SPL3 Technical Documentation

Course code: SE801

Submitted by

Radowan Mahmud Redoy

BSSE 1002

Supervised By

Adwait Nadkarni

William & Mary

Williamsburg, VA

Internal Supervisor

Kishan Kumar Ganguly

Institute of Information Technology

University of Dhaka

Mentored By

Amit Seal Ami

William & Mary

Williamsburg, VA

Submission date: 22nd September, 2022Tec

Technical Documentation

MASC Tool: Mutation Analysis for evaluating Static Crypto-API misuse detectors Tool

This project document has been approved by the following person.

Prepared By

Radowan Mahmud Redoy

BSSE - 1002

Internal Supervisor

Kishan Kumar Ganguly

Institute of Information Technology

University of Dhaka

Letter of Transmittal

September 22, 2022

BSSE 4th Year Exam Committee

Institute of Information Technology

University of Dhaka

Subject: Submission of technical report of “MASC Tool”.

Sir,
With due respect, I would like to provide the technical report on Software Project Lab III's "MASC Tool." I've tried my best to deliver a reliable report. However, it might not be faultless. I thus sincerely hope that you will be so kind as to accept my report and comply as a result.

Sincerely yours,

Radowan Mahmud Redoy
BSSE 1002
Institute of Information Technology
University of Dhaka.

Acknowledgement

Firstly, I would like to thank the Almighty for helping me complete the report. I am grateful to the Institute of Information Technology, University of Dhaka and William & Mary university in Williamsburg, Virginia, for giving me such a tremendous opportunity to work on "**MASC Tool - Mutation Analysis for evaluating Static Crypto-API misuse detectors**". I would like to convey my tremendous gratitude to my supervisor, Dr. Adwait Nadkarni,William & Mary university in Williamsburg, mentor Amit Seal Ami, William & Mary university in Williamsburg, local Supervisor Kishan Kumar Ganguly, Institute of Information Technology, University of Dhaka, for providing me guidelines about how I can prepare this report. They helped me a lot by sharing their valuable knowledge with me and providing necessary instructions to complete my project. Lastly, I would like to thank my classmates. They have always been helpful and provided valuable insights from time to time.

Abstract

This document contains the software requirements and specifications, architectural design and test planning of "**MASC Tool - Mutation Analysis for evaluating Static Crypto-API misuse detectors Tool**". This system can be used to detect flaws in crypto misuse detectors. The system uses mutation based analysis techniques in order to mutate code and later analyze the reports of the crypto misuse detectors to detect flaws. This tool will enable a user to easily and efficiently interact with MASC and use MASC according to their need. Besides specified operators and mutants the user will also be able to define their own strategy for mutation. Users can also set configurations according to their own needs and see the mutations in an example code before mutating specific source codes. Currently MASC provides support for JAVA with three mutation scopes and aims to increase its number of operators and scopes. MASC Tool will give the opportunity to easily use the MASC core. Besides there will also be work on various components which help to enhance MASC core as well. Overall, the product will enable MASC to become a full fledged tool for detecting flaws in crypto misuse detectors.

Table Of Contents

1. Introduction **8**

2. Motivation **2**

3. User Story **2**

4. Quality Function Deployment **3**

4.1 Normal Requirements 3

4.2 Expected Requirements 3

4.3 Exciting Requirements 4

5. Scenario Based Modeling **4**

5.1 Definition of Use Case 4

5.2 Use Case Diagram **5**

Level: 1 5

Level: 1.1 6

Level: 1.1.1 7

Level: 1.1.1.3 8

Level: 1.1.2 9

Level: 1.2 10

Level: 1.2.2 11

Level: 1.2.3 12

Level: 1.3 13

Level: 1.4 13

Level: 1.5 14

6. DataBase Modeling **16**

6.1 ER Diagram 16

6.2 Schema 16

7. Class Based Diagram **19**

7.1 General Classification 19

7.2 Selection criteria 21

7.3 Class Cards 23

7.4 Class Modeling Diagram 27

8. Architectural Design **28**

8.1 Representing the system in context 28
8.2 Define Archetypes 29
8.3 Refine the Architecture into Components 30
8.4 Describe instantiations of the system 30
8.5 Mapping requirements to software architecture 31

9. Preliminary test plan 32
9.1 Summary of Items & Features To be Tested 32
9.2 Validation criteria and testing plan 33

10. Conclusion 36

References 37

1. Introduction

Modern software systems must use cryptography correctly in order to protect confidentiality & availability of data. But as the domain of softwares are growing, so are the challenges of ensuring their security. Software security is not equal to secure software. So, the use of cryptographic techniques, cryptographic APIs may provide security to our software but it does not imply that we have made a secure software. Their inappropriate use can result in severe security vulnerabilities which can go undetected and, worst of all, create a false impression of security.

With the aim of solving this issue a number of tools have been developed over the years in order to detect the misuse of cryptographic APIs so that they do not create any kind of threat. These tools are mainly used in development or in the quality assurance phase in order to detect any kind of cryptographic misuse.

The issue is that the stakeholders are using these Crypto-detectors so enthusiastically despite the lack of solid confirmation of their efficacy. There are a few manually curated benchmarks; however, given the vast array of varied cryptographic protocols and APIs, these benchmarks may well be unreliable, inaccurate, and difficult to keep up with. As a result, they cannot be a valid method of evaluating the performance of Crypto-detectors.Here comes the MASC framework[1].

Mutation Analysis for evaluating Static Crypto-API misuse detectors– the MASC framework. MASC provides The first reliable and evolving evaluation technique that attempts to solve this problem of validation methods of evaluating the performance of the Crypto-detectors. MASC is a data-driven framework that uses mutation based analysis testing techniques to discover design flaws of Crypto-detectors. MASC mainly mutates Android/Java code by injecting codes containing crypto misuses. The mutants that the Crypto-detectors failed to detect can be examined to identify any design flaws in the Crypto-detectors.

2. Motivation

MASC is currently evolving day by day, in order to completely gain the benefits of the MASC framework there are a certain number of areas that need to be addressed. It needs further work to develop into an independent, polished piece of software. Overall it provides a wide range of research and development areas. Therefore I have chosen

to work with MASC in order to further enrich my knowledge and help MASC reach its ultimate goal.

3. User Story

A user may want to easily explore the operators provided by MASC and see the effects of these operators on their code. Operators are functions that produce mutants to indicate a specific kind of API abuse with which given codes will be mutated.

After their exploration the user will upload the source code in order to mutate them. Here it is necessary to maintain the status of different phases of this mutation process. The user will also require a profile management system in order to keep track of the processes that are running under the users ownership.

There are certain cases when it is necessary to create profiling for tools. We need to run various operators based on a predefined ranking of operators and generate a proper profile for different tools so that a user can easily know about that crypto misuse detection tools, inspect their strong points and which misuses they failed to detect and select the one which is necessary for them.

All operators are not necessarily needed or supported for all tools. Some tools have strong points or are designed for specific cases. Therefore a general profiling model for tools may not give the proper knowledge about that tool. Hence it is expected that the ranking criteria shall be or ranking of the operators that will be used for mutation, shall be configurable for various tools. The user shall be able select the ranking of the operators a side from the general ranking for mutating their codes and prepare the tool profiling upon that result.

Currently existing operators of MASC can cover 69 of the 105 total misuse cases identified. So there are discovered cases which need to be addressed. So, it is expected to further explore such areas in order to research and propose new operators. Overall, the main objective is to make MASC accessible to end users and further development of the framework.

4. Quality Function Deployment

In order to solve the problems which have been identified in the previous section I have identified some functionalities which have to be implemented. These functionalities include

4.1 Normal Requirements

The normal requirements for the project includes

- ### Web APP

  Developing a full fledged web app so that end users can easily use MASC. The web app will provide an interface to the user in the forms of MASC Lab and MASC Engine. MASC lab will provide a testing environment to the user in order to test different operators and inspect the mutated codes. MASC Engine will give users the privilege to upload source code and mutate them. Besides the front end will carry features like user profile management, history of the running processes etc.

- ### Profile and Data Management

  It is mandatory to ensure a full fledged database management system in order to keep track of the tools and user profiles. It will also be necessary to ensure the integrity of the database. It is necessary to maintain the status of the processes which are running and at the same can be linked to their owner.

- ### Wrapper for CryptoGuard

  CryptoGuard[2]: High Precision Detection of Cryptographic Vulnerabilities in Massive-sized Java Projects, but it does not provide its output report in the standard SARIF format. For this, a converter wrapper is required that will convert its output into the standard SARIF format

4.2 Expected Requirements

The expected requirements for the project includes

- ### Profiling for tools

  Different tools will be tested with different mutations in order to build a proper profile so that based on the generated profile they can update their failings if any and at the same time the users can also learn more about the tools and select which one fits them the most.

- ### Configurable Operator Ranking module

  The user will have the ability to configure the series of operators for tool profiling. It is possible that a tool is designed for specific cases which shall not support all kinds of mutations. Hence such cases are not necessarily needed for this tool. So the user will have the ability to configure a rank with the schemas and the profiling of the tools shall be run upon that ranked schema.

- Resource Management

  Manage all property files submitted by the user that is edit, update and delete any files. These files will contain all necessary properties which will be needed to run MASC

- Logging Mechanism

  Introduce a proper logging mechanism in MASC core in order to keep track of the running processes and their respective outputs..

4.3 Exciting Requirements

There are a lot of areas that still need to be explored which are included in the scopes of MASC. So it is possible and expected that these areas are explored and if possible new operators and schemas, mutation scopes are proposed for the MASC framework.

Besides, it is needed to work on some necessary fixes and updates in the overall MASC framework and overall work to improve the MASC framework.

5. Scenario Based Modeling

Scenario based modeling is an inexpensive rapid prototyping technique. This method is effective when systems are being built with the requirements vaguely known at the outset. Users are involved right from the start, to build prototypes evolving towards the final product.

5.1 Definition of Use Case

A use case is a plan that captures how the system will behave in different scenarios in response to a request from one of its stakeholders. A use case essentially offers a stylized tale of how a system user interacts with it in a particular set of circumstances.

Primary Actor

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

Secondary Actor

