System Design Document

For

Option 9

Team member: Michael, Zach, Troy, Dana

Version/Author	Date
Version 1.0; Everyone	9/29/2023
Version 2.0; Everyone	10/30/2023
Version 3.0; Everyone	11/17/2023
Version 3.1; Everyone	11/21/2023

TABLE OF CONTENT

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

1.1 Purpose and Scope

Team Vetronica (including Michael Hall, Dana Merry, Troy Neubauer, and Zachary Shepley) is developing a cybersecurity planning for executing vulnerability assessment for operational environments, application of cybersecurity in operational environments and conducting decision analysis of crucial infrastructure by implementing model-based systems engineering.

1.2 Project Executive Summary

This section provides a description of the project from a management perspective and an overview of the framework within which the conceptual system design was prepared. If appropriate, include the information discussed in the subsequent sections in the summary.

1.2.1 System Overview

This section describes the system in narrative form using non-technical terms. It should provide a high-level system architecture diagram showing a subsystem breakout of the system, if applicable. The high-level system architecture or subsystem diagrams should, if applicable, show interfaces to external systems. Supply a high-level context diagram for the system and subsystems, if applicable. Refer to the requirements trace ability matrix (RTM) in the Functional Requirements Document (FRD), to identify the allocation of the functional requirements into this design document.

1.2.2 Design Constraints

This section describes any constraints in the system design (reference any trade-off analyses conducted such as resource use versus productivity, or conflicts with other systems) and includes any assumptions made by the project team in developing the system design.

We plan to support performing analysis on models created in both Capella and MagicDraw. Downslecting to one modeling tool will greatly simplify both the import logic for our tool, as well as letting us make more assumptions about the data format plus being able to support features of one which are not in the other.

We Intend on supporting both, but as we only have limited information on the technical scope of integrating both modeling tools, this is an ongoing decision which we will have to weigh as more information comes in.

1.2.3 Future Contingencies

This section describes any contingencies that might arise in the design of the system that may change the development direction. Possibilities include lack of interface agreements with outside agencies or unstable architectures at the time this document is produced.

Address any possible workarounds or alternative plans.

As mentioned above, we may have to select a single modeling tool based on a variety of factors such as: scope of integration, priority of other features, and timeline deadlines.

1.3 Document Organization

The design of this document is to give the reader an idea of the design of our system. The sections that follow provide information on how the product works, the interactions, limitations, security, and designs of both the hardware as well as the software.

1.4 Project References

This section provides a bibliography of key project references and deliverables that have been produced before this point.

Our github page can be found at:
https://github.com/Vetronica/CybAnalysisTool/tree/main

1.5 Glossary

CLI - Command Line Interface
GUI - Graphical User Interface
MBsE - Model Based systems Engineering
Rust - The Rust programming language: A language empowering everyone to build reliable and efficient software.

2 SYSTEM ARCHITECTURE

In this section, describe the system and/or subsystem(s) architecture for the project. References to external entities should be minimal, as they will be described in detail in Section 6, External Interfaces.

2.1 System Hardware Architecture

Any modern 64-bit x86 computer. Our software will be written with cross-platform compatibility in mind using a modern language with first-class support for OS abstractions like Rust. The OS must also be capable of running Java SE 8. The system that you are running must also have at least 4 GB of ram, and at least 16 GB available of hard disk space.

2.2 System Software Architecture

Our software will be distributed as a monolithic, statically-linked, binary to make distribution simple. Similar modules will be grouped together to support encapsulation and data hiding. The exact scope of the user experience (CLI vs GUI) is yet to be determined, however we will start with a CLI based approach which will allow us to re-use business logic in core modules if we decide to provide a GUI.

2.3 Internal Communications Architecture

N/A. Data will flow internally through function calls with some state stored using the actor pattern.

Note: The diagrams should map to the FRD context diagrams.

3 HUMAN-MACHINE INTERFACE

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Any additional information may be added to this section and may be organized according to whatever structure best presents the operator input and output designs. Depending on the particular nature of the project, it may be appropriate to repeat these sections at both the subsystem and design module levels. Additional information may be added to the subsections if the suggested lists are inadequate to describe the project inputs and outputs.

3.1 Inputs

This section is a description of the input media used by the operator for providing information to the system; show a mapping to the high-level data flows described in Section 1 .2.1, System Overview. For example, data entry screens, optical character readers, bar scanners, etc. If appropriate, the input record types, file structures, and database structures provided in Section 3, File and Database Design, may be referenced. Include data element definitions, or refer to the data dictionary.

Provide the layout of all input data screens or graphical user interfaces (GUTs) (for example, windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary.

This section should contain edit criteria for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also address data entry controls to prevent edit bypassing.

Discuss the miscellaneous messages associated with operator inputs, including the following:

- Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
- Description of any access restrictions or security considerations
- Each transaction name, code, and definition, if the system is a transaction-based processing system

The software will import a model file from MagicDraw or Capella via the filesystem. This model file will represent the network and hardware architecture of a system of interest. The model file to use will be indicated to the application by a command line argument, or as a standard File -> Open menu in the GUI.

Next, the user will enter the networks of interest, threats of concern, usage spheres, etc. This communication will be done via a config file for the CLI, or as an interactive menu in the GUI.

