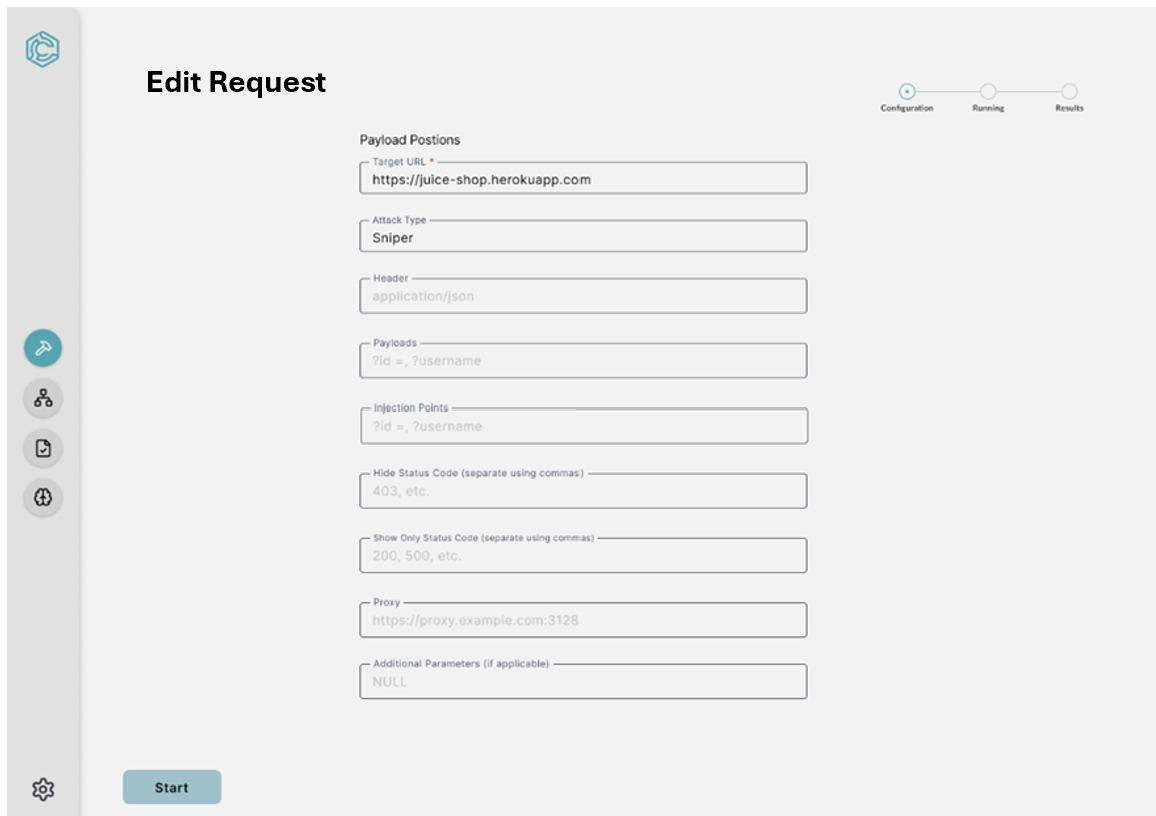


Figure 3.1.1-15 Intruder Configuration

**Logical Characteristic**

**[SRS 15]** The Edit Request Configuration Screen shall allow the user to set up payloads and attack parameters to execute sniper-style injection attacks and include the following components (as shown in figure 15):

1. Section title labeled as “Edit Request”.
2. A sub text labeled as “Payload Options”.
3. A list of input boxes labeled (Target URL, Attack Type, Header, Payloads, Injection Points, Hide Status Code, Show Status Code, Proxy, Additional Parameters). Each input box has a placeholder text.
4. Buttons and progress bars (as shown in Figure 3.1.1-11).
5. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 15.1]** The system shall provide real-time hover text for each field, explaining its purpose.

**[SRS 15.2]** The system shall highlight incomplete or invalid fields in red to prompt user correction.

1. **TRACE Tool Dashboard – Edit Request Running**

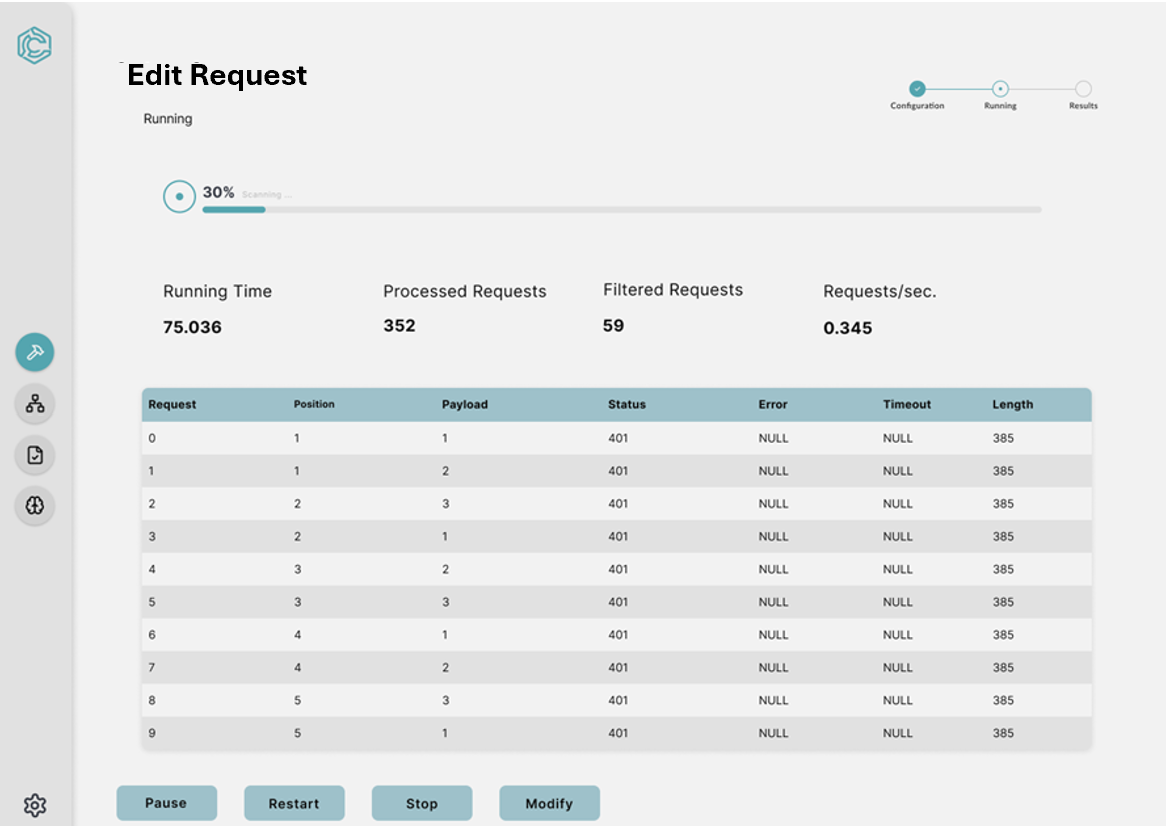


Figure 3.1.1-16 Intruder Running

**Logical Characteristic**

**[SRS 16]** The Edit Request Running Screen shall display the progress and details of an ongoing attack simulation. It shall include the following components (as shown in figure 3.1.1-16):

1. Section title labeled as “Edit Request Running”.
2. A sub text labeled “Running”.
3. A visual progress indicator showing the percentage completion of the scan (30% Scanning…).
4. Four key metrics display “Running Time”, “Processed Requests”, “Filtered Requests”, and “Requests/sec.”.
5. A data table displaying the Request, Position, Payload, Status, Error, Timeout, and Length.
6. Buttons and progress bars (as shown in Figure 3.1.1-12).
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 16.1]** The system shall dynamically update the table with new requests as they are processed.

**[SRS 16.2]** The system shall display the table data when the scan is paused or stopped.

1. **TRACE Tool Dashboard – Intruder Repeater**

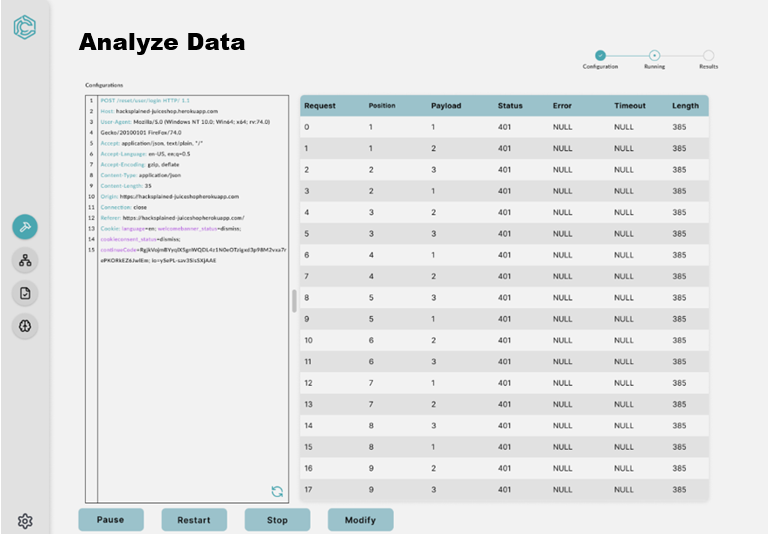


Figure 3.1.1-17 Analyze Data

**Logical Characteristic**

**[SRS 17]** The Analyze Data Screen shall allow users to analyze request and response data during an ongoing attack simulation. It shall include the following components (as shown in figure 3.1.1-17):

1. Section title labeled as “Analyze Data”.
2. A scrollable pane labeled “Configuration” displaying detailed request configuration data with a refresh icon on the bottom right.
3. Buttons and progress bars (as shown in Figure 3.1.1-13).
4. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 17.1]** The system shall dynamically update both the Configurations Panel and Data Table as new requests are processed.

**[SRS 17.2]** The system shall allow users to scroll independently through the Configurations Panel and Data Table.

1. **TRACE Tool Dashboard – Intruder Results**

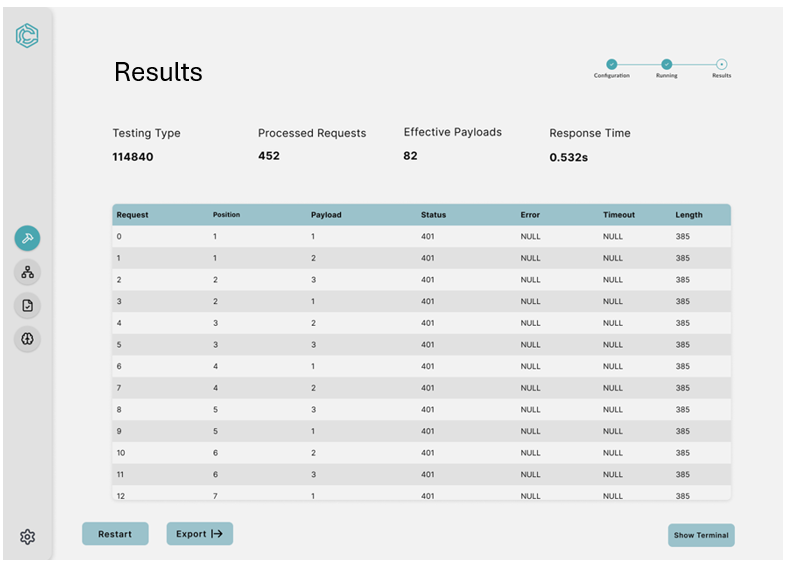


Figure 3.1.1-18 Change Request Results

**Logical Characteristic**

**[SRS 18]** The Intruder Results View shall allow users to analyze and export the outcomes of completed attack simulations. It shall include the following components (as shown in figure 3.1.1-18):

1. Section title labeled as "Results”.
2. A scrollable text area showing the complete server response.
3. Key metrics (as shown in Figure 3.1.1-16)
4. Buttons and progress bars (as shown in Figure 3.1.1-14).
5. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 18.1]** The system shall automatically sort the Data Table by Request Number.

1. **TRACE Tool Dashboard – SQL Injection Configuration**

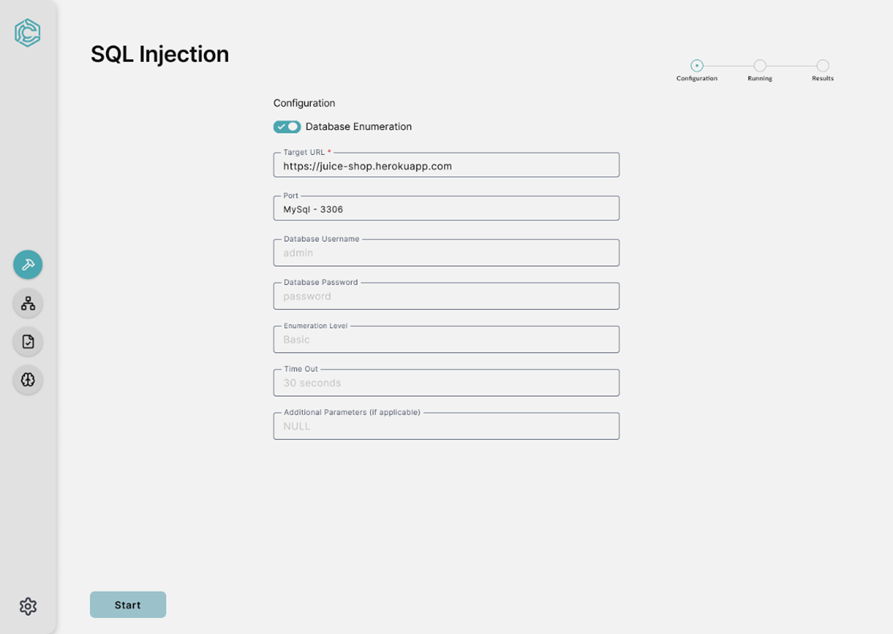


Figure 3.1.1-19 SQL Injection Configuration

**Logical Characteristic**

**[SRS 19]** The SQL Injection Configuration View shall allow users to set up and configure the parameters for performing database-focused SQL injection attacks. It shall include the following components (as shown in figure 3.1.1-19):

1. Section title labeled as “SQL Injection”.
2. A sub text labeled as “Configuration”.
3. A list of input boxes labeled (Target URL, Port, Database Username, Database Password, Enumeration Level, Time Out, Additional Parameters). Each input box has a placeholder text.
4. A toggle switch for “Database Enumeration”.
5. Buttons and progress bars (as shown in Figure 3.1.1-11).
6. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 19.1]** The system shall validate the input boxes before allowing the user to proceed.

**[SRS 19.2]** The system shall provide real-time hover text for each field, explaining its purpose.

**[SRS 19.3]** The system shall highlight incomplete or invalid fields in red to prompt user correction.

1. **TRACE Tool Dashboard – SQL Injection Running**

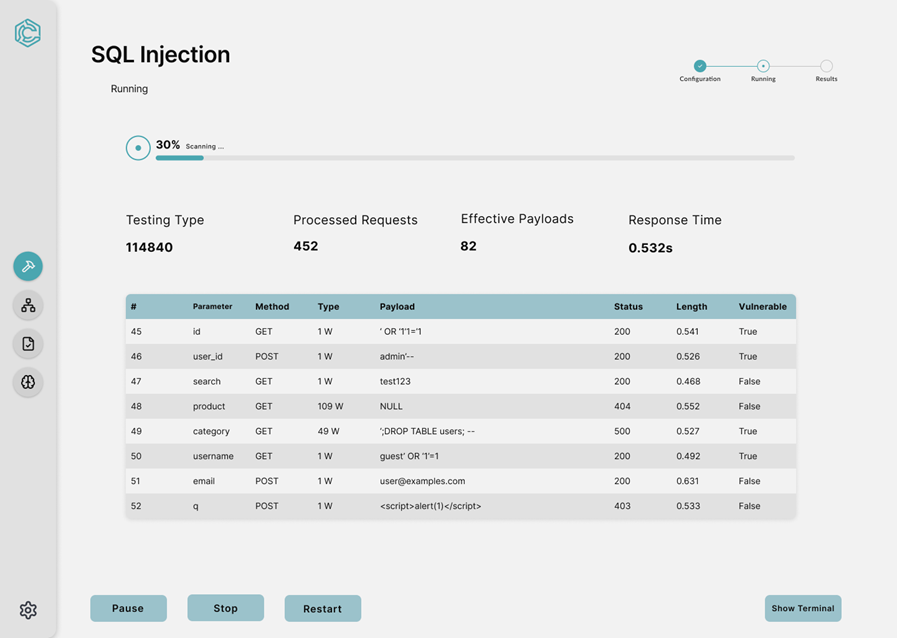


Figure 3.1.1-20 SQL Injection Running

**Logical Characteristic**

**[SRS 20]** The SQL Injection Scan page shall provide a functional interface for users to monitor and analyze SQL injection tests on a specified target. It shall include the following components (as shown in figure 3.1.1-20):

1. Section title labeled as “SQL Injection”.
2. A sub text labeled as “Running”.
3. A visual progress indicator showing the percentage completion of the scan (30% Scanning…).
4. Four key metrics display “Testing Type”, “Processed Requests”, “Effective Payloads”, and “Response Time”.
5. A data table displaying the #, Parameter, Method, Type, Payload, Status, Length, and Vulnerability.
6. Buttons and progress bars (as shown in Figure 3.1.1-12).
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 20.1]** The system shall display the progress bar dynamically, updating in real-time to reflect the scan’s progress accurately.

**[SRS 20.2]** The system shall display error messages in the Status column in a contrasting red color for quick identification.

**[SRS 20.3]** The system shall provide hover text for all action buttons with clear descriptions (“Pause the scan temporarily without losing progress”).

**[SRS 20.4]** The system shall require a confirmation prompt from the analyst before restarting a completed scan to prevent unintentional duplication.

**[SRS 20.5]** The system shall display an option to stop or modify settings to prevent a scan form continuing indefinitely.

1. **TRACE Tool Dashboard – SQL Injection Results**

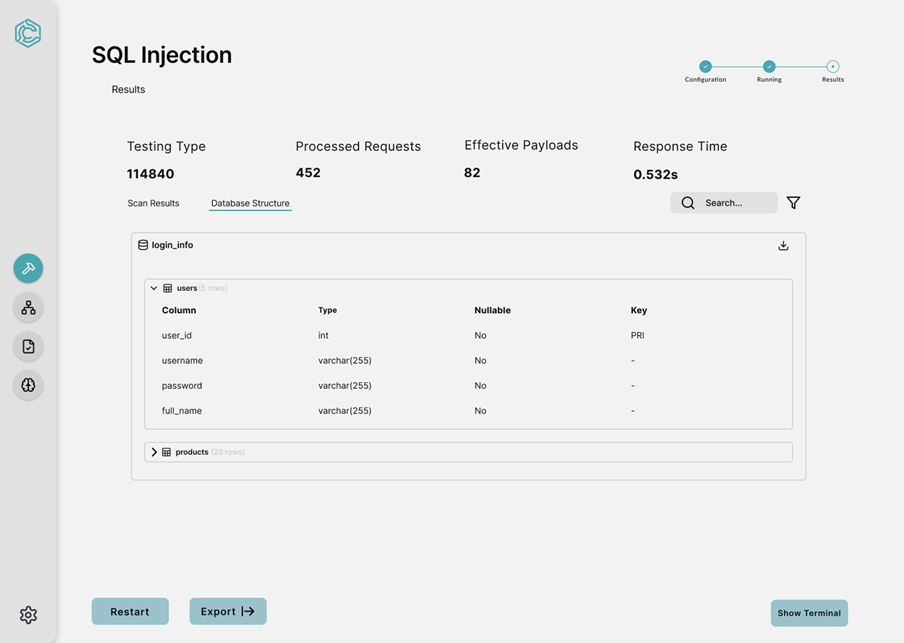


Figure 3.1.1-21 SQL Injection Results

**Logical Characteristic**

**[SRS 21]** The SQL Injection Results page shall present the outcomes of the SQL injection scan in an organized and actionable manner. It shall include the following components (as shown in figure 3.1.1-21):

1. Section title labeled as “SQL Injection”.
2. A sub text labeled as “Results”.
3. A tab interface with two tabs labeled “Scan Results” and “Database Structure”. Where “Database Structure” displays a table with collapsable tabs:
   1. Login Info includes column, type, nullable and key of the logins.
4. Four key metrics display “Testing Type”, “Processed Requests”, “Effective Payloads”, and “Response Time”.
5. Buttons and progress bars (as shown in Figure 3.1.1-14).
6. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 21.1]** The system shall display column details in a readable table format with appropriate padding and alignment for clarity.

**[SRS 21.2]** The system shall allow users to collapse or expand table schemas dynamically to focus on specific parts of the database structure.

**[SRS 21.3]** The system shall display hover text for all interactive icons and buttons, explaining their functionality ("Export results").

**[SRS 21.4]** The system shall ensure the full response if viewable via scrolling.

1. **TRACE Tool Dashboard – Parameter Fuzzing Configuration**



Figure 3.1.1-22 Parameter Fuzzing Configuration

**Logical Characteristic**

**[SRS 22]** The Parameter Fuzzing Setup page shall allow users to configure input parameters and settings for testing the robustness of web application inputs. It shall include the following components (as shown in figure 3.1.1-22):

1. Section title labeled as “Parameter Fuzzing”.
2. A sub text labeled as “Configuration”.
3. A list of input boxes labeled (Target URL, Word List, Cookies, Hide Status, Show Status, Filter by Content Length, Additional Parameters). Each input box has a placeholder text.
4. A button labeled “upload wordlist”.
5. Radio buttons labeled “GET”, “PUT”, and “POST”.
6. Buttons and progress bars (as shown in Figure 11).
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 22.1]** The system shall validate the fields for correctness and completeness before enabling the Start button.

**[SRS 22.2]** The system shall provide hover text for all input fields and buttons, explaining their purpose ("Enter the target URL for fuzzing")

**[SRS 22.3]** The system shall disable unused fields dynamically based on the selected HTTP method to avoid user confusion (disable Cookies if GET is selected)

**[SRS 22.4]** The system shall highlight errors in user input with a red outline and accompanying error messages.

1. **TRACE Tool Dashboard – Parameter Fuzzing Running**

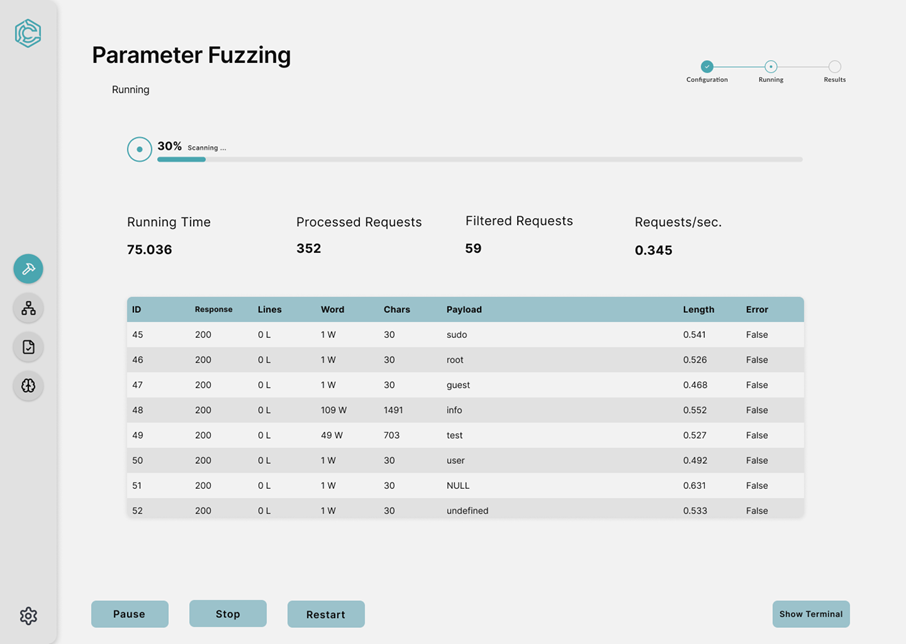


Figure 3.1.1-23 Parameter Fuzzing Running

**Logical Characteristic**

**[SRS 23]** The Parameter Fuzzing Scan page shall display real-time information and results of the ongoing parameter fuzzing process. It shall include the following components (as shown in figure 3.1.1-23):

1. Section title labeled as “Parameter Fuzzing”.
2. A sub text labeled “Running”.
3. A visual progress indicator showing the percentage completion of the scan (30% Scanning…).
4. Four key metrics display “Running Time”, “Processed Requests”, “Filtered Requests”, and “Requests/sec.”.
5. A data table displaying the ID, Response, Lines, Word, Chars, Payload, Length, and Error.
6. Buttons and progress bars (as shown in Figure 3.1.1-12).
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 23.1]** The system shall update the progress bar dynamically in real time, accurately reflecting the progress of the fuzzing process.

**[SRS 23.2]** The system shall provide detailed hover text for table headers to clarify the meaning of columns ("Payload: The input tested for vulnerabilities").

**[SRS 23.3]** The system shall display a confirmation prompt to the analyst before stopping a scan to prevent accidental termination.

1. **TRACE Tool Dashboard – Parameter Fuzzing Results**

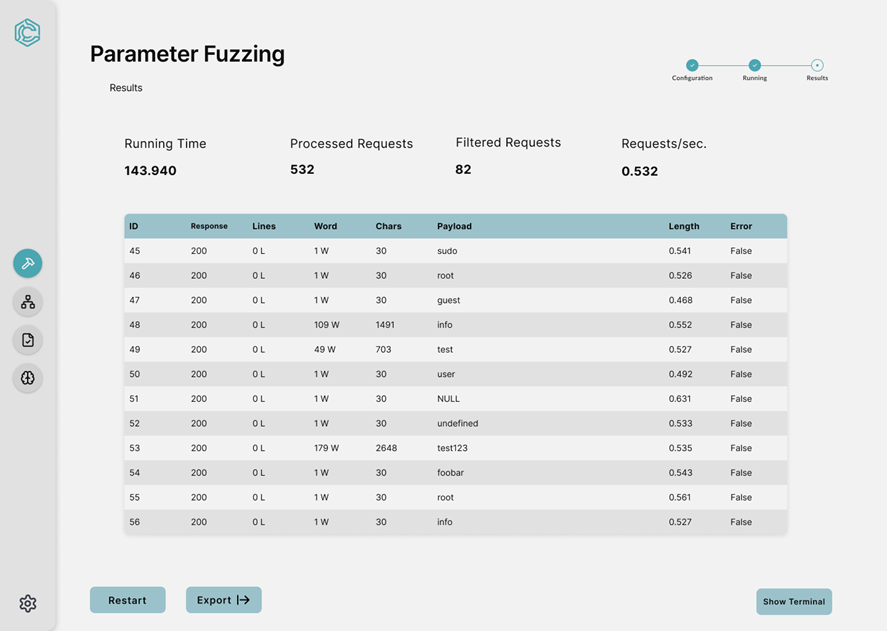


Figure 3.1.1-24 Parameter Fuzzing Results

**Logical Characteristic**

**[SRS 24]** The Parameter Fuzzing Result page shall present the finalized results of the parameter fuzzing scan. It shall include the following components (as shown in figure 3.1.1-24):

1. Section title labeled as “Parameter Fuzzing”.
2. A sub text labeled as “Results”.
3. A data table displaying the ID, Response, Lines, Word, Chars, Payload, Length, and Error.
4. Four key metrics display “Running Time”, “Processed Requests”, “Filtered Requests”, and “Requests/sec.”
5. Buttons and progress bars (as shown in Figure 3.1.1-14).
6. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 24.1]** The system shall display column details in a readable table format with appropriate padding and alignment for clarity.

**[SRS 24.2]** The system shall highlight errors or suspicious responses ("500 Internal Server Error") with a unique background color for easy identification.

**[SRS 24.3]** The system shall display hover text for all interactive icons and buttons, explaining their functionality ("Export results").

**[SRS 24.4]** The system shall ensure that all of the results are visible to the analyst with the use of scrolling.

1. **TRACE Tool Dashboard –Terminal**

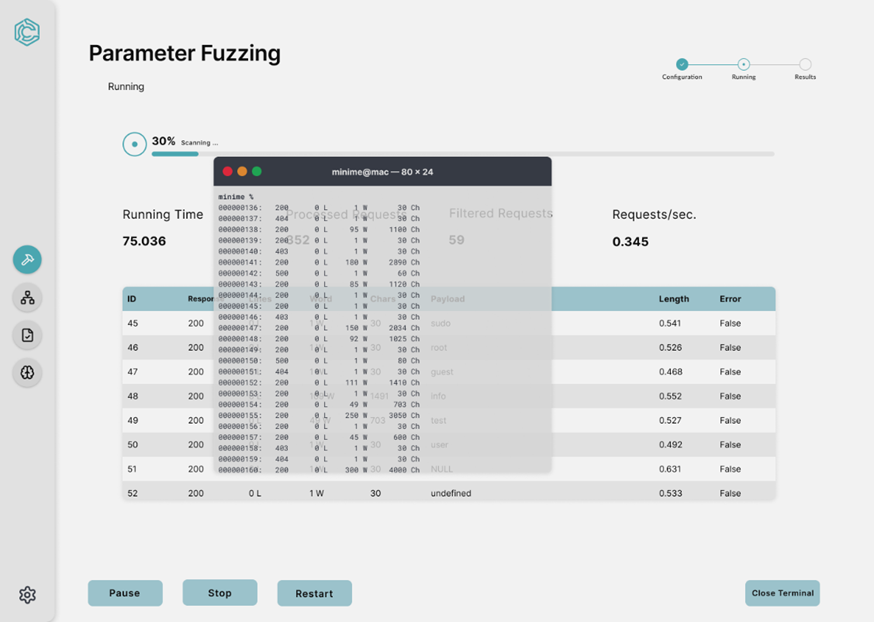


Figure 3.1.1-25 Terminal

**Logical Characteristic**

**[SRS 25]** The Parameter Fuzzing Scan page with Terminal Overlay shall provide an interface to monitor real-time scan progress alongside terminal output. It shall include the following components (as shown in figure 3.1.1-25):

1. Running Service screen (as shown in figure 3.1.1-24).
2. A collapsable terminal window overlay positioned centrally over the results table which displays real time requests and responses from services.
3. The terminal window shall include a close button at the top left and button labeled “Close Terminal” in the running service page.
4. The terminal window includes a text above in the center detailing the machine and the window size “minime@mac-80x24”.

**Optimized Aspects**

**[SRS 25.1]** The system shall provide a real-time, draggable, scrollable terminal view displaying all HTTP requests and responses, updated dynamically during the scan.

**[SRS 25.2]** The system shall maintain access to the Pause, Stop, and Restart buttons, even when the terminal overlay is open, to ensure uninterrupted user control

**[SRS 25.3]** The terminal shall include a Copy to Clipboard feature for users to export specific log entries directly from the overlay.

**[SRS 25.4]** The system shall maintain transparency or an adjustable size to allow the analyst to view critical metrics.

**[SRS 25.5]** The system shall disable the terminal screen once the scan is stopped to ensure the interface remains relevant to the task.

**[SRS 25.6]** The system shall disable the terminal screen once the scan is completed to ensure the interface remains relevant to the task.

1. **TRACE Tool Dashboard – Brute Force Configuration**

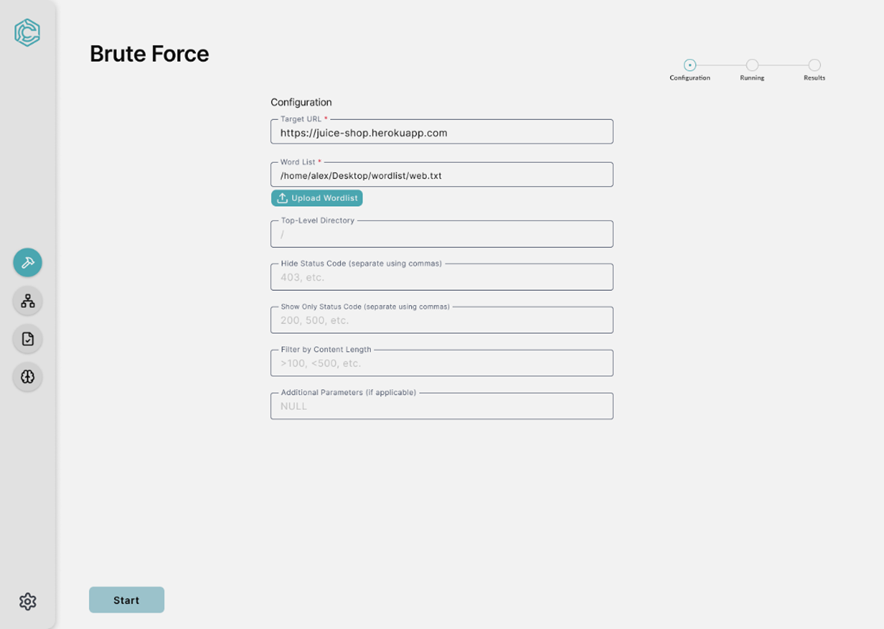


Figure 3.1.1-26 Brute Force Configuration

**Logical Characteristic**

**[SRS 26]** The Brute Force Setup Page shall allow users to configure a brute force scan for discovering hidden directories and files on a target system. It shall include the following components (as shown in figure 3.1.1-26):

1. Section title labeled as “Brute Force”.
2. A sub text labeled as “Configuration”.
3. A list of input boxes labeled (Target URL, Word List, Top Level Directory, Hide Status, Show Only Status, Filter by Content Length, Additional Parameters). Each input box has a placeholder text.
4. An upload button labeled “Upload Wordlist” with an upload icon.
5. Buttons and progress bars (as shown in Figure 3.1.1-11).
6. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 26.1]** The system shall validate all required fields (Target URL and Word List) before enabling the Start button.

**[SRS 26.2]** The system shall ensure the uploaded word list is compatible with the tool, checking for file extensions and format.

**[SRS 26.3]** The system shall highlight errors in user input with a red outline and accompanying error messages.

**[SRS 26.4]** The system shall display an error message if the Target URL is not provided.

**[SRS 26.5]** The system shall display an error message if the Word List is not provided.

1. **TRACE Tool Dashboard – Brute Force Running**

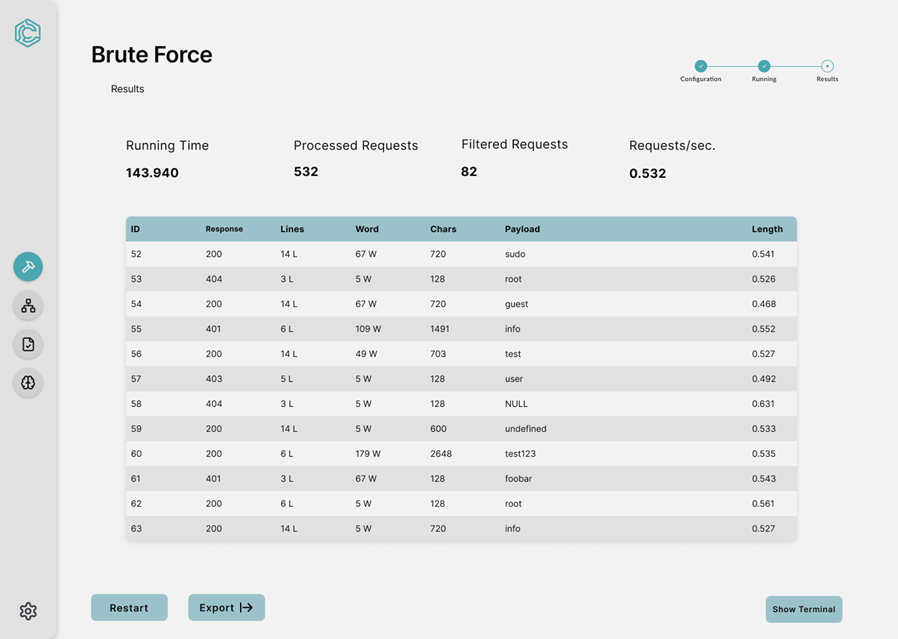


Figure 3.1.1-27 Brute Force Running

**Logical Characteristic**

**[SRS 27]** The Brute Force Running Page shall provide a real-time view of the ongoing brute force operation, displaying the progress, key statistics, and detailed request data. It shall include the following components (as shown in figure 3.1.1-27):

1. Section title labeled as “Brute Force”.
2. A sub text labeled “Running”.
3. A visual progress indicator showing the percentage completion of the scan (30% Scanning…).
4. Four key metrics display “Running Time”, “Processed Requests”, “Filtered Requests”, and “Requests/sec.”.
5. A data table displaying the ID, Response, Lines, Word, Chars, Payload, Length, and Error.
6. Buttons and progress bars (as shown in Figure 3.1.1-12).
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 27.1]** The system shall dynamically update statistics and the results table in real time without refreshing the page.

**[SRS 27.2]** The system shall allow the user to sort and filter the table based on any column (Response, Payload).

**[SRS 27.3]** The system shall require the analyst to confirm before stopping a scan to prevent accidental termination.

1. **TRACE Tool Dashboard – Brute Force Results**

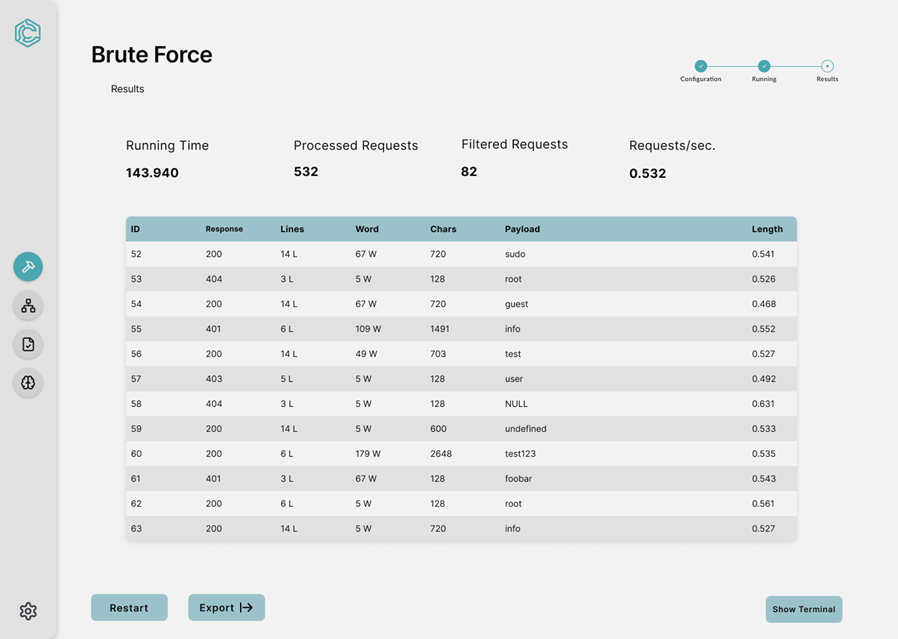


Figure 3.1.1-28 Brute Force Results

**Logical Characteristic**

**[SRS 28]** The Brute Force Results Page shall provide a detailed summary of the completed brute force operation. It shall include the following components (as shown in figure 3.1.1-28):

1. Section title labeled as “Brute Force”.
2. A sub text labeled as “Results”.
3. A data table displaying the ID, Response, Lines, Word, Chars, Payload, Length, and Error.
4. Four key metrics display “Running Time”, “Processed Requests”, “Filtered Requests”, and “Requests/sec.”
5. Buttons and progress bars (as shown in Figure 3.1.1-14).
6. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 28.1]** The system shall display column details in a readable table format with appropriate padding and alignment for clarity.

**[SRS 28.2]** The system shall allow users to sort the results table by any column (Response, Payload) to facilitate easy analysis.

**[SRS 28.3]** The system shall display hover text for all interactive icons and buttons, explaining their functionality ("Export results").

**[SRS 28.4]** The system shall ensure the full response is visible via the use of scrolling.

1. **TRACE Tool Dashboard – Crawler Configuration**

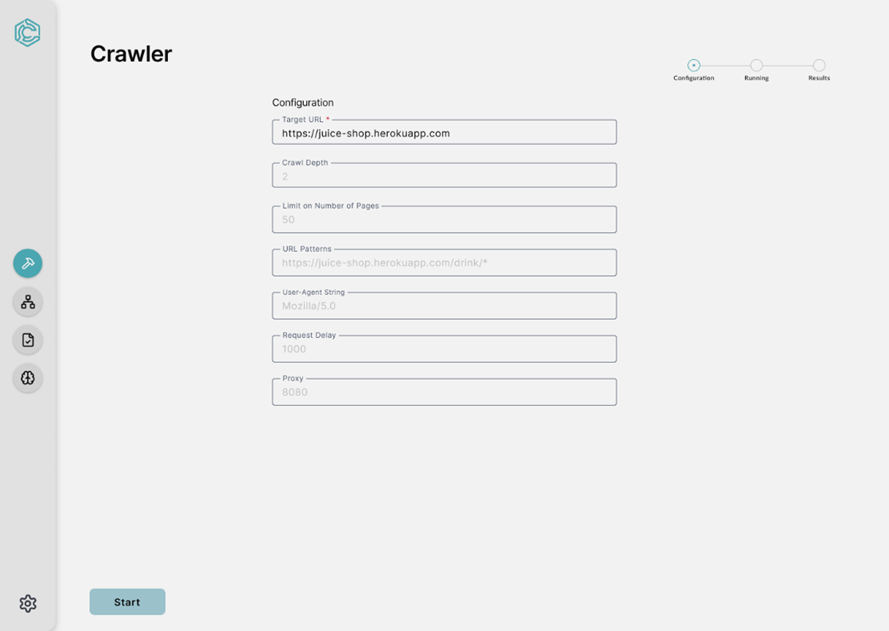


Figure 3.1.1-29 Crawler Configuration

**Logical Characteristic**

**[SRS-29]** The Crawler Configuration Page shall provide an interface for users to set parameters for crawling a target website. It shall include the following components (as shown in figure 3.1.1-29):

1. Section title labeled as “Crawler”.
2. A sub text labeled as “Configuration”.
3. A list of input boxes labeled (Target URL, Crawl Depth, Limit on Number of Pages, User Agent String, Request Delay, Proxy, Additional Parameters). Each input box has a placeholder text.
4. Buttons and progress bars (as shown in Figure 3.1.1-11).
5. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 29.1]** The system shall validate all required fields (Target URL) before enabling the Start button.

**[SRS 29.2]** The system shall display a tooltip or hover text for each field to explain its functionality ("Crawl Depth: Number of link levels to traverse from the base URL").

**[SRS 29.3]** The system shall highlight errors in user input with a red outline and accompanying error messages.

**[SRS 29.4]** The system shall display an error message if the Target URL is not found.

**[SRS 29.5]** The system shall require only positive and real values.

1. **TRACE Tool Dashboard – Crawler Running**

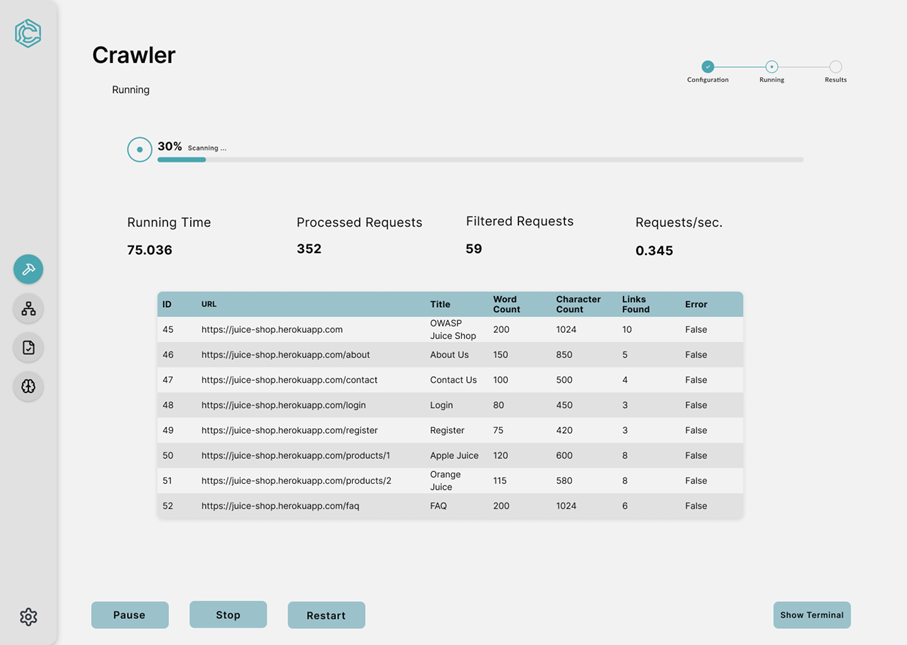


Figure 3.1.1-30 Crawler Running

**Logical Characteristic**

**[SRS-30]** The Crawler Running Page shall provide real-time updates on the ongoing crawl operation and display detailed information about each processed URL. It shall include the following components (as shown in figure 3.1.1-30):

1. Section title labeled as “Crawler”.
2. A sub text labeled “Running”.
3. A visual progress indicator showing the percentage completion of the scan (30% Scanning…).
4. Four key metrics display “Running Time”, “Processed Requests”, “Filtered Requests”, and “Requests/sec.”.
5. A data table displaying the ID, URL, Title, Word Count, Character Count, Links Found, and Error.
6. Buttons and progress bars (as shown in Figure 3.1.1-12).
7. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 30.1]** The system shall update the statistics and progress bar dynamically in real-time to reflect the current state of the crawl.

**[SRS 30.2]** The results table shall allow users to sort columns (e.g., by Word Count or Links Found) for better data organization.

**[SRS 30.3]** The system shall provide tooltips or hover text for each action button to clarify their functionality (“Stop: End the current crawl and save results”).

**[SRS 30.4]** The system shall require confirmation from the analyst before stopping the scan to prevent accidental termination.

1. **TRACE Tool Dashboard – Crawler Results**

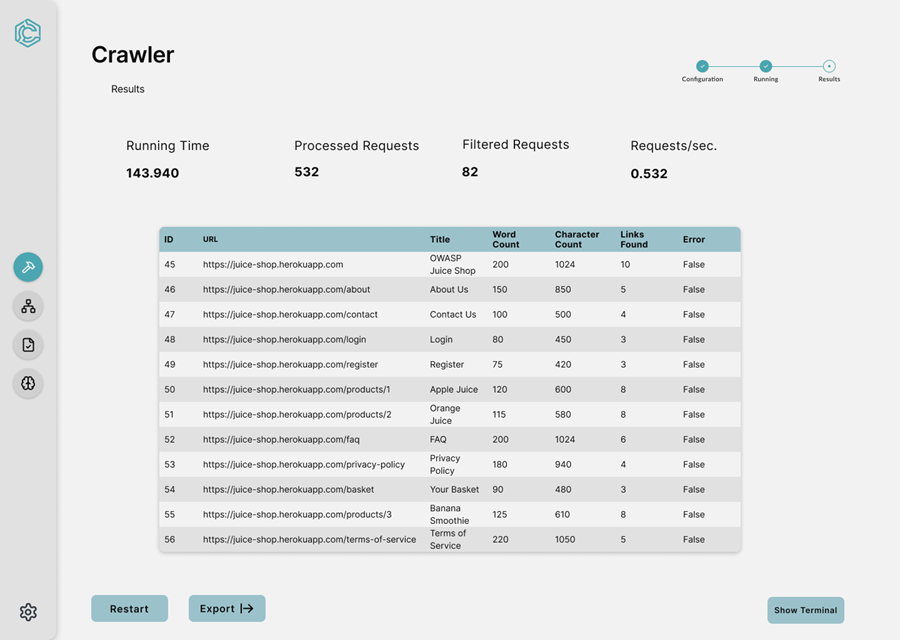


Figure 3.1.1-31 Crawler Results

**Logical Characteristic**

**[SRS 31]** The Crawler Results Page shall provide a summary of the crawl operation results and allow users to review detailed information about each crawled URL. It shall include the following components (as shown in figure 3.1.1-31):

1. Section title labeled as “Crawler”.
2. A sub text labeled as “Results”.
3. A data table displaying the ID, URL, Title, Word Count, Character Count, Links Found, and Error.
4. Four key metrics display “Running Time”, “Processed Requests”, “Filtered Requests”, and “Requests/sec.”
5. Buttons and progress bars (as shown in Figure 3.1.1-14).
6. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 31.1]** The system shall display column details in a readable table format with appropriate padding and alignment for clarity.

**[SRS 31.2]** The system shall display hover text for all interactive icons and buttons, explaining their functionality ("Export results").

**[SRS 31.3]** The system shall ensure the full response is viewable via the use of scrolling.

1. **TRACE Tool Dashboard – Results**

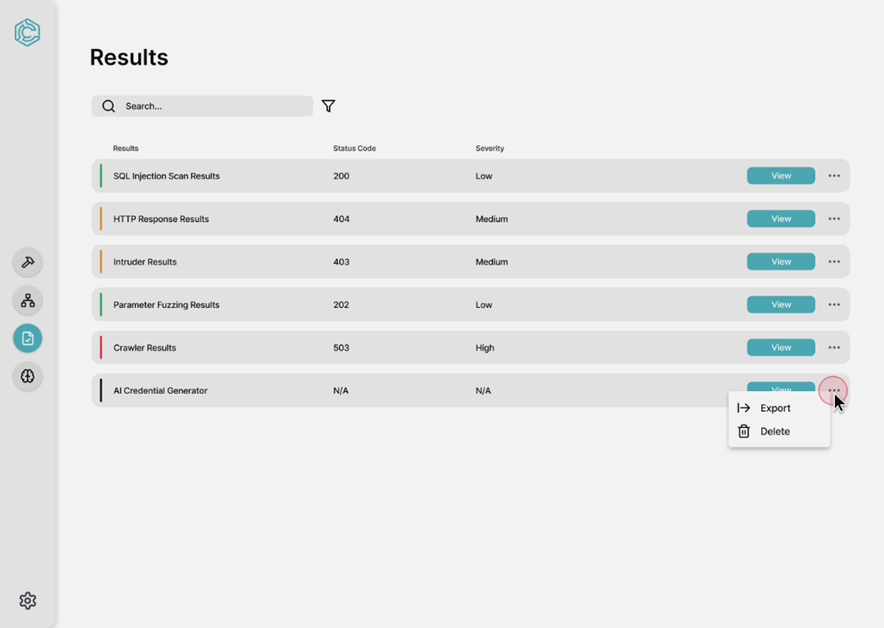


Figure 3.1.1-32 Results

**Logical Characteristic**

**[SRS-32]** The Crawler Results Page shall provide a summary of the crawl operation results and allow users to review detailed information about each crawled URL. It shall include the following components (as shown in figure 3.1.1-32):

1. Section title labeled as “Results”.
2. A search bar to filter results dynamically with a filter icon to provide advanced sorting and filtering.
3. A list of result cards with the following information:
   1. Results Name, Status Code, Severity, and a button labeled “view”.
   2. An ellipsis with additional options including two buttons labeled “Export” and “Delete”.
4. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 32.1]** The system shall ensure that the status indicators (colors) are universally distinguishable and accompanied by text for accessibility.

**[SRS 32.2]** The search bar shall filter results and allow multi-criteria searches ("Severity: High").

**[SRS 32.3]** The system shall prompt confirmation dialogs before performing irreversible actions like deletion.

**[SRS 32.4]** The system shall only display complete results.

**[SRS 32.5]** The system shall display in progress results on their service-specific dashboard.

**[SRS 32.6]** The system shall display filters on the table to ensure consistency.

1. **TRACE Tool Dashboard – Tree Graph List**

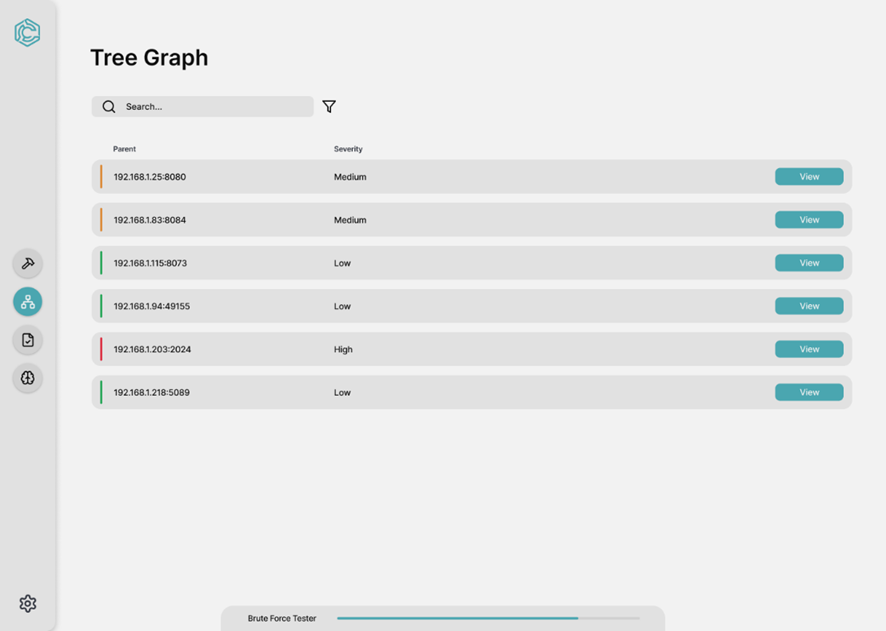


Figure 3.1.1-33 Tree Graph List

**Logical Characteristic**

**[SRS 33]** The Tree Graph List Page shall display a structured overview of parent nodes discovered during a scan in the TRACE system. It shall include the following components (as shown in figure 3.1.1-33):

1. Section title labeled as “Tree Graph”.
2. A search bar to filter results dynamically with a filter icon to provide advanced sorting and filtering.
3. A list of result cards with the following information:
   1. Parent, Severity, and a button labeled “view”.
   2. Status indicators with green, orange and red colors indicating the severity of each IP/Port combination.
4. Displays the progress of the currently running scan (Brute Force Tester) at the bottom of the screen with a loading bar.
5. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 33.1]** The system shall ensure severity indicators are consistently displayed with both color and accompanying text for accessibility.

**[SRS 33.2]** The search bar shall filter results and allow multi-criteria searches ("Severity: High").

**[SRS 33.3]** The progress bar at the bottom shall display the status of the ongoing scan in real time, with clear labeling of the tool being used.

1. **TRACE Tool Dashboard – Tree Graph View**

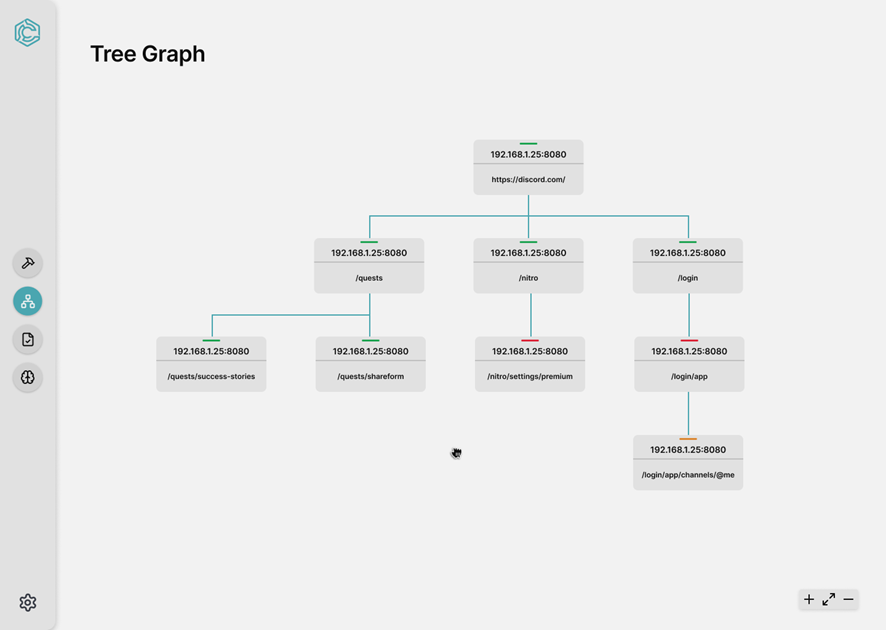


Figure 3.1.1-34 Tree Graph View

**Logical Characteristic**

**[SRS 34]** The Tree Graph View Page shall display a hierarchical representation of nodes and their connections discovered during a scan in the TRACE system. It shall include the following components (as shown in figure 3.1.1-34):

1. A tree like graph is represented in the center of the screen where it displays the following:
   1. A primary node at the top of the tree structure with children nodes below them.
   2. A severity bar located in the center top of each node with a descriptive text.
   3. Child Nodes are connected to the root and include paths or endpoint descriptions (“/quests”, “/login”) for clarity.
2. Zoom in, zoom out, and reset buttons located on the bottom right of the screen to allow users to adjust the view.
3. A draggable viewport for easy navigation through complex tree structures.
4. A side bar with options (as shown in Figure 3.1.1-9).

**Optimized Aspects**

**[SRS 34.1]** The system shall ensure all nodes are clearly labeled with their respective paths and severity levels.

**[SRS 34.2]** The system shall provide a responsive layout to handle varying numbers of nodes and maintain usability on different screen sizes.

**[SRS 34.3]** The navigation tools (Zoom In, Zoom Out, Reset) shall be intuitive and prominently displayed.

**[SRS 34.4]** The system shall allow users to drag and reposition the viewport for exploring large graphs.

**[SRS 34.5]** The system shall ensure readability for dense structures by not allowing nodes to overlap or clutter the graph.

1. **TRACE Tool Dashboard – AI Credential Generator Setup**

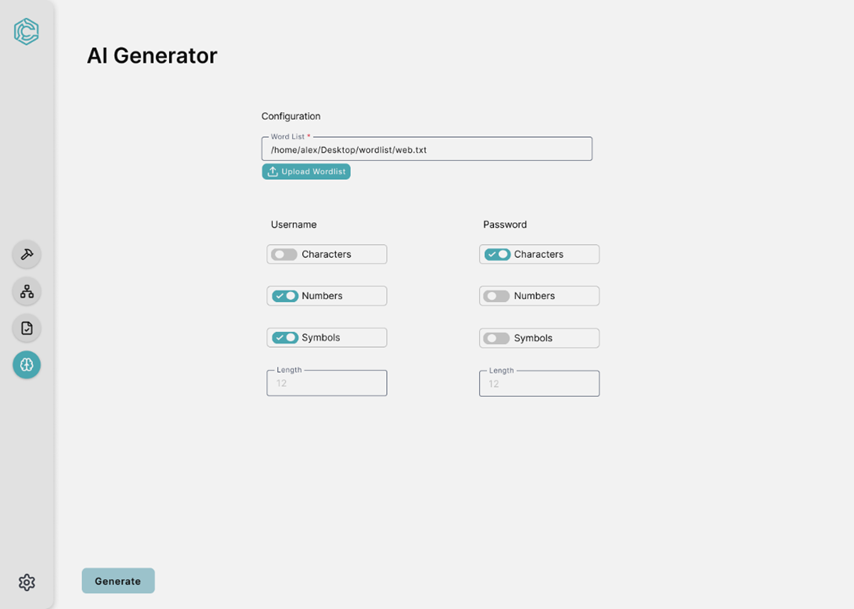


Figure 3.1.1-35 AI Setup

**Logical Characteristic**

**[SRS 35]** The AI Credential Generator Setup Screen shall allow users to configure the parameters for generating usernames and passwords. It shall include the following components (as shown in figure 3.1.1-35):

1. Section title labeled as “AI Generator”.
2. A sub text labeled as “Configuration”.
3. A text input box to specify the file path for the wordlist.
4. A button labeled as “Upload Wordlist” with a file upload icon.
5. Toggles for enabling/disabling the inclusion of the following for username and passwords:
   1. Characters (e.g., A-Z, a-z)
   2. Numbers (e.g., 0-9)
   3. Symbols (e.g., @, #, $)
6. Input box to specify the length of the generated password/username.
7. A button labeled “Generate” located on the left bottom corner of the page.
8. A side bar with options (as shown in Figure 3.1.1-9).

**[SRS 37] Students will take the code provisioned by DAC staff and perform the modifications required as dictated in the segment below:**

1. Logging will need to be implemented within the MDP Process

**Artificial Intelligence (AI) Subroutine:**

1. The development team shall evaluate and modify, as necessary the following aspects of the AI subroutine:
   1. Review and modify the order parameters current values to further optimize the generation of viable passwords
   2. assess whether the state transitions capture sufficient context for meaningful credential generation
   3. Evaluate and modify the calculate\_password\_strength method's weights.
   4. Evaluate and modify the calculate\_username\_quality method's weights.
   5. Review and modify the following learning parameters:
      1. Epsilon value (Currently .1, deals with exploration-exploitation balance)
      2. Learning rate (Currently .1, deals with Q-value updates)
      3. Gamma Factor (Currently .9, deals with future reward considerations)
   6. Improve the algorithm to dynamically heighten rewards given to states or actions that place approved special characters in positions deemed as important.
   7. The MDP algorithm shall incorporate a distinct symbol state in order to determine which special characters will be selected for the username/password generation.

**Web Scraping Subroutine:**

1. The Web Scraping subroutine shall be updated to perform the following
   1. The system shall organize URLs into a hierarchy tree structure to facilitate structured exploration, enabling efficient parallel processing and CSV generation.
   2. The scraper algorithm shall be modified to extract all words associated with logos, labels and class titles.
   3. The scraper algorithm shall be modified to perform batch processing in order to increase the efficiency.
   4. The web scraper shall leverage aiohttp in order to parallelize web scraping.

**Natural Language Processing (NLP) Subroutine:**

1. THE NLP subroutine shall be updated to perform the following:
   1. The NLP algorithm shall normalize all text in its output by removing non-english characters and resolving encoding issues.
   2. The NLP algorithm shall break up compound words (ex. sister-in-law -> sister,in,law)
   3. The NLP algorithm shall remove all definite articles, Demonstrative Determiners, Distributive Determiners and Interrogative Determiners from the text leveraging a NLP algorithm.
   4. The NLP algorithm shall remove all words less than 4 characters long unless the word is determined to be an acronym.
   5. The NLP algorithm shall remove all instances of possessive determiners except for those determined to be as part of a technological name or user role/name

**Optimized Aspects**

**[SRS 35.1]** The system shall validate user input (ensure the length values are positive integers and within a reasonable range)

**[SRS 35.2]** The toggles for Characters, Numbers, and Symbols shall have distinct, visible states to indicate activation.

**[SRS 35.3]** The system shall send an error message if the crawler service has not been executed.

1. **TRACE Tool Dashboard – AI Credential Generator Results**

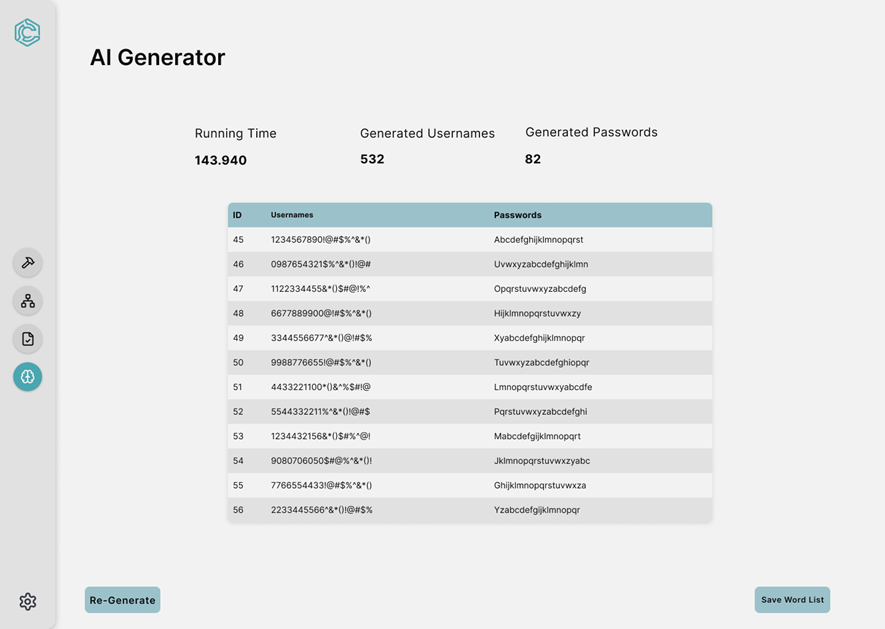


Figure 3.1.1-36 AI Results

**Logical Characteristic**

**[SRS 36]** The AI Credential Generator Results Screen shall display the generated usernames and passwords along with their associated metadata. It shall include the following components (as shown in figure 3.1.1-36):

1. Section title labeled as “AI Generator”.
2. Four key metrics display “Running Time”, “Generated Usernames”, and “Generated Passwords”.
3. A table displaying ID, Usernames, and Passwords.
4. A button labeled “Re-Generate” located on the left bottom corner of the page.
5. A button labeled "Save Word List" located on the right bottom of the page.
6. A side bar with options (as shown in Figure 9).

**Optimized Aspects**

**[SRS 36.1]** The system shall save the generated credentials after they are generated.

**[SRS 36.2]** The "Save Word List" button shall display a confirmation message after successful saving.

**[SRS 36.3]** The system shall enable horizontal scrolling to ensure that all usernames and passwords are visible.

# **Hardware Interfaces**

**[SRS 3.1.2.1]** TRACE shall operate on hardware meeting the following minimum specifications:

1. Processor: 4-core, 2.5 GHz or higher
2. RAM: 8GB or higher
3. Storage: 10GB of available disk space
4. Network Interface: Ethernet or wireless adapter supporting 100Mbps or higher.

**[SRS 3.1.2.2]** TRACE shall interact with the following external hardware:

1. Servers: TRACE shall send and receive data from secure servers configured within the internal network.
2. Switches/Routers: TRACE shall perform reconnaissance tasks on networks through switches and routers.
3. End Devices: TRACE shall interface with target machines for scanning and data collection.

**[SRS 3.1.2.3]** TRACE may require additional storage devices to store project files, such as project logs and project results.

# **Software Interfaces**

**[SRS 3.1.3.1]** TRACE shall not interact with other software systems.

# **Communications Interfaces**

**[SRS 3.1.4.1]** TRACE shall support the following communication protocols:

1. TRACE shall use HTTPS to communicate with web-based applications and servers during testing.
2. TRACE shall perform network scans using TCP/IP for communication with targeted machines.
3. TRACE shall use ICMP for ping and ARP requests to test the accessibility of hosts.
4. TRACE shall send SQL queries during SQL injection testing to gather database information.

**[SRS 3.1.4.2]** TRACE shall function on a closed, internal network and will not require external internet connectivity for its operations.

**[SRS 3.1.4.3]** TRACE shall use secure communication protocols to protect data transmission:

1. SSL/TLS for HTTPS requests.
2. Encrypted channels for all data exchanges to prevent interception or tampering.

**[SRS 3.1.4.1]** TRACE shall interface with external systems using standard API’s for importing/exporting data in CSV or XML formats.

# **Behavioral Requirements**

# **Same Class of User**

**[SRS 3.2.1.1]** TRACE shall have one level of access privilege assigned as “Lead Analyst” with full access to the system’s functionality.

**[SRS 3.2.1.2]** TRACE shall have another level of access privilege assigned as “Analyst” that will have access to all system functionality except for Creating, locking and deleting a project.

# **Related Real-world Objects**

Real-world objects represent business elements/components that interact to provide the services of the System of Interest. Table 3.2.2-1 shows TRACE’s real-world objects (business objects) and their purpose.

**Table 3.2.2-1: TRACE Real World Objects.**

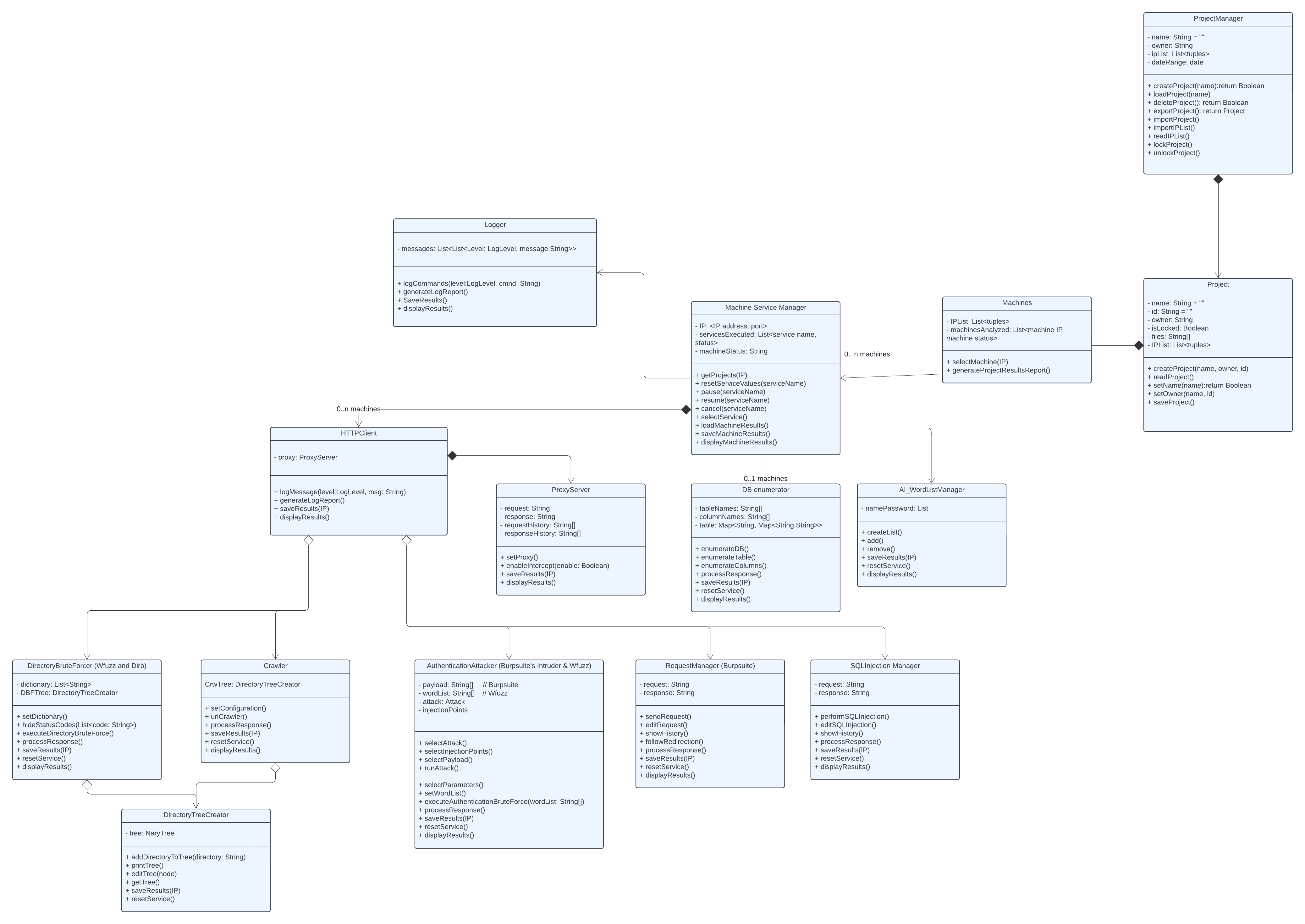
|  |  |
| --- | --- |
| **Real-World Object Name** | **Purpose** |
| Logger | The logger class creates a log for the actions and decisions of the system based on how it reacts to the user’s inputs and actions. It generates a log report at the end of the project lifecycle. |
| Project Manager | The Project Manager allows the analyst to manage the creation or deletion of a project. The analyst is able to create, load, lock, delete, export, or import a project. |
| Project | The Project class stores all of the information of the project, including the project name, the owner, if it is locked or unlocked and the IP list. |
| HTTP Client | The HTTP Client class allows the analyst to set up a proxy server with the intended machines. |
| Directory Brute Forcer (Wfuzz and dirb) | The Directory Brute Force Service class allows the analyst to execute the directory brute force service on the selected machines. With the information gathered from running the brute force, it builds the directory path. |
| Crawler | The Crawler class allows the analyst to crawl through the selected URL and create a path. |
| Directory Tree Creator | The Directory Tree Creator takes the information from the Crawler and Directory Brute Force Service classes and builds a directory tree based on their paths. |
| Authentication Attacker | The authentication attacker allows the analyst to select the attack it shall perform on the machines. It also allows the analyst to select the parameters, set the word list, execute the brute force process, and process the response. |
| SQL Injection Manager | The SQL Injection class allows the analyst to perform or modify an SQL injection attack and process the response from the attacked machines. |
| Proxy Server | The Proxy server class allows the analyst to set a proxy and enable an intercept. It will also store the request and response histories. |
| DB Enumerator | The DB enumerator class shall allow the analyst to enumerate the tables, columns, and responses from the database. |
| AI\_Wordlist Manager | The AI wordlist manager allows the analyst to create, add, or remove a word list. The analyst can also save or reset a service. |
| Request Manager | The repeater class allows the analyst to send, edit, and process the response of sending requests to the machines. |
| Machine Service Manager | The machine service manager allows the analyst to select, reset, pause, resume, and cancel a service. |
| Machines | The machines class holds the IP list, the list of machines analyzed, and shall allow the analyst to select a machine and save the machine results. |

**The relations among real-world objects for this system are shown in Figure 3.2.2-1.**

The purpose of the class diagram is to show the interaction among real-world objects to facilitate an understanding of how a system’s services can be managed from a business point of view. The class diagram must be revisited when design constraints, quality attributes, and business constraints are considered during the architecture and detail design so relations and responsibilities may change, and new implementation classes can be added. The legend of the class diagram is shown in Table 3.2.2-2: Class Diagram Legend.

**Table 3.2.2-2: Class Diagram Legend**

|  |  |  |
| --- | --- | --- |
| **NAME** | **NOTATION** | **DESCRIPTION** |
| Class | A screenshot of a computer  Description automatically generated | A component of the system with its attributes (things it knows) and operations (things it does) |
| Unidirectional Association | A diagram of a company  Description automatically generated | ClassA is aware of ClassB and its relationship to ClassB. ClassB is not aware of ClassA. |
| Bidirectional Association | A black background with white squares  Description automatically generated | ClassA and ClassB are aware of each other and the relationship between them. ClassA is associated with N instances of ClassB. ClassB is associated with M instances of ClassA. |
| Composition | A diagram of a diagram  Description automatically generated with medium confidence | ClassA is strongly a part of ClassB. The lifetime of ClassA depends on the lifetime of ClassB. |

Figure 3.2.2-1. Class Diagram for the TRACE system

# **Logger Class**

The logger class creates a log for the actions and decisions of the system based on how it reacts to the user’s inputs and actions. It generates a log report at the end of the project lifecycle.

[SRS 3.2.2.1.1] The Logger class shall store the attributes as defined in Table 3.2.2.1.-1.

**Table 3.2.2.1.-1: Attributes of the Logger Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | messages | Required; Editable; List | List containing the logs generated by the system while in use. |

[SRS 3.2.2.1.2] The Logger class shall generate a log for user actions.

[SRS 3.2.2.1.3] The Logger class shall append a user action to a log.

[SRS 3.2.2.1.4] The Logger class shall append system actions during the lifetime of the project.

[SRS 3.2.2.1.5] The Logger class shall save the logging results.

[SRS 3.2.2.1.6] The Logger class shall be able to display the logging results.

# **Machine Service Manager Class**

The machine service manager allows the analyst to select, reset, pause, resume, and cancel a service.

[SRS 3.2.2.2.1] The Machine Service Manager class shall store the attributes as defined in Table 3.2.2.3.-1:

**Table 3.2.2.3.-1: Recon Manager Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | Machines | Required; Editable; List | A list containing the IP address and port tuple to identify the machine being analyzed. |
| 2 | ServicesExecuted | Required; Editable; List | A list containing the services and their status. |
| 3 | machineStatus | Required; Editable; String | A string that specifies the status of a machine (add possible status. |

[SRS 3.2.2.2.2] The machine service manager shall get the system’s projects.

[SRS 3.2.2.2.3] The machine service manager shall reset the service values as specified by the service name.

[SRS 3.2.2.2.4] The machine service manager shall pause a service when a service name is provided.

[SRS 3.2.2.2.5] The machine service manager shall resume a paused service as specified by the analyst.

[SRS 3.2.2.2.6] The machine service manager shall cancel the service specified by the analyst.

[SRS 3.2.2.2.7] The machine service manager shall select a service specified by the analyst.

[SRS 3.2.2.2.8] The machine service manager shall load the machine results.

[SRS 3.2.2.2.9] The machine service manager shall save the machine results.

[SRS 3.2.2.2.10] The machine service manager shall store a list of the services executed and their status.

[SRS 3.2.2.2.11] The machine service manager shall store the machine status information.

[SRS 3.2.2.2.12] The machine service manager shall store the IP, port information to identify the machines.

[SRS 3.2.2.2.13] The machine service manager shall display the machine results.

# **Machines Class**

The machines class holds the IP list, the list of machines analyzed, and shall allow the analyst to select a machine and save the machine results.

[SRS 3.2.2.3.1] The machines class shall store the attributes as defined in Table 3.2.2.3.-1:

**Table 3.2.2.3.-1: Project Manager Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | IPList | Required; Editable; List | A list containing the tuples of IP and Port that identify the allowed machines. |
| 2 | machinesAnalyzed | Required; Editable; List | A list containing the machine IP and machine status to identify what machines have been analyzed. |

[SRS 3.2.2.3.2] The machines class shall store the list of all machines.

[SRS 3.2.2.3.3] The machines class shall store the list of the machines that have been analyzed.

[SRS 3.2.2.3.4] The machines class shall be able to select a machine using the machine’s IP address.

[SRS 3.2.2.3.5] The machines class shall be able to generate a report containing the project results.

# **Project Manager Class**

The project manager allows the analyst to manage the creation or deletion of a project. The analyst is able to create, load, lock, delete, export, or import a project.

[SRS 3.2.2.4.1] The Project Manager class shall store the attributes as defined in Table 3.2.2.4.-1:

**Table 3.2.2.4.-1: Project Manager Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | name | Required; Editable; String | A string containing the name of the project manager. |
| 2 | owner | Required; Editable; String | A string containing the owner of the project. |
| 3 | IPList | Required; Editable; List | A list containing the tuples of IP and Port that identify the allowed machines. |
| 4 | dateRange | Required; Editable; Date | A date specifying the range of time the project will be active. |

[SRS 3.2.2.4.2] The Project manager class shall store the name of the project.

[SRS 3.2.2.4.3] The project manager class shall create a project and return a boolean value based on the success of the project creation.

[SRS 3.2.2.4.4] The project manager class shall load a project specified by the name the analyst provides.

[SRS 3.2.2.4.5] The project manager class shall delete a project specified by a lead analyst given that the project is not locked.

[SRS 3.2.2.4.6] The project manager shall return a Boolean value when a project is deleted.

[SRS 3.2.2.4.7] The project manager shall export the project.

[SRS 3.2.2.4.8] The project manager shall import a project specified by the analyst.

[SRS 3.2.2.4.9] The project manager shall import the IP list.

[SRS 3.2.2.4.10] The project manager shall read the IP list.

[SRS 3.2.2.4.11] The project manager class shall lock a project specified by the name the lead analyst provides.

[SRS 3.2.2.4.12] The project manager shall make the project read-only to other analysts and preventing it from being erased when it is locked.

# **Project Class**

The project class stores all of the information of the project, including the project name, the owner, if it is locked or unlocked and the IP list.

[SRS 3.2.2.5.1] The Project class shall store the attributes as defined in Table 3.2.2.5.-1:

**Table 3.2.2.4.-1: Project Manager Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | name | Required; Editable; String | A string containing the name of the project. |
| 2 | id | Required; Editable; String | A string containing the ID of the project. |
| 3 | owner | Required; Editable; String | A string containing the initials of the owner of the project. |
| 4 | isLocked | Required; Editable; Boolean | A Boolean value to signify if the project is locked or not. |
| 5 | files | Required; Editable; Array | An array of strings that contains the files of data relating to the project. |
| 6 | IPList | Required; Editable; List | A list of tuples containing the IP/Port combination to identify each machine that the system will analyze. |

[SRS 3.2.2.5.2] The project class shall create a project with the project’s name, owner and id.

[SRS 3.2.2.5.3] The project class shall return a Boolean value when setting the project name.

[SRS 3.2.2.5.4] The project class shall set the owner with the name and the id.

[SRS 3.2.2.5.5] The project class shall set the Boolean value to determine if the project is locked.

[SRS 3.2.2.5.6] The project class shall unlock the project and set the Boolean value to False to demonstrate that the project is unlocked.

[SRS 3.2.2.5.7] The project class shall import the IP list specified by the analyst.

[SRS 3.2.2.5.8] The project class shall save the project.

# **HTTP Client Class**

[SRS 3.2.2.6.1] The HTTP Client Class shall store the attributes defined in Table 3.2.2.6.-1:

**Table 3.2.2.6.-1: The HTTP Client Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | proxy | Required; Editable; ProxyServer | A proxy server that will connect the TRACE system to the specified machines. |

[SRS 3.2.2.6.2] The HTTP Client Class shall establish the connection to the proxy server.

[SRS 3.2.2.6.3] The HTTP Client Class shall log a message for the actions of the system.

[SRS 3.2.2.6.3] The HTTP Client Class shall generate a log report from the log messages.

[SRS 3.2.2.6.3] The HTTP Client Class shall save the results for each machine that is analyzed.

[SRS 3.2.2.6.4] The HTTP Client Class shall display the results.

# **Proxy Server Class**

The Proxy server class allows the analyst to set a proxy and enable an intercept. It will also store the request and response histories.

[SRS 3.2.2.7.1] The Proxy Server Class shall store the attributes defined in Table 3.2.2.7.-1:

**Table 3.2.2.7.-1: The Proxy Server Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | request | Required; Editable; String | A string containing a request that will be sent to the machines being analyzed. |
| 2 | response | Required; Editable; String | A string containing the response from the machine that received the request. |
| 3 | requestHistory | Required; Editable; String Array | An array of strings containing the previously sent requests. |
| 4 | responseHistory | Required; Editable; String Array | An array of strings that contains the previous responses. |

[SRS 3.2.2.7.2] The proxy server class shall store the requests sent from the system to the machines.

[SRS 3.2.2.7.2] The proxy server class shall store the responses from the machine.

[SRS 3.2.2.7.2] The proxy server class shall set the proxy server to connect to the specified machines.

[SRS 3.2.2.7.2] The proxy server class shall enable the intercept to the machines.

[SRS 3.2.2.7.2] The proxy server class shall save the results of the connection.

[SRS 3.2.2.7.2] The proxy server class shall display the results of the connection.

# **DB Enumerator Class**

The DB enumerator class shall allow the analyst to enumerate the tables, columns, and responses from the database.

[SRS 3.2.2.8.1] The DB Enumerator Class shall store the attributes defined in Table 3.2.2.8.-1:

**Table 3.2.2.8.-1: The DB Enumerator Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | tableNames | Required; Editable; Array | An array of strings that contains the names of the tables in the database. |
| 2 | columnNames | Required; Editable; Array | An array of strings that contains the names of the columns of the database. |
| 3 | table | Required; Editable; Map | A table containing the information of the database. |

[SRS 3.2.2.8.2] The DB Enumerator Class shall store the tables from the database.

[SRS 3.2.2.8.3] The DB Enumerator Class shall store the names of the tables.

[SRS 3.2.2.8.4] The DB Enumerator Class shall store the names of the coumns.

[SRS 3.2.2.8.5] The DB Enumerator Class shall enumerate the database.

[SRS 3.2.2.8.6] The DB Enumerator Class shall enumerate the tables found within the database.

[SRS 3.2.2.8.7] The DB Enumerator Class shall enumerate the columns found within the database.

[SRS 3.2.2.8.8] The DB Enumerator Class shall process the response provided from the selected machines.

[SRS 3.2.2.8.9] The DB Enumerator Class shall save the results from the service.

[SRS 3.2.2.8.10] The DB Enumerator Class shall reset the service to start over or start with another database.

[SRS 3.2.2.8.10] The DB Enumerator Class shall display the results.

# **AI Wordlist Manager Class**

The AI wordlist manager allows the analyst to create, add, or remove a word list. The analyst can also save or reset a service.

[SRS 3.2.2.9.1] The AI Wordlist Manager Class shall store the attributes defined in Table 3.2.2.9.-1:

**Table 3.2.2.9.-1: The AI Wordlist Manager Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | namePassword | Required; Editable; List | A list generated by the artificial intelligence component that contains possible usernames and passwords. |

[SRS 3.2.2.9.2] The AI Wordlist Manager Class shall create a list of possible usernames and passwords.

[SRS 3.2.2.9.3] The AI Wordlist Manager Class shall add a wordlist provided by the analyst.

[SRS 3.2.2.9.4] The AI Wordlist Manager Class shall be able to remove a previously created username and password list.

[SRS 3.2.2.9.5] The AI Wordlist Manager Class shall save the results of creating the username and password list.

[SRS 3.2.2.9.6] The AI Wordlist Manager Class shall be able to reset the service.

[SRS 3.2.2.9.7] The AI Wordlist Manager Class shall display the results of creating the username and password list.

# **Directory Brute Forcer Class**

The Directory Brute Forcer class allows the analyst to execute the directory brute force service on the selected machines. With the information gathered from running the brute force, it builds the directory path.

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall store the attributes defined in Table 3.2.2.10.-1:

**Table 3.2.2.10.-1: The Directory Brute Forcer Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | dictionary | Required; Editable; String | A list containing the directories found by the system |
| 2 | DBFTree | Required; Editable; Tree | A tree created by the directory tree creator with the information found by the Directory Brute Forcer |

[SRS 3.2.2.10.2] The Directory Brute Forcer Class shall store the path of the machines found by the system.

[SRS 3.2.2.10.3] The Directory Brute Forcer Class shall store the tree generated from the paths found

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall set the dictionary with the machine path.

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall hide the status codes of the machine.

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall execute the brute force service on the machine’s directory.

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall process the response from the machine.

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall save the results of the brute forcing.

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall reset the service and clear the information.

[SRS 3.2.2.10.1] The Directory Brute Forcer Class shall display the results of the brute forcing.

# **Crawler Class**

The crawler class allows the analyst to crawl through the selected URL and create a path.

[SRS 3.2.2.11.1] The Crawler Class shall store the attributes defined in Table 3.2.2.11.-1:

**Table 3.2.2.11.-1: The Crawler Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | CrwTree | Required; Editable; Tree | A tree containing the path found by the crawler. |

[SRS 3.2.2.11.2] The Crawler Class shall store the tree generated by the crawler’s information.

[SRS 3.2.2.11.3] The Crawler Class shall set the configuration for the crawler service.

[SRS 3.2.2.11.4] The Crawler Class shall process the response received from the machines.

[SRS 3.2.2.11.5] The Crawler Class shall save the results of the crawler service.

[SRS 3.2.2.11.6] The Crawler Class shall reset the service.

[SRS 3.2.2.11.7] The Crawler Class shall display the results of the crawler service.

# **Directory Tree Creator Class**

The directory tree creator takes the information from the crawler and Directory Brute Forcer classes and builds a directory tree based on their paths.

[SRS 3.2.2.12.1] The Directory Tree Creator Class shall store the attributes defined in Table 3.2.2.12.-1:

**Table 3.2.2.12-1: The Directory Tree Creator Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | tree | Required; Editable; Tree | A tree containing the path generated by the information provided to it by the Directory Brute Forcer or the crawler. |

[SRS 3.2.2.12.2] The Directory Tree Creator Class shall store the tree generated by the machine’s information.

[SRS 3.2.2.12.3] The Directory Tree Creator Class shall add a found directory to the tree.

[SRS 3.2.2.12.4] The Directory Tree Creator Class shall print the tree as it is being built.

[SRS 3.2.2.12.5] The Directory Tree Creator Class shall allow the editing of the tree by editing the nodes.

[SRS 3.2.2.12.6] The Directory Tree Creator Class shall get the tree generated.

[SRS 3.2.2.12.7] The Directory Tree Creator Class shall save the results of the tree creation.

[SRS 3.2.2.12.8] The Directory Tree Creator Class shall reset the service and clear the information.

# **Authentication Attacker Class**

The authentication attacker allows the analyst to select the attack it shall perform on the machines. It also allows the analyst to select the parameters, set the word list, execute the brute force process, and process the response.

[SRS 3.2.2.13.1] The Authentication Attacker Class shall store the attributes defined in Table 3.2.2.13.-1:

**Table 3.2.2.13-1: The Authentication AttackerComponent**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | payload | Required; Editable; Array | An array of strings containing the payload from the system to the machine being analyzed. |
| 2 | wordList | Required; Editable; Array | An array containing a selected or generated wordlist. |
| 3 | attack | Required; Editable; String | A string containing the name of the attack that was performed. |
| 4 | InjectionPoints | Required; Editable; List | A list containing the injection points for a request. |

[SRS 3.2.2.13.2] The Authentication attacker Class shall select the attack to be performed on the machine.

[SRS 3.2.2.13.3] The Authentication attacker Class shall select the injection point of the machine.

[SRS 3.2.2.13.4] The Authentication attacker Class shall select the payload to inject into the base request.

[SRS 3.2.2.13.5] The Authentication attacker Class shall run the selected attack on the machine.

[SRS 3.2.2.13.6] The Authentication attacker Class shall select the parameters for the attack.

[SRS 3.2.2.13.7] The Authentication attacker Class shall set the selected word list.

[SRS 3.2.2.13.8] The Authentication attacker Class shall be able to execute the authentication brute force service on the machine.

[SRS 3.2.2.13.9] The Authentication attacker Class shall process the response provided by the machine.

[SRS 3.2.2.13.10] The Authentication attacker Class shall save the results of running the service.

[SRS 3.2.2.13.11] The Authentication attacker Class shall be able to reset the service and clear the unnecessary data.

[SRS 3.2.2.13.12] The Authentication attacker Class shall display the results of running the service.

# **Request Manager Class**

The repeater class allows the analyst to send, edit, and process the response of sending requests to the machines.

[SRS 3.2.2.14.1] The Request Manager Class shall store the attributes defined in Table 3.2.2.14.-1:

**Table 3.2.2.14-1: The Request Manger Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | request | Required; Editable; String | A string containing an HTTP request that is to be sent to the machine. |
| 2 | response | Required; Editable; String | A string containing the response from the machine. |

[SRS 3.2.2.14.2] The Request Manager Class shall send the request to the specified machine.

[SRS 3.2.2.14.3] The Request Manager Class shall be able to resend the same requests to the machine.

[SRS 3.2.2.14.4] The Request Manager Class shall be able to edit the request.

[SRS 3.2.2.14.5] The Request Manager Class shall show the history of sent requests.

[SRS 3.2.2.14.6] The Request Manager Class shall show the history of received responses.

[SRS 3.2.2.14.7] The Request Manager Class shall follow any redirection specified by the analyst.

[SRS 3.2.2.14.8] The Request Manager Class shall process the response from the machine.

[SRS 3.2.2.14.9] The Request Manager Class shall save the results from the HTTP requests.

[SRS 3.2.2.14.10] The Request Manager Class shall be able to reset the service and clear any unnecessary data.

[SRS 3.2.2.14.11] The Request Manager Class shall display the results of the HTTP requests.

# **SQL Injection Manager Class**

The SQL Injection class allows the analyst to perform or modify an SQL injection attack and process the response from the attacked machines.

[SRS 3.2.2.15.1] The SQL Injection Manager Class shall store the attributes defined in Table 3.2.2.15.-1:

**Table 3.2.2.15-1: The SQL Injection Manger Component**

|  |  |  |  |
| --- | --- | --- | --- |
| Req Number | Attribute | Values and Constraints | Description |
| 1 | request | Required; Editable; String | A string containing an injection request that is to be sent to the machine. |
| 2 | response | Required; Editable; String | A string containing the response from the attacked machine |

[SRS 3.2.2.15.2] The SQL Injection Manager Class shall store the requests sent to the target machine.

[SRS 3.2.2.15.3] The SQL Injection Manager Class shall store the responses received from the target machine.

[SRS 3.2.2.15.4] The SQL Injection Manager Class shall perform the SQL injection attack on the selected machine.

[SRS 3.2.2.15.5] The SQL Injection Manager Class shall edit the SQL injection request.

[SRS 3.2.2.15.6] The SQL Injection Manager Class shall show the history of the sent requests.

[SRS 3.2.2.15.7] The SQL Injection Manager Class shall show the history of the received responses.

[SRS 3.2.2.15.8] The SQL Injection Manager Class shall process the response from the targetted machines.

[SRS 3.2.2.15.9] The SQL Injection Manager Class shall save the results of the SQL injection service.

[SRS 3.2.2.15.10] The SQL Injection Manager Class shall be able to reset the service and clear any unnecessary data.

# **Stimulus**

In TRACE, these stimuli are essential for defining how the system interacts with external inputs, processes data, and generates outputs to meet its operational goals. Within TRACE, various stimuli initiate tasks such as directory crawling, credential brute-forcing, HTTP request modifications, and SQL injection analysis. By systematically organizing and responding to these stimuli, TRACE ensures efficient data aggregation, vulnerability identification, and user interaction through its structured workflows and dynamic model representations, including state and data flow diagrams. This section outlines the key stimuli influencing TRACE’s operations, including their corresponding system responses and dependencies. Detailed scenarios such as the user interfaces, the introduction of malformed HTTP requests, and other real-world triggers are explored to illustrate TRACE's response mechanisms and ensure alignment with cybersecurity assessment objectives.

**[SRS 3.2.3-1]** If a directory crawling task is initiated, the system shall recursively traverse the specified directory, aggregate file metadata, and log the results in a structured format for further analysis.

**[SRS 3.2.3-2]** If the system receives malformed HTTP requests, TRACE shall analyze the requests for potential vulnerabilities and log any relevant details.

**[SRS 3.2.3-3]** If a credential brute-forcing process is triggered, the system shall attempt to systematically test the provided credentials within the AI generated file.

**[SRS 3.2.3-4]** If an SQL Injection pattern is detected in the provided input, the system shall execute a payload attack to attempt and takeover a target database.

**[SRS 3.2.3-5]** If a vulnerability is identified during an analysis task, TRACE will log the details in its log report and notify the analysts.

**[SRS 3.2.3-6]** If an analyst creates a new project, the system shall prompt for a project name and any other required configuration information.

**[SRS 3.2.3-7]** If an analyst selects a specific tool for scanning, the system shall load the configuration for the corresponding tool and block any other scan from running until the current scan has completed.

**[SRS 3.2.3-8]** If an analyst imports a project, then the system shall verify the integrity of the project, and load all associated files and configurations, while updating the interface to reflect the status of the imported project.

**[SRS 3.2.3-9]** If an analyst exports a project, then the system shall package all relevant project files, such as the results and the logs, to the specified format to ensure compatibility, by importing files into TRACE.

**[SRS 3.2.3-10]** If an analyst views an analysis graph, then the system shall dynamically generate a graphical representation of the project’s results.

**[SRS 3.2.3-11]** If an analyst modifies the configuration of a project, (e.g. scanning parameters or tool configurations), the system shall update the project metadata and ensure the following scans follow the updated settings.

**[SRS 3.2.3-12]** If an analyst exports analysis results, the system shall generate a report in the specified format, to include vulnerability details, and prompt the user for a destination file.

**[SRS 3.2.3-13]** If an analyst permanently deletes a project, the system will prompt for confirmation and securely remove all associated files and metadata for the deleted project.

**[SRS 3.2.3-14]** If an analyst deletes a project, then the project will move from the roster of projects to select from, to the roster of Recently Deleted projects, where they have the option to restore or permanently delete a project.

**[SRS 3.2.3-15]** If a Recently Deleted Project is restored, then the project is placed back into the Project Selection dashboard.

**[SRS 3.2.3-16]** If an analyst schedules a scan task, the system shall store the task in a scheduler, and trigger it at a specified time, or as soon as it may, alerting the user when it completes the scan, and provide a summary with the results.

**[SRS 3.2.3-17]** If an analyst pauses an ongoing scan, the system shall halt the scanning process without losing any progress and allow the user to continue from where the scan stopped or cancel it directly

**[SRS 3.2.3-18]** If an analyst resumes a paused scan, the system will continue from where the scan left off when it had completed the previous state.

**[SRS 3.2.3-19]** If an analyst views the scan logs, the system shall display the logs, allowing the analyst to perform further analysis.

**[SRS 3.2.3-20]** When an analyst prompts the AI to generate the credential list, the system will provide all relevant information (e.g scan results, vulnerabilities found) to the model to rank classifications into “successful enough” combinations.

# **User Interfaces**

1. **Landing Page**

**[SRS 3.2.3.1 – 1a]** When selecting the "Start" and “Dashboard” button, the system shall navigate the user to the "Project Selection" page. (as shown in Section 3.2.3.1.3)

**[SRS 3.2.3.1- 1b]** When selecting the "Settings" button, the system shall display the system configuration window, allowing users to adjust preferences and system settings.

1. **Project Dashboard Sidebar**

**[SRS 3.2.3.1- 2a]** When selecting the "Folder" icon, the system shall navigate the user to the "Project Selection" page, where they can view, create, import, or open projects. (as shown in Section 3.2.3.1.3)

**[SRS 3.2.3.1- 2b]** When selecting the "Directory" icon, the system shall navigate the user to the "Project Folders" page, displaying project folders. (as shown in Section 3.2.3.1.5)

**[SRS 3.2.3.1- 2c]** When selecting the "Trash Bin" icon, the system shall navigate the user to the "Deleted Projects" page, where previously deleted projects are listed with options to restore or permanently delete them. (as shown in Section 3.2.3.1.6)

**[SRS 3.2.3.1- 2d]** When selecting the "Settings" icon, the system shall display the "Settings" window, allowing users to configure system preferences and adjust operational parameters.

**[SRS 3.2.3.1- 2e]** When selecting the "TRACE" icon, the system shall navigate the user to the "Project Selection" page again. (as shown in Section 3.2.3.1.3)

1. **Project Selection – My Projects**

**[SRS 3.2.3.1- 3a]** When selecting the "Create New" button, the system shall display the "Create New Project" window, allowing the user to configure a project name, description, and other relevant metadata. (as shown in Section 3.2.3.1.7)

**[SRS 3.2.3.1- 3b]** When selecting any project card in the "Recent Projects" section, the system shall navigate the user to the “Tool Dashboard” page. (as shown in Section 3.2.3.1.38)

**[SRS 3.2.3.1- 3c]** When selecting the "Run Scan" button for any project in the list, the system shall initiate a scan for the selected project and display the "Tool Dashboard" page. (as shown in Section 3.2.3.1.38)

1. **Project Selection – Shared Projects**

**[SRS 3.2.3.1- 4a]** When selecting the "Create New" button, the system shall display the "Create New Project" window, allowing the user to define a project name, description, and other relevant metadata. (as shown in Section 3.2.3.1.7)

**[SRS 3.2.3.1- 4b]** When selecting any project card in the "Recent Projects" section, the system shall navigate the user to the “Tool Dashboard” page. (as shown in Section 3.2.3.1.38)

**[SRS 3.2.3.1- 4c]** When selecting the "Join" button for any shared project in the list, the system shall add the user to the project and provide access to its “Tool Dashboard” page. (as shown in Section 3.2.3.1.38)

1. **Project Folders**

**[SRS 3.2.3.1- 5a]** When selecting the "Create New" button, the system shall display the "Create New Folder" window, allowing the user to define the folder name, description, and other relevant metadata.

**[SRS 3.2.3.1- 5b]** When selecting any folder in the "Recent Folders" section, the system shall open the corresponding project folder for the user to view its contents.

**[SRS 3.2.3.1- 5c]** When selecting the "Join" button for any shared project in the list, the system shall add the user to the project and provide access to its “Tool Dashboard” page. (as shown in Section 3.2.3.1.38)

1. **Deleted Projects**

**[SRS 3.2.3.1- 6a]** When selecting "Restore" from the "More Options" menu, the system shall display a confirmation dialog to restore the selected project to its original state in the "Project Selection" page. (as shown in Section 3.2.3.1.3)

1. **Create Project**

**[SRS 3.2.3.1- 7a]** When the "Create" button is clicked, the system shall create a new project and display the “Tool Dashboard” page. (as shown in Section 3.2.3.1.38)

**[SRS 3.2.3.1- 7b]** When the "Cancel" button and “X” icon is clicked, the system shall close the "Create Project" modal without saving any entered information and display the “Project Selection” page. (as shown in Section 3.2.3.1.3)

1. **Import Project**

**[SRS 3.2.3.1- 8a]** When a user clicks the file zone, the system shall open the file explorer, allowing the user to select a file manually.

**[SRS 3.2.3.1- 8b]** When the “X” icon is clicked, the system shall close the "Import Project" modal and display the “Project Selection” page. (as shown in Section 3.2.3.1.3)

1. **HTTP Tester Configuration**

**[SRS 3.2.3.1- 9a]** When the “Start” button is selected, the system shall initiate the HTTP testing process using the specified parameters and navigate to the “Running” page. (as shown in Section 3.2.3.1-10)

1. **HTTP Tester Run Scan**

**[SRS 3.2.3.1– 10a]** When selecting the ‘Pause’ button, the system shall temporarily halt the HTTP testing process, preserving the current progress and allowing the user to resume from the same state at a later time.

**[SRS 3.2.3.1- 10b]** When selecting the “Restart” button, the system shall reset the test parameters to the default values and allow the user to reconfigure the HTTP test. (as shown in Section 3.2.3.1.9)

**[SRS 3.2.3.1– 10c]** When selecting the “Stop” button, the system shall terminate the test execution and navigate to the “Results” page to display the incomplete results. (as Shown in Section 3.2.3.1.12)

**[SRS 3.2.3.1- 10d]** When selecting the ‘Modify’ button, the system shall navigate to the ‘Repeater’ page, preserving the previously entered settings for further modification. (as shown in Section 3.2.3.1.11)

1. **HTTP Tester Repeater**

**[SRS 3.2.3.1– 11a]** When selecting the ‘Pause’ button, the system shall temporarily halt the HTTP testing process, preserving the current progress and allowing the user to resume from the same state at a later time.

**[SRS 3.2.3.1- 11b]** When selecting the “Restart” button, the system shall reset the test parameters to the default values and allow the user to reconfigure the HTTP test. (as shown in Section 3.2.3.1.9)

**[SRS 3.2.3.1– 11c]** When selecting the “Stop” button, the system shall terminate the test execution and navigate to the “Results” page to display the incomplete results. (as shown in Section 3.2.3.1.12)

**[SRS 3.2.3.1- 11d]** When selecting the “Update” button (circular icon at the bottom of the Repeater page), the system shall refresh the current configuration and results table by reapplying the updated parameters or payloads entered by the user. Any changes made to the configuration will immediately reflect in the displayed results without restarting the scan.

1. **HTTP Tester Results**

**[SRS 3.2.3.1- 12a]** When selecting the “Export” button, the system shall open a modal dialog (pop-up screen) from the computer’s file explorer, allowing the user to choose the desired file location and name for saving the test results. If the location is confirmed, the system shall save the results in the selected directory in a standardized format for external use.

**[SRS 3.2.3.1- 12b]** When selecting the “Show Terminal” button, the system shall display a terminal window showing the raw output of the HTTP test for advanced users.

**[SRS 3.2.3.1- 12c]** When selecting the “Restart” button, the system shall clear the current results and navigate back to the “Configuration” page for a new test setup. (as shown in Section 3.2.3.1.9)

1. **Intruder Configuration Page**

**[SRS 3.2.3.1- 13a]** When the “Start” button is clicked, the system shall validate the entered parameters and begin the intruder testing process, navigating to the “Running” page. (as shown in Section 3.2.3.1.14)

1. **Intruder Running Page**

**[SRS 3.2.3.1– 14a]** When selecting the “Pause” button, the system shall temporarily halt the scanning process, preserving the current progress. The user shall have the option to resume the scan at a later time without losing any data or configuration.

**[SRS 3.2.3.1– 14b]** When selecting the “Restart” button, the system shall clear all current scan progress and results, returning the user to the “Configuration” page to set up and initiate a new scan.( as shown in Section 3.2.3.1.13)

**[SRS 3.2.3.1– 14c]** When selecting the “Stop” button, the system shall terminate the test execution and navigate to the “Results” page to display the incomplete results. (as shown in Section 3.2.3.1.16)

**[SRS 3.2.3.1- 14d]** When the “Modify” button is selected, the system shall allow users to adjust parameters and re-run tests using updated configurations. (as shown in Section 3.2.3.1.15)

1. **Intruder Repeater Page**

**[SRS 3.2.3.1- 15a]** When selecting the ‘Pause’ button, the system shall temporarily halt the Intruder testing process, preserving the current progress and allowing the user to resume from the same state at a later time.

**[SRS 3.2.3.1- 15b]** When selecting the “Update” button (circular icon at the bottom of the Repeater page), the system shall refresh the current configuration and results table by reapplying the updated parameters or payloads entered by the user. Any changes made to the configuration will immediately reflect in the displayed results without restarting the scan.

**[SRS 3.2.3.1- 15c]** When selecting the “Restart” button, the system shall clear all current scan progress and results, returning the user to the “Configuration” page to set up and initiate a new scan.( as shown in Section 3.2.3.1.13)

**[SRS 3.2.3.1– 15d]** When selecting the “Stop” button, the system shall terminate the test execution and navigate to the “Results” page to display the incomplete results. (as shown in Section 3.2.3.1.16)

1. **Intruder Results Page**

**[SRS 3.2.3.1- 16a]** When selecting the “Export” button, the system shall generate a file containing the test results for download.

**[SRS 3.2.3.1- 16b]** When selecting the “Show Terminal” button, the system shall provide a raw output log of the test results for advanced analysis.

**[SRS 3.2.3.1- 16c]** If the “Restart” button is clicked, the system shall clear all results and return to the “Configuration” page for a new setup. (as shown in Section 3.2.3.1.13)

1. **SQL Injection Configuration**

**[SRS 3.2.3.1- 17a]** When the “Start” button is clicked, the system shall validate the entered parameters and initiate the SQL Injection testing process, navigating to the “Scan” page. (as shown in Section 3.2.3.1.19)

1. **SQL Injection Scan Page**

**[SRS 3.2.3.1- 18a]** When selecting the “Pause” button, the system shall temporarily halt the scan while retaining the current state.

**[SRS 3.2.3.1- 18b]** When selecting the “Stop” button, the system shall terminate the scan and preserve partial results for user review on the “Results” page. (as shown in Section 3.2.3.1.21)

**[SRS 3.2.3.1- 18c]** When selecting the “Restart” button, the system shall clear all current scan progress and results, returning the user to the “Configuration” page to set up and initiate a new scan. (as shown in Section 3.2.3.1.18)

**[SRS 3.2.3.1- 18d]** When selecting the “Show Terminal” button, the system shall provide a raw output log of the test results for advanced analysis.

1. **SQL Injection Run Page**

**[SRS 3.2.3.1- 19a]** When selecting the “Restart” button, the system shall clear all results and navigate back to the “Configuration” page for a new test. (as shown in Section 3.2.3.1.18)

**[SRS 3.2.3.1- 19b]** When selecting the “Export” button, the system shall generate a downloadable file containing the full test results.

**[SRS 3.2.3.1- 19c]** When selecting the “Show Terminal” button, the system shall provide a raw output log of the test results for advanced analysis.

1. **SQL Injection Result Page**

**[SRS 3.2.3.1- 20a]** When selecting the “Restart” button, the system shall clear all results and navigate back to the “Configuration” page for a new test. (as shown in Section 3.2.3.1.18)

**[SRS 3.2.3.1- 20b]** When selecting the “Export” button, the system shall generate a downloadable file containing the full test results.

**[SRS 3.2.3.1– 20c]** When selecting the “Show Terminal” button, the system shall display a raw log output of the SQL Injection test for advanced analysis.

1. **Parameter Fuzzing Configurations**

**[SRS 3.2.3.1- 21a]** When the “Start” button is clicked, the system shall validate the entered parameters and initiate the parameter fuzzing process, navigating to the “Scan” page. (as shown in Section 3.2.3.1.23)

1. **Parameter Fuzzing Scan Page**

**[SRS 3.2.3.1- 22a]** When selecting the “Pause” button, the system shall temporarily halt the fuzzing process without losing the current progress.

**[SRS 3.2.3.1- 22b]** When selecting the “Stop” button, the system shall terminate the scan and preserve partial results for user review on the “Results” page. (as shown in Section 3.2.3.1.24)

**[SRS 3.2.3.1– 22c]** When selecting the “Restart” button, the system shall clear all results and navigate back to the “Configuration” page for a new test. (as shown in Section 3.2.3.1.22)

**[SRS 3.2.3.1– 22d]** When selecting the “Show Terminal” button, the system shall display a raw log output of the Parameter Fuzzing test for advanced analysis.

1. **Parameter Fuzzing Result Page**

**[SRS 3.2.3.1- 23a]** When selecting the “Restart” button, the system shall clear all results and navigate back to the “Configuration” page for a new test. (as shown in Section 3.2.3.1.22)

**[SRS 3.2.3.1- 23b]** When selecting the “Export” button, the system shall generate a downloadable file containing the test results in a standardized format.

**[SRS 3.2.3.1– 23c]** When selecting the “Show Terminal” button, the system shall display a raw log output of the Parameter Fuzzing test for advanced analysis.

1. **Parameter Fuzzing Scan with Terminal Overlay**

**[SRS 3.2.3.1- 24a]** When selecting the “Pause” button, the system shall temporarily halt the fuzzing process, preserving the current progress. The user shall have the option to resume the process from the same point without losing any data or configurations. (as shown in Section 3.2.3.1.23)

**[SRS 3.2.3.1- 24b]** When selecting the “Stop” button, the system shall terminate the ongoing fuzzing process and preserve any partial results. The user shall be directed to the “Results” page to review the completed portion of the scan. (as shown in Section 3.2.3.1.24)

**[SRS 3.2.3.1– 24c]** When selecting the “Restart” button, the system shall clear all current progress and results. The user shall be navigated back to the “Configuration” page to reconfigure and initiate a new parameter fuzzing test. (as shown in Section 3.2.3.1.22)

**[SRS 3.2.3.1– 24d]** When selecting the “Close Terminal” button, the system shall hide the terminal overlay, returning the user to the main “Scan” page view. The fuzzing process shall continue uninterrupted, and the user can reopen the terminal overlay as needed for advanced analysis.

1. **Brute Force Configurations**

**[SRS 3.2.3.1- 25a]** When the “Start” button is clicked, the system shall validate the entered parameters and initiate the brute force process, navigating to the “Scan” page. (as shown in Section 3.2.3.1.27)

1. **Brute Force Scan Page**

**[SRS 3.2.3.1- 26a]** When the “Pause” button is clicked, the system shall temporarily halt the brute force test while retaining the current state.

**[SRS 3.2.3.1– 26b]** When the “Stop” button is clicked, the system shall terminate the brute force process and allow users to review partial results on the “Result” page. (as shown in Section 3.2.3.1.28)

**[SRS 3.2.3.1– 26c]** When the “Restart” button is clicked, the system shall clear all current progress and results. The user shall be navigated back to the “Configuration” page to reconfigure and initiate a new Brute Force Scan. (as shown in Section 3.2.3.1.26)

**[SRS 3.2.3.1- 26d]** When selecting the “Show Terminal” button, the system shall display a raw log of the brute force process, including detailed request and response data.

1. **Brute Force Results Page**

**[SRS 3.2.3.1- 27a]** When the “Restart” button is clicked, the system shall clear all current progress and results. The user shall be navigated back to the “Configuration” page to reconfigure and initiate a new Brute Force scan. (as shown in Section 3.2.3.1.26)

**[SRS 3.2.3.1- 27b]** When the “Export” button is clicked, the system shall generate a downloadable file containing the brute force test results in a standardized format.

**[SRS 3.2.3.1– 27c]** When selecting the “Show Terminal” button, the system shall provide a raw output log of the test, including all sent requests and received responses.

1. **Crawler Configuration**

**[SRS 3.2.3.1- 28a]** When the “Start” button is clicked, the system shall validate the entered configuration parameters and initiate the crawler process, navigating to the “Scan” page. (as shown in Section 3.2.3.1.30)

1. **Crawler Scan Page**

**[SRS 3.2.3.1- 29a]** When the “Pause” button is clicked, the system shall temporarily halt the crawler process while retaining the current progress.

**[SRS 3.2.3.1– 29b]** When the “Stop” button is clicked, the system shall terminate the crawling process and allow users to review the partial results on the “Result” page. (as shown in Section 3.2.3.1.31)

**[SRS 3.2.3.1- 29c]** When the “Restart” button is clicked, the system shall clear all current progress and results. The user shall be navigated back to the “Configuration” page to reconfigure and initiate a new Crawler scan. (as shown in Section 3.2.3.1.29)

**[SRS 3.2.3.1- 29d]** When selecting the “Show Terminal” button, the system shall display a raw log of the crawling process, showing each request and response in real-time.

1. **Crawler Results Page**.

**[SRS 3.2.3.1- 30a]** When the “Restart” button is clicked, the system shall clear all current progress and results. The user shall be navigated back to the “Configuration” page to reconfigure and initiate a new Crawler scan. (as shown in Section 3.2.3.1.29)

**[SRS 3.2.3.1- 30b]** When the “Export” button is clicked, the system shall generate a downloadable file containing the crawl results in a standardized format.

**[SRS 3.2.3.1- 30c]** When selecting the “Show Terminal” button, the system shall provide raw output logs of the crawl process, including all sent requests and received responses.

1. **Tool Results**

**[SRS 3.2.3.1- 31a]** When selecting the "view" button of any of the result widgets displayed on screen, the system shall close the "Results" page and open the page associated with the corresponding widget. For example, if the user clicks to view the SQL Injection results, the system will route the user to the page where those results reside (as shown in Section 3.2.3.1.21)

1. **Tree Graph Directories**

**[SRS 3.2.3.1- 32a]** When selecting the "view" button of a parent directory node, the system shall close the "Tree Graph" page and open the "Tree Graph View" page associated with the selected parent directory node. (as shown in Section 3.2.3.1.34)

1. **Tree Graph View**

**[SRS 3.2.3.1- 33a]** When selecting the “arrow” button at the top left of the page, the system shall route the user back to the “Tree Graph Directories” page (as shown in Section 3.2.3.1.33)

**[SRS 3.2.3.1- 33b]** When selecting a node on display, the system shall route the user back to the “Tree Graph Database” page (as shown in Section 3.2.3.1.35)

1. **Tree Graph Database**

**[SRS 3.2.3.1- 34a]** When selecting the “X” icon on the top left of the modal view, the system shall route the user back to the “Tree Graph View” page (as shown in Section 3.2.3.1.34)

1. **AI Setup**

**[SRS 3.2.3.1- 35a]** When selecting the "Generate" button, the system shall route the user to the "AI Results” page and the “Tree Graph View” page to show the generated usernames and passwords (as shown in Section 3.2.3.1.36)

1. **AI Results**

**[SRS 3.2.3.1- 36a]** When selecting the Save Wordlist button, the system shall save the generated results into the machine's designated file location. If the file location has not been designated beforehand, then the system will open the file explorer, so the user shall configure where to save the generated wordlist

**[SRS 3.2.3.1- 36b]** When selecting the "regenerate" button, the system shall regenerate a wordlist to display to the user, using the same configurations input in the "AI Setup" page (as shown in Section 3.2.3.1.33)

1. **Tool Sidebar**

**[SRS 3.2.3.1- 37a]** When selecting the hammer icon button called "Tools Dashboard", the system shall navigate the user to the "Tools Dashboard" page (as shown in Section 3.2.3.1.38)

**[SRS 3.2.3.1- 37b]** When selecting the tree icon button called "Tree Graph", the system shall navigate the user to the "Tree Graph Directories" page (as shown in Section 3.2.3.1.33)

**[SRS 3.2.3.1- 37c]** When selecting the file with the checkmark icon called "Results", the system shall navigate the user to the page called "Tool Results" (as shown in Section 3.2.3.1.32)

**[SRS 3.2.3.1- 37d]** When selecting the brain icon button called "AI Credential Generator", the system shall navigate the user to the "AI Setup" page (as shown in Section 3.2.3.1.35)

1. **Tool Dashboard**

**[SRS 3.2.3.1- 38a]** When selecting the "Set Up" button, the system shall display a pop-up screen detailing the configuration fields for the corresponding tool for future scans.

**[SRS 3.2.3.1- 38b]** When selecting the "View" button, the system shall navigate the user to the complete scan results corresponding to the selected tool. (as shown in Section 3.2.3.1.23)

# **State Transitions**

The system states define the different ways the system behaves by responding to specific events when certain conditions are met. The available functionality and the way the system provides the functionality may change based on the state of the system at that moment.

The legend of the state transition diagrams is defined in Table 3.2.4-1: State Transition Diagram Legend.

**Table 3.2.4.1: State Transition Diagram Legend**

|  |  |  |
| --- | --- | --- |
| NAME | NOTATION | DESCRIPTION |
| State | A screenshot of a computer  Description automatically generated | Defines the system's behavior at the given point in time. It may include entry or exit action and an activity. |
| Final State | A screenshot of a computer  Description automatically generated | Start of a state transition diagram. |
| Initial State | A screenshot of a computer  Description automatically generated | End of a state transition diagram. |
| Transition | A screenshot of a computer  Description automatically generated | State transition from **State A** to **State B** may occur when **Event A** occurs, and the **Condition** (optional Boolean expression) is satisfied. The action (optional activity) is performed only if the transition is done. |

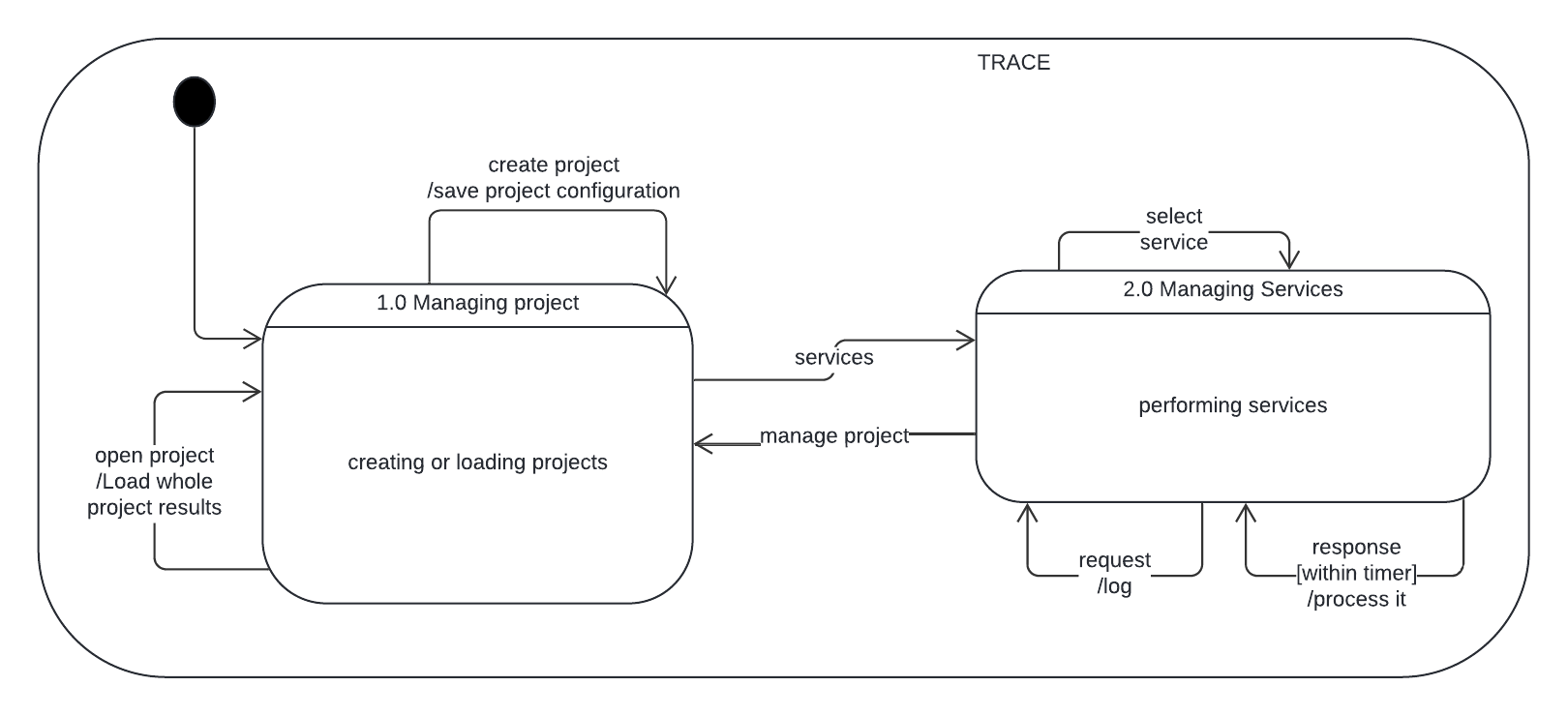


Figure 3.2.4.1-1 TRACE State Transition Diagram.

[SRS 3.2.4.1.1] TRACE shall include the states Managing Project and Managing Services.

[SRS 3.2.4.1.2] Managing project shall be the start state.

[SRS 3.2.4.1.3] When in the Managing Project state, if a new project is created, then the system shall save the project configuration.

[SRS 3.2.4.1.4] When in the managing project state, if a project is opened then the whole project results shall be loaded.

[SRS 3.2.4.1.5] When in the managing project state, the system shall create or load projects.

[SRS 3.2.4.1.6] When in the managing project state, if the services are provided by the analyst, the system shall shift to the Managing Services state.

[SRS 3.2.4.1.7] When in the Managing services state the analyst shall be able to select the services to be performed.

[SRS 3.2.4.1.8] When in the Managing Services state, the system shall create a log for every request.

[SRS 3.2.4.1.9] When in the Managing Services state, the TRACE system shall process every response it receives within a specified timer.

[SRS 3.2.4.1.10] When in the Managing Services state, the system shall perform the services specified.

[SRS 3.2.4.1.11] When in the Managing Services state, when the analyst selects to manage project, the system shall move to the managing project state.

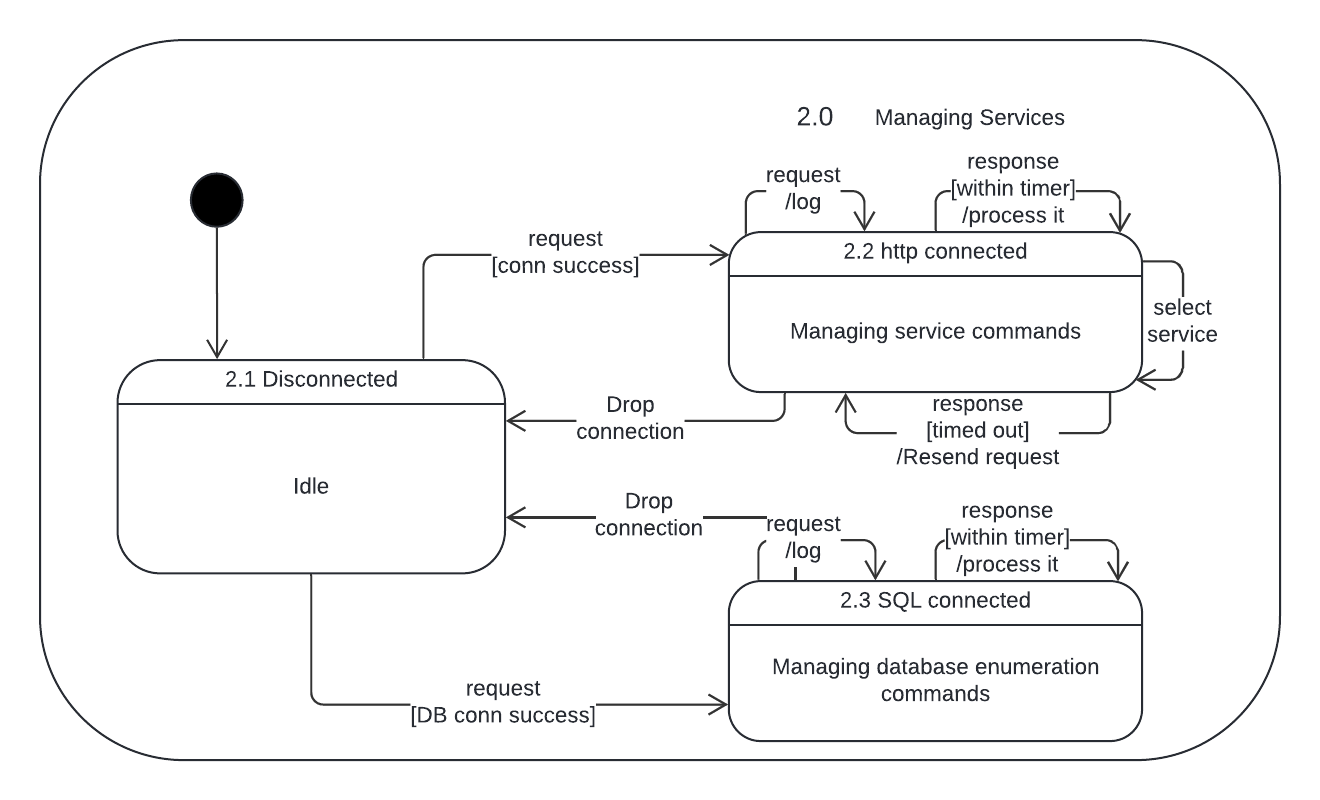


Figure 3.2.4.1-2 TRACE State Transition Diagram – Level 2.

[SRS 3.2.4.1.2.1] The Managing Services state shall include the states Disconnected, HTTP connected, and SQL connected.

[SRS 3.2.4.1.2.2] The Disconnected state shall be the start state.

[SRS 3.2.4.1.2.3] When in the disconnected state, if the analyst sends a request and the HTTP connection was a success the TRACE system shall move to the state HTTP connected.

[SRS 3.2.4.1.2.4] When in the disconnected state, if the analyst sends a request and the database connection was a success the TRACE system shall move to the state SQL connected.

[SRS 3.2.4.1.2.5] When in the HTTP connected state, if the connection is dropped then the system shall move to the disconnected state.

[SRS 3.2.4.1.2.6] When in the HTTP connected state, the system shall manage the service commands provided by the analyst.

[SRS 3.2.4.1.2.7] When in the HTTP connected state, the system shall create a log for every request.

[SRS 3.2.4.1.2.8] When in the HTTP connected state, the system shall process every response received during the specified time.

[SRS 3.2.4.1.2.9] When in the HTTP connected state, the system shall resend a request if a response is sent after it has timed out.

[SRS 3.2.4.1.2.10] When in the HTTP connected state, the system shall select the services specified by the analyst.

[SRS 3.2.4.1.2.11] When in the SQL connected state, the system shall create a log for all requests sent.

[SRS 3.2.4.1.2.12] When in the SQL connected state, the system shall process the responses sent within the specified time.

[SRS 3.2.4.1.2.13] When in the SQL connected state, the system shall manage the database enumeration commands.

[SRS 3.2.4.1.2.14] When in the SQL connected state, if the connection is dropped then the system shall move to the disconnected state.

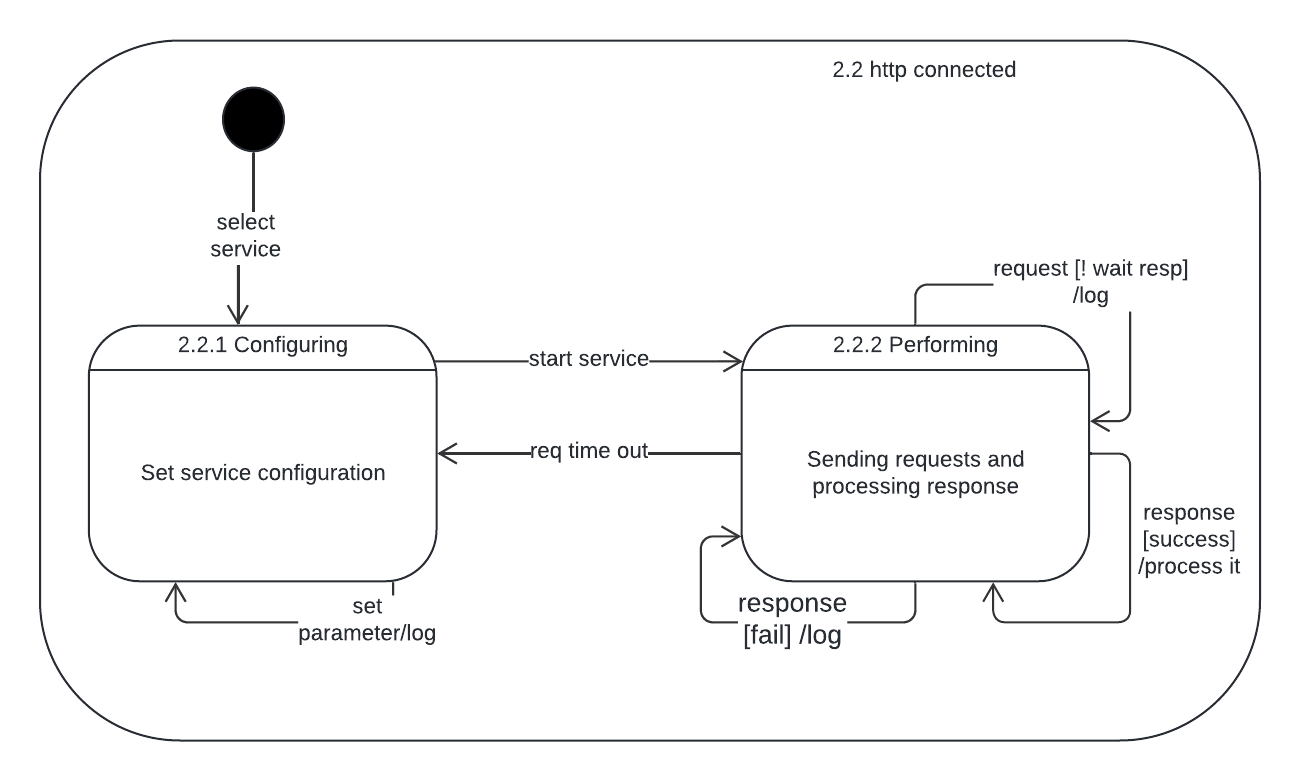


Figure 3.2.4.1-3 HTTP Connected State Transition Diagram – Level 3.

[SRS 3.2.4.1.3.1] The HTTP connected state shall contain the states Configuring and Performing.

[SRS 3.2.4.1.3.2] The Configuring state shall be the start state.

[SRS 3.2.4.1.3.3] When the service is selected the system shall move into the Configuring state.

[SRS 3.2.4.1.3.4] When in the Configuring state, when a parameter is set the system shall create a log message to track it.

[SRS 3.2.4.1.3.5] When in the Configuring state, the system shall set the service configuration.

[SRS 3.2.4.1.3.6] When in the Configuring state, the system shall log the parameters being set.

[SRS 3.2.4.1.3.7] When in the Configuring state, if the analyst selects to start a service, then the system shall move to the Performing state.

[SRS 3.2.4.1.3.8] When in the Performing state, the system shall send requests to the machines and process the response.

[SRS 3.2.4.1.3.9] When in the Performing state, if the analyst sends a request and does not wait for a response then the system shall log the request.

[SRS 3.2.4.1.3.10] When in the Performing state, if the system receives a successful response, then the system shall create a log for it and process it.

[SRS 3.2.4.1.3.11] When in the Performing state, if the system receives a failed connection, then the system shall log the failure.

[SRS 3.2.4.1.3.12] When in the Performing state, if the request times out then the system moves to the Configuring state.

# **Non-behavioral Requirements / Quality Attributes**

This section describes performance, availability, security, maintainability, and portability requirements. The performance requirements give a qualitative description of the system's process execution time.

# **Performance Requirements**

[SRS 3.3.1.1] TRACE shall support up to 4 concurrent analysts, each working on distinct or shared projects.

[SRS 3.3.1.2] TRACE shall support up to 4 active terminals or devices for internal network scanning and data aggregation per session.

[SRS 3.3.1.3] TRACE shall process requests such as scan updates, data retrieval, and UI interactions.

[SRS 3.3.1.4] TRACE shall process requests and responses within 5 seconds during peak usage.

[SRS 3.3.1.5] TRACE shall process 50 IP:PORT combinations per minute during network scans under normal conditions.

[SRS 3.3.1.6] TRACE shall generate a list of plausible usernames and passwords for the machines being analyzed.

[SRS 3.3.1.7] The Tree graph representation shall update with new information every second during normal scanning operations.

# **Availability**

[SRS 3.3.2.1] TRACE shall create automatic checkpoints during scans to save progress.

[SRS 3.3.2.2] TRACE shall allow the analysts to configure the intervals for the checkpoints or create checkpoints on demand.

[SRS 3.3.2.3] In case of a crash, or unexpected loss of power, TRACE shall recover to the last saved checkpoint before the loss of power.

[SRS 3.3.2.4] TRACE shall restart within a minute after an unexpected shut down.

[SRS 3.3.2.5] TRACE shall resume scanning operations from the last checkpoint without requiring reconfiguration.

[SRS 3.3.2.6] TRACE shall notify the analyst if the database is corrupted.

[SRS 3.3.2.7] TRACE shall initiate a safe restoration of the database if it is corrupted using backup copies.

# **Security**

This section specifies the factors that shall protect the software from accidental or malicious access, use, modification, destruction, or disclosure.

[SRS 3.3.3.1] TRACE shall verify validity periods on all application messages using WS-Security or SAML assertions to prevent the use of outdated security tokens.

[SRS 3.3.3.2] TRACE shall maintain system-activity logs, including access to sensitive functions, and modification of system configurations.

[SRS 3.3.3.3] TRACE shall allow analysts to view activity logs.

[SRS 3.3.3.4] TRACE shall use role-based access control (RBAC) mechanisms to ensure that users can only access functionalities and data that are relevant to their roles and responsibilities.

[SRS 3.3.3.5] TRACE shall employ algorithms or checksums to verify the integrity of data at various checkpoints throughout the system's operation.

[SRS 3.3.3.6] TRACE shall meet the requirements of the 32 CAT I STIG vulnerabilities that are applicable and listed in <https://www.stigviewer.com/stig/application_security_and_development/> (see Table 3.3.3-1 for applicable vulnerabilities).

[SRS 3.3.3.6] The logs shall include timestamps and user actions to ensure traceability and prevent misuse.

[SRS 3.3.3.6] Sessions shall time out after a defined period of inactivity to enhance the system’s security.

# **Maintainability**

[SRS 3.3.4.1] TRACE shall use version control systems to manage any code changes and keep track of progressive developments.

[SRS 3.3.4.2] TRACE shall be designed with modular components that allow for easy replacement or updating of individual modules.

[SRS 3.3.4.3] All third-party dependencies shall be documented, including version, source, and any modifications made. This documentation shall also include guidelines for updating these dependencies and assessing the potential impacts of such updates on the system.

[SRS 3.3.4.3] TRACE code’s repository shall contain a markdown file that justifies the 32 CAT 1 STIG vulnerabilities.

[SRS 3.3.4.4] TRACE markdown file shall include a table that addresses the vulnerability ID, description, and mitigating reason for each vulnerability as shown in Table 3.4.3-1.

# **Portability**

[SRS 3.3.5.1] TRACE shall ensure that no more than 10% of its components contain host-dependent code to facilitate easier portability and reduce platform-specific dependencies.

[SRS 3.3.5.2] TRACE’s total codebase shall contain no more than 5% of host-dependent code.   
This minimizes reliance on any single host's operating system or hardware specifications, enhancing system portability across different environments.

[SRS 3.3.5.3] TRACE shall standardize the use of cross-platform third-party libraries, such as OpenSSL for cryptography.

# **Design and Implementation Constraints**

[SRS 3.3.6.1]The system shall be developed with a mix of Python 3.11.4 or above for the backend and Sveltekit 2.5.21 for the frontend (JavaScript framework).

[SRS 3.3.6.2]The system shall function on the Kali Linux operating system due to the tools that are pre-packaged within Kali Linux complementing the services the system provides such as brute force, database enumeration etc.

[SRS 3.3.6.3] The machines shall have no component that outperforms other machines by a margin of 5 percent to ensure all machines will have no machine-specific issues with the application, results, or storage of results.

[SRS 3.3.6.4] The system shall have the minimum specifications requirements of have at least 8 gigabyte RAM, a 4-Core 2.5Ghz processor, and 10 gigabytes of available storage space.

[SRS 3.3.6.5] If the machine is not Linux based, the machine shall have a Kali Linux virtual machine installed by the application “Virtual Box” from Oracle.

[SRS 3.3.6.6] The system shall only consist of open-source third party services that are independent of updates or esoteric knowledge to continue to maintain the system (No APIs).

[SRS 3.3.6.7] The system shall use a structured database to store all the general data corresponding to the system itself.

[SRS 3.3.6.8] The system shall only run one tool at a time but allow users to set up and queue tools while one is running.

[SRS 3.3.6.9] The system shall have a log that has the max length of 250 entries. If it exceeds, remove the first entry and make the second entry the first.

[SRS 3.3.6.10] The system shall have results exported and transferable as an XML or CSV file.

[SRS 3.3.6.11]The system shall have the lead analyst create, delete, modify, and lock projects within the system. All analysts can open/work on, import, and export projects if they are unlocked.

[SRS 3.3.6.12]The system shall adhere to the guidelines provided by the Security Technical Implementation Guide from the National Institute of Standards and Technology (NIST) 800-53 document. More specifically, it is imperative for the system to be void of CAT I vulnerabilities.

# **Other Requirements**

# **Data Flow**

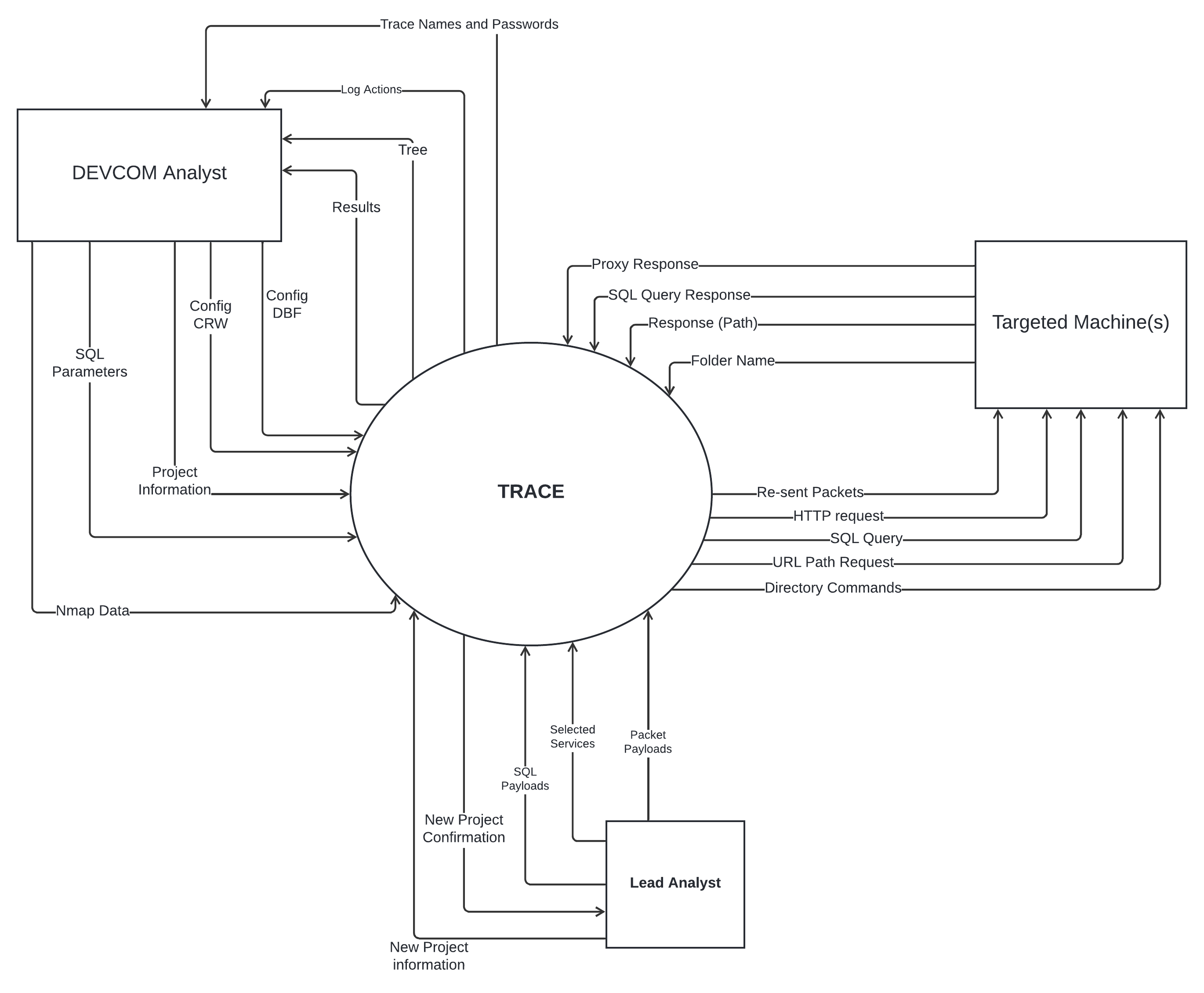
The system receives information (data inputs) from external entities and transforms them into data outputs consumed by external entities. Figure 14 shows the Context Diagram (also known as Data Flow Diagram (DFD) L0) of the TRACE with its data inputs and outputs. A DFD L1, see Figure 3.4.1-1, shows internal process transformations that convert data inputs from either external entity, other process transformation(s), or data storage(s) into data outputs that feed other process transformation(s), data storage(s) or external entities. Information about data terms can be seen in the data dictionary on Section 3.4.2.

# **Context Diagram (AKA DFD Level 0)**

Table 3.4.1-1 displays the legend for the context diagram.

**Table 3.4.1-1: Context Diagram Legend**

|  |  |  |
| --- | --- | --- |
| NAME | NOTATION | DESCRIPTION |
| System | A white circle with black text  Description automatically generated | The system being modeled. Represented as a single process that takes input data and transforms it into output data. |
| Terminator | A white rectangular sign with black text  Description automatically generated | External entity that interacts with the system by sending or receiving data. Also known as source or sink. |
| Data Flow | A black text with a arrow pointing to the right  Description automatically generated | Represents data or information that flows between the system and a terminator or vice versa. The name of a data flow may be followed by an apostrophe – i.e. **<data\_name>’**. This signifies that the data is altered in some way. |

Figure 3.4.1-1. TRACE Context Diagram

# **Data flow diagrams**

Table 3.4.1.2 displays the legend for the data flow diagrams.

**Table 3.4.1-2: Data Flow Diagram Legend**

|  |  |  |
| --- | --- | --- |
| NAME | NOTATION | DESCRIPTION |
| Process | A white circle with black text  Description automatically generated | Takes input data and transforms it into resulting output data. Also known as Data Transformation. It has a process number useful to keep track of the diagram level. |
| Sink | A close-up of a sign  Description automatically generated | Consumes information produced by the system. Also known as a Terminator or External Entity. |
| Source | A close-up of a sign  Description automatically generated | Produces information used by the system. Also known as a Terminator or External Entity. |
| Data Store |  | Stores data persistently for later use. |
| Data Flow | A close-up of a data flow  Description automatically generated | Represents data or information that flows between process-process, process-data store, process-sink, process-source. The name of a data flow may be followed by an apostrophe – i.e. **<data\_name>’** when the data altered in some way. |

A diagram of a company

Description automatically generated

Figure 3.4.1-2. TRACE Data Flow Diagram

[SRS 3.4.1.2.1] The “Manage Project” process shall receive data inputs *Project Information, Nmap data,* and, *Project Configuration, New Project information*, and produce outputs *Project Confirmation’, New Project, Nmap data* andby the following steps:

1. Gather project information
2. Gather Nmap data
3. Create new project and store project information
4. Store Nmap data
5. Modify the project configuration
6. Save project information

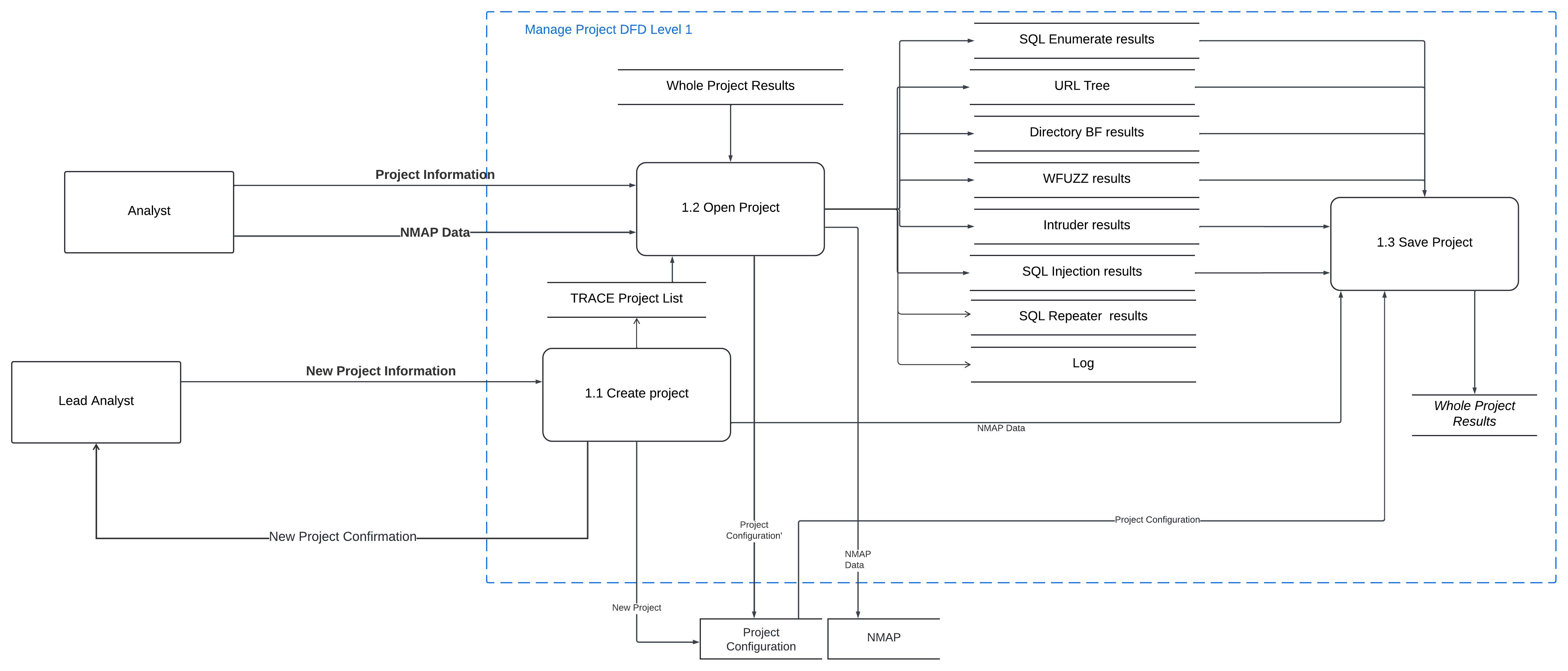


Figure 3.4.1-3. Manage Project DFD Level 2

[SRS 3.4.1.2.1.1] The “Create Project” process shall receive data input *New Project Information* and produce outputs *New Project Confirmation, Nmap data, New Project* and shall add the project to the TRACE project List by the following steps:

1. Gather project information
2. Decompose project information into Nmap data, project configuration, and project name
3. Create and store new project information
4. Log the system actions

[SRS 3.4.1.2.1.2] The “Open Project” process shall receive data inputs *Project Information, Nmap Data, Whole Project Results, TRACE Project List* and produce the outputs *SQL Enumerate results, URL Tree, Directory BF Results, WFUZZ Results, Intruder Results, SQL Injection Results, SQL Repeater Results, Project Configuration’, Nmap Data,* and a *Open Project Log* by the following steps:

1. Gather the project information
2. Gather the project results.
3. Gather Nmap data
4. Gather the project list
5. Open project
6. Modify Project Configuration
7. Log the system actions

[SRS 3.4.1.2.1.3] The “Save Project” process shall receive data inputs *SQL Enumerate results, URL Tree, Directory BF results, WFUZZ results, intruder results, SQL Injection results, SQL repeater results,* and the *log* of the system and produce outputs *Whole Project Results,* and *Save Project Command* by the following steps:

1. Generate a log for saving the project
2. Gather the project information
3. Save the project

[SRS 3.4.1.2.2] The “HTTP Client and Proxy” process shall receive data inputs *URL, Nmap Data, Request (Name and Password), SQL Command, Selected Packets, Modified Packets,* and *Proxy Response* and produce the outputs *Re-sent Packets, HTTP Request, WFuzz Response, Resend Packets Response, SQL Results, Modify Packets Results,* and *Raw Packets* by the following steps:

1. Gather the project information
2. Establish a Proxy Server
3. Send packets
4. Send requests
5. Receive responses
6. Save received responses
7. Create a log of the system actions

[SRS 3.4.1.2.3] The “Show Log Actions” process shall take the inputs from *User Actions Log* and produce the output *Log Actions* by the following steps:

1. Gather the log data
2. Display the log data to the analyst

[SRS 3.4.1.2.4] The “Show Tree” process shall take the input *Tree* and produce the outputs *Tree,* and *Show Tree Command* by the following steps:

1. Gather the tree information
2. Generate a log command to show the tree
3. Display the tree to the analyst

[SRS 3.4.1.2.5] The “WFuzz” process shall take the inputs *WFuzz Response, and Names and Passwords* and produce the outputs *WFuzz Command,* and *Request (Name and Password)* by the following steps:

1. Gather the Names and Passwords
2. Send the Request
3. Gather the response
4. Log the WFuzz Command

[SRS 3.4.1.2.6] The “Show TRACE Names and Passwords” process shall take the inputs *TRACE Names and Passwords* and produce outputs *Show TRACE names and Passwords Command,* and *Results* by the following steps:

1. Gather the generated names and passwords
2. Generate a log for the show TRACE names and passwords command
3. Generate Results

[SRS 3.4.1.2.7] The “Modify Packets (Intruder)" process shall take the inputs *Modify Packets Results, Packets, Payload for Insertion Points* and produce the outputs *Modified Packets, Modify Packets Command,* and *Modify Packets Actions* by the following steps:

1. Gather packet information
2. Modify packets as specified by the analyst
3. Send modified packets
4. Send log of actions
5. Send log of modify packets command

[SRS 3.4.1.2.8] The “Crawler" process shall take the inputs *Response (Path),* and *Config CRW* and produce outputs *URL Path Request, URL Path,* and *Crawler Actions* by the following steps:

1. Gather configuration information
2. Send the URL Path request
3. Gather the response
4. Store the URL Path
5. Generate and store a log for crawler actions

[SRS 3.4.1.2.9] The process “AI Create Usernames and Passwords” process shall take the inputs *WFuzz Response, Repeater Response, SQL Results, Modify Packets Results, Selected Services, Enumerate Results,* and *Tree* and produce the outputs *TRACE names and Passwords,* and *AI Actions* by the following steps:

1. Gather all results
2. Generate Usernames and Passwords using the AI component
3. Save the generated Usernames and Passwords
4. Log the AI actions
5. Send the generated Usernames and Passwords to the DEVCOM analyst

[SRS 3.4.1.2.10] The “SQL Injection” process shall take the input *SQL Payloads* and produce the outputs *SQL Command,* and *SQL Actions* by the following steps:

1. Gather SQL injection information
2. Generate and send the SQL Command
3. Log the SQL Actions

[SRS 3.4.1.2.11] The “Directory Brute Force” process shall take the inputs *Config DBF, Folder Name,* and *Directory List* and produce the outputs *Directory Commands, Directory Path,* and *Brute Force Actions* by the following steps:

1. Gather Brute force information
2. Perform directory brute force
3. Generate the Directory Path
4. Generate the directory commands
5. Log the Brute force actions

[SRS 3.4.1.2.12] The “Enumerate DB” process shall take the inputs *SQL Parameters*, *SQL Query Response* and produce the outputs *Enumerate Results, Enumerate Actions,* and *SQL Query* by the following steps:

1. Gather SQL information
2. Send SQL Query
3. Receive the SQL Query response
4. Generate the results
5. Log the Enumerate actions

[SRS 3.4.1.2.13] The “Resend Packets (Repeater)” process shall take the inputs *Selected Packets* and *Modified Packets* and produce the outputs *Resend Packets Command, Modified Packets,* and *Selected Packets* by the following steps:

1. Gather packet information
2. Modify packets
3. Send selected packets
4. Log the command to resend packets

# **Data Dictionary**

|  |
| --- |
| Table  Description automatically generated |

Figure 3.4.1.3-1. Data Dictionary Legend

**Table 3.4.1.3-1: Data Dictionary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Composite /  Atomic Term Name | Composite formula / Atomic Value | Description | Type | Default Value |
| Project Information | = ProjectName + AIN | The information necessary to create a project. | String, String | Null |
| ProjectName | = {Ch}125 | The name provided by the analyst to identify the project. | String | Null |
| AIN | = Ch + [Ch] + Ch | The initials of the analyst. Needed for any analysts that join a project. | String | Null |
| Nmap Data | = {IP + Port# + ServiceList}n | The network map data including the list of services possible. | String, Int, List | Null |
| IP | = {Ch}12 | The Internet Protocol that is being utilized for the project. | String | 000.000.000.000 |
| Port# | = Int | The port number of the IP that is being utilized. | Int | Null |
| ServiceList | = {Service}7 | A list consisting of all of the services that can be executed. | List | Null |
| Service | = {Ch}32 | The name of a service that will be utilized within TRACE. | String | Null |
| New Project Information | = ProjectName + AIN + IsLead + DateRange | Information for creating a new project. | String, String, Boolean, Date, Date | Null |
| IsLead | = Boolean | A Boolean value for whether the analyst is a lead or not. | Boolean | False |
| DateRange | = StartDate + EndDate | The range of dates the project is expected to remain active. | Date, Date | Null |
| StartDate | = Date | The starting date of the project. | Date | Null |
| EndDate | = Date | The expected end date of the project. | Date | Null |
| New Project Confirmation | = Boolean + {String} | A message that informs of the status of a project’s creation. | Boolean, String | False, “*Reason for failure*” |
| New Project | = ProjectName + AIN + IsLead +{AIN}N + DateRange | Information for a new project. | String, String, Boolean, Date, Date | Null |
| Project Configuration | = ProjectName + {AIN}N + IsLead + DateRange | Configuration information for a project. | String, String, Boolean, Date, Date | Null |
| Project Configuration’ | = ProjectName + {AIN}N + IsLead + DateRange | Modified configuration information. | String, String, Boolean, Date, Date | Null |
| Selected Services | = {Service}7 | A list consisting of up to all of the services that have been selected for execution. | List | Null |
| URL | = {Ch}255 | A URL following standard formatting. | String | Null |
| URL Path Request | = String | A path request made to the targeted machine. | String | Null |
| URL Path | = String | The paths from the targeted machine’s response. | String | Null |
| Re-sent Packets | = {Packet}N | Packets that have been re-sent through the HTTP Client to the targeted machine. | List | Null |
| Selected Packets | = {Packet}N | Specific packets that have been selected to be sent through the Repeater. | List | Null |
| HTTP Request | = String | An HTTP request sent through the HTTP Client to the targeted machine. | String | Null |
| Proxy Response | = {String}N | The proxy response from the targeted machine back to the system. (Structure of Strings will depend on the service). | List | Null |
| Packet | = {String}N | The analyst chosen HTTP request/response to be used within the service. (Structure of Strings will depend on the service). | List | Null |
| Raw Packets | = {Packet}N | The unmodified packets to be modified by the Intruder Service. | List | Null |
| Modified Packets | = {Packet}N | Packets modified based on the needs for their usage. | List | Null |
| Intruder Packets | = {Packet}N | Packets to be sent to the Intruder Service. | List | Null |
| Payloads for Insertion Points | = {String}N | Payloads that are chosen by the analyst to be utilized within the Intruder Service. | List | Null |
| Request (Name and Password) | = String | The request includes the names and passwords sent by the WFuzz Service (multiple parameters would be separated by a delimiter). | String | Null |
| Names and Passwords | = {String + String}N | A list of words to be used with the WFuzz Service and/or the Directory Brute Force Service. | List (String, String) | Null |
| WFuzz Response | = {String}N | The System response for the WFuzz Service (A list of responses from the server). | List | Null |
| WFuzz Command | = String | The user-input command for running the WFuzz Service. | String | Null |
| Repeater Response | = {String}N | The System response for the Repeater Service (formatted as an HTTP response). | String | Null |
| Repeater Command | = {String}N | The user-input command for running the Repeater Service. | String | Null |
| SQL Results | = String | The System results for the SQL Injection Service (Confirmation or rejection regarding exploitability). | String | Null |
| SQL Command | = String | The user-input command for running the SQL Injection Service. | String | Null |
| SQL Query | = String | The user-input query to be run by the system. | String | Null |
| SQL Query Response | = String | The System response for the SQL Query Service. | String | Null |
| SQL Payloads | = {String}N | Payloads that are chosen by the analyst to be utilized within the SQL Injection Service. | List | Null |
| SQL Parameters | = IP + Port# + SQL Query | Parameters to be used within the Database Enumeration Service. | String | Null |
| SQL Actions | = {String}N | A logged note of the actions done by the SQL Injection Service. | String | Null |
| Intruder Results | = {String}N | The System results for the Intruder Service. | List | Null |
| Intruder Command | = {String}N | The user-input command for running the Intruder Service. | String | Null |
| Intruder Actions | = {String}N | A logged note of the actions done by the Intruder Service. | String | Null |
| Enumerate Results | = {String}N | The System results for the Database Enumeration Service (Consists of all available versions that can extracted). | String | Null |
| Enumerate Actions | = {String}N | A logged note of the actions done by the Database Enumeration Service. | String | Null |
| Config CRW | = CrawlSchedule + ExcludedURLs + Depth | Configuration settings to be provided by the analyst for the Crawler Service. | Time, Date, List, Int | Null |
| CrawlSchedule | = Time + Date | The time and date as specified by the analyst during which they want to perform the crawl. | Time, Date | Null |
| ExcludedURLs | = {URL}N | A list of URLs the Crawler Service is not allowed to touch. | List | Null |
| Depth | = Int | The depth the Crawler Service is allowed to go per crawl. | Int | 3 |
| Crawler Actions | = {String}N | A logged note of the actions done by the Crawler Service. | String | Null |
| Response (Path) | = {String}N | The System response for the Crawler Service. | String | Null |
| Config DBF | = DomainName + Directory List  + AttemptLimit FilterOptions | Configuration settings to be provided by the analyst for the Directory Brute Force Service. | String, List, Int | Null |
| Directory List | = {String}N | The list of directories as provided by the analyst. | List | Null |
| DomainName | = {Ch}255 | The name of the domain that is being targeted by the Directory Brute Force Service. | String | Null |
| AttemptLimit | = Int | The number of attempts the Directory Brute Force Service can make before moving on. (-1 indicates unlimited). | Int | -1 |
| FilterOptions | = {Error Code}^n | Directory Bruteforce display only specific error codes | List | Null |
| Error Code | = Integer | HTTP status code e.g. 404 | Int | -1 |
| Brute Force Actions | = {String}N | A logged note of the actions done by the Brute Force Service. | String | Null |
| Folder Name | = {Ch}125 | The name of the folder as returned by the Targeted Machine. | String | Null |
| Directory Path | = String | The path leading to the directory (using standard directory formatting). | String | Null |
| Directory Commands | = String | The user-input command for the directory. | String | Null |
| Log Actions | = {SQL Actions + Intruder Actions + Enumerate Actions + Crawler Actions + Brute Force Actions + AI Actions + Resend Packets Command + Modify Packets Command}n | A logged text file of the actions done by all internal parts of the system. | Text File | Null |
| AI Actions | = {String}N | A logged note of the actions done by the AI. | String | Null |
| Trace Names and Passwords | = {String}N | The AI-generated list of names and passwords. | Text File | Null |
| Show Trace Names and Passwords Command | = {String + String}N | The user-input command for showing the AI-generated names and passwords list. | List (String, String) | Null |
| Tree | = {String}N | The TRACE generated tree that shows all of the paths and sub-directories found through the Crawler and Directory Brute Force Services. | Directory Tree | Null |
| Show Tree Command | = String | The user-input command for showing the AI-generated tree. | String | Null |
| Results | = {String + String}N | The AI-generated list of names and passwords. | Text File | Null |

# **Operations**

[SRS 3.4.2.1] The TRACE system shall consolidate the following:

[SRS 3.4.2.3] The actions of various enterprise testing tools/techniques within one view.

[SRS 3.4.2.2] Information gathered from the tools/techniques within one organized, modern, and appealing GUI.

[SRS 3.4.2.4] Username and Password Generation: The TRACE system shall leverage the results of these tools/techniques with sentiment analysis and novel shallow learning search algorithms to generate customized username and password lists.

[SRS 3.4.2.5] The TRACE system shall have a UI interface for analysts to interact with via a dashboard.

[SRS 3.4.2.6] The TRACE system shall allow users to configure projects and default pathing to relevant executables and wordlists.

[SRS 3.4.2.7] The TRACE system shall output a tree graph representation of the system with each node containing a label of the IP:PORT combination, a confirmed URL, a screenshot of the technology present at that URL, and the option to expand additional windows.

# **CAT 1 STIG Vulnerabilities**

**Table 3.4.3-1: Justification of CAT 1 STIG Vulnerabilities**

|  |  |  |
| --- | --- | --- |
| **VULNERABILITY ID** | **DESCRIPTION** | **MITIGATING REASON** |
| V – 222400 | Validity periods must be verified on all application messages using WS-Security or SAML assertions. | This is a security measure to prevent the use of outdated or expired security tokens. |
| V – 222404 | The application must use both the NotBefore and NotOnOrAfter elements or OneTimeUse element when using the Conditions element in a SAML assertion. | For this project this would not apply. |
| V – 222612 | The application must not be vulnerable to overflow attacks. | Overflow attacks, such as buffer overflows, can lead to serious security vulnerabilities, allowing attackers to overwrite memory and execute arbitrary code.  TRACE mitigates overflow attacks by implementing:  Input Validation  Buffer Size Checks  Secure Coding Practices  Memory Safety Languages  Static Code Analysis  Dynamic Analysis |
| V – 222578 | The application must destroy the session ID value and/or cookie on logoff or browser close. | TRACE erases the session ID value or cookies on logoff to ensure privacy, security and system integrity. |
| V – 222430 | The application must execute without excessive account permissions. | TRACE is used locally in a browser like Mozilla Firefox and doesn't connect to the internet.  TRACE manages session IDs and cookies to protect sensitive information and user sessions.  TRACE implements the following:  Session Management  Cookie Attributes (‘HTTP Only’, ‘Same Site’)  Session Timeout  Clear Cookies on browser close  Secure local storage |
| V – 222432 | The application must enforce the limit of three consecutive invalid logon attempts by a user during a 15-minute time period. | TRACE blocks the analyst from attempting to login after three consecutive invalid login attempts to ensure there is no unauthorized access to the system. |
| V – 222577 | The application must not expose session IDs. | TRACE implements measures to prevent session ID exposure by implementing:  Protect against session fixation  Session ID rotation  Session timeout  Use secure cookies |
| V – 222609 | The application must not be subject to input handling vulnerabilities. | TRACE address input handling vulnerabilities by implementing:  Avoid trusting user input  Use parameterized queries  File upload security (valid file types and restrict size)  Error Handling  Input Validation |
| V – 222608 | The application must not be vulnerable to XML-oriented attacks | TRACE mitigates XML-oriented attacks by implementing:  Avoid XML External Entity Attacks  Use a secure XML Parser  Input validation for XML Data |
| V – 222602 | The application must protect from Cross-Site Scripting (XSS) vulnerabilities | TRACE mitigates Cross-Site Scripting (XSS) vulnerabilities by implementing:  Output encoding  Content Security Policy (CSP)  Use frameworks with built-in protection (e.g. React, Angular, Vue...)  Escape Dynamic JavaScript values |
| V – 222601 | The application must not store sensitive information in hidden fields | TRACE does not store sensitive information in hidden fields by implementing these practices:  Avoid Storing Sensitive Data  Secure Communication Channels  Use local storage  Data Masking |
| V – 222607 | The application must not be vulnerable to SQL Injection. | MongoDB is a NoSQL database and not susceptible to traditional SQL injection attacks.  TRACE prevents NoSQL injection by implementing:  Use parameterized queries  Sanitize input  Error Handling  Update dependencies |
| V – 222604 | The application must be protected from command injection. | TRACE mitigates command injection vulnerabilities, especially in the context of using MongoDB, by implementing:  Use parameterized queries  Sanitize input  Error Handling  Update dependencies |
| V – 222403 | The application must use the NotOnOrAfter condition when using the SubjectConfirmation element in a SAML assertion. | The analysts shall not be required to log in to multiple applications, so this does not apply. |
| V – 222585 | The application must fail to a secure state if system initialization fails, shutdown fails, or aborts fail. | TRACE fails to a secure state by implementing:  Error handling  Graceful initialization  Shutdown process  Abort handling  Testing Scenarios |
| V – 222550 | The application, when utilizing PKI-based authentication, must validate certificates by constructing a certification path to an accepted trust anchor. | TRACE shall validate certificates as necessary to ensure system security and trustworthiness. |
| V – 222522 | The application must uniquely identify and authenticate organizational users (or processes acting on behalf of organizational users). | TRACE uses unique user IDs and passwords |
| V – 222554 | The application must not display passwords/PINs as clear text. | TRACE uses and stores passwords. |
| V – 222596 | The application must protect the confidentiality and integrity of transmitted information. | TRACE uses secure channels. |
| V – 222399 | Messages protected with WS\_Security must use time stamps with creation and expiration times. | N/A  TRACE will not use WS\_Security tokens. |
| V – 222658 | All products must be supported by the vendor or the development team. | All products from TRACE will be supported by the development team and vendor |
| V – 222659 | The application must be decommissioned when maintenance or support is no longer available. | When decommissioned TRACE will be available to terminate any files and storage |
| V – 222551 | The application, when using PKI-based authentication, must enforce authorized access to the corresponding private key. | All PKI-Based Authentication shall give access to a corresponding private key. |
| V – 222620 | Application web servers must be on a separate network segment from the application and database servers if it is a tiered application operating in the DoD DMZ. | TRACE will have no use of the internet. |
| V – 222536 | The application must enforce a minimum 15-character password length. | TRACE makes use of a minimum 15-character password length. |
| V – 222643 | The application must have the capability to mark sensitive/classified output when required. | TRACE gives the option to mark sensitive data to specific users. |
| V – 222542 | The application must only store cryptographic representations of passwords. | TRACE store cryptographic representations of passwords. |
| V – 222543 | The application must transmit only cryptographically-protected passwords. | TRACE transmits cryptographically-protected passwords. |
| V – 222425 | The application must enforce approved authorizations for logical access to information and system resources in accordance with applicable access control policies. | TRACE gives the approved user specific data they are entrusted with. |
| V – 222642 | The application must not contain embedded authentication data. | TRACE requires analysts to give authentication details each time a user logs in. |
| V - 222662 | Default passwords must be changed. | TRACE requires analysts to change default passwords. |
| V - 222555 | The application must use mechanisms meeting the requirements of applicable federal laws, Executive Orders, directives, policies, regulations, standards, and guidance for authentication to a cryptographic module. | TRACE will follow all the guidelines of Federal Laws, Executive Orders, Policies, Regulations, Stands for the cryptographic module. |

Ω