- Define Network(s) of interest
- Define Threats of concern
- Establish applicable Conceptual data infrastructure and data structure (storage and processing)
- Establish applicable Data sphere
- Establish usage spheres
- Define cybersecurity strategies for specific threats
- Identify Data Acquisition methods (sensors, streaming, security log files)
- Identify current sensor vantage and blind spots (visual schemas)

3.2 Outputs

This section describes the system output design relative to the user/operator; show a mapping to the high-level data flows described in Section 1.2.1. System outputs include reports, data display screens and GUIs, query results, etc. The output files are described in Section 3 and may be referenced in this section. The following should be provided, if appropriate:

- Identification of codes and names for reports and data display screens
- Description of report and screen contents (provide a graphic representation of each layout and define all data elements associated with the layout or reference the data dictionary)
- Description of the purpose of the output, including identification of the primary users
- Report distribution requirements, if any (include frequency for periodic reports)
- Description of any access restrictions or security considerations

Upon completion of analysis, the application will report results to the user. The exact mechanism for this is yet to be determined, however will likely include a textual report printed to the console if run as a CLI, or an interactive pane if run as a GUI.

4 DETAILED DESIGN

This section provides the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate COTS packages into a single system. Every detailed requirement should map back to the FRD, and the mapping should be presented in an update to the RTM and include the RTM as an appendix to this design document.

4.1 Hardware Detailed Design

A hardware component is the lowest level of design granularity in the system. Depending on the design requirements, there may be one or more components per system. This section should provide enough detailed information about individual component requirements to correctly build and/or procure all the hardware for the system (or integrate COTS items).

If there are many components or if the component documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each component and its functions, adequately. Industry-standard component specification practices should be followed. For COTS procurements, if a specific vendor has been identified, include appropriate item names. Include the following information in the detailed component designs (as applicable):

- Power input requirements for each component
- Signal impedances and logic states
- Connector specifications (serial/parallel, 11-pin, male/female, etc.)
- Memory and/or storage space requirements
- Processor requirements (speed and functionality)
- Graphical representation depicting the number of hardware items (for example, monitors, printers, servers, I/O devices), and the relative positioning of the components to each other
- Cable type(s) and length(s)
- User interfaces (buttons, toggle switches, etc.)
- Hard drive/floppy drive/CD-ROM requirements
- Monitor resolution

N/A: Hardware is provided by the user and we aim to maximize hardware compatibility with our software. The software is unlikely to have strict hardware requirements as we have no real time requirements. A user on an older system will simply have a slower experience, but will still be able to utilize all features.

4.2 Software Detailed Design

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

- A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc.; explain any algorithms used by the module in detail
- For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
- Data elements, record structures, and file structures associated with module input and output
- Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
- Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section
- Report layout

The software design depends on the features from Capella and MagicDraw we wish to implement/leverage, and therefore is to be determined.

4.3 Internal Communications Detailed Design

If the system includes more than one component there may be a requirement for internal communications to exchange information, provide commands, or support input/output functions. This section should provide enough detailed information about the communication requirements to correctly build and/or procure the communications components for the system. Include the following information in the detailed designs (as appropriate):

- The number of servers and clients to be included on each area network
- Specifications for bus timing requirements and bus control
- Format(s) for data being exchanged between components
- Graphical representation of the connectivity between components, showing the direction of data flow (if applicable), and approximate distances between components; information should provide enough detail to support the procurement of hardware to complete the installation at a given location
- LAN topology

As we will be distributing a statically linked binary, the only communications our application will have are:

- Input of model file from Capella or MagicDraw into application memory via the file system
- Input of networks of interest, attack vectors user cares about, etc. to inform conditional analysis options
- Internal data flow via functions and shared state to implement analysis business logic
- Output results of analysis (initially printed to console)

5 EXTERNAL INTERFACES

External systems are any systems that are not within the scope of the system under development, regardless whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

5.1 Interface Architecture

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams in Section 1.2.1. If appropriate, use subsections to address each interface being implemented.

N/A

5.2 Interface Detailed Design

For each system that provides information exchange with the system under development, there is a requirement for rules governing the interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

- The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined
- Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
- Format(s) for error reports exchanged between the systems; should address the disposition of error reports; for example, retained in a file, sent to a printer, flag/alarm sent to the operator, etc.
- Graphical representation of the connectivity between systems, showing the direction of data flow
- Query and response descriptions

If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.

N/A

6 SYSTEM INTEGRITY CONTROLS

Sensitive systems use information for which the loss, misuse, modification of, or

unauthorized access to that information could affect the conduct of State programs, or the privacy to which individuals are entitled.

Developers of sensitive State systems are required to develop specifications for the following minimum levels of control:

- Internal security to restrict access of critical data items to only those access types required by users
- Audit procedures to meet control, reporting, and retention period requirements for operational and management reports
- Application audit trails to dynamically audit retrieval access to designated critical data
- Standard Tables to be used or requested for validating data fields
- Verification processes for additions, deletions, or updates of critical data

Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed.

Unauthorized access to analysis results showing possible vulnerabilities in the system could be critical to a user if accessed by a threat actor. Specific protections against unauthorized access are best left to system administrators since this configuration is outside the scope of our analysis tool and an unsafe network will compromise confidential data despite any attempts within the analysis tool to keep models or analysis data private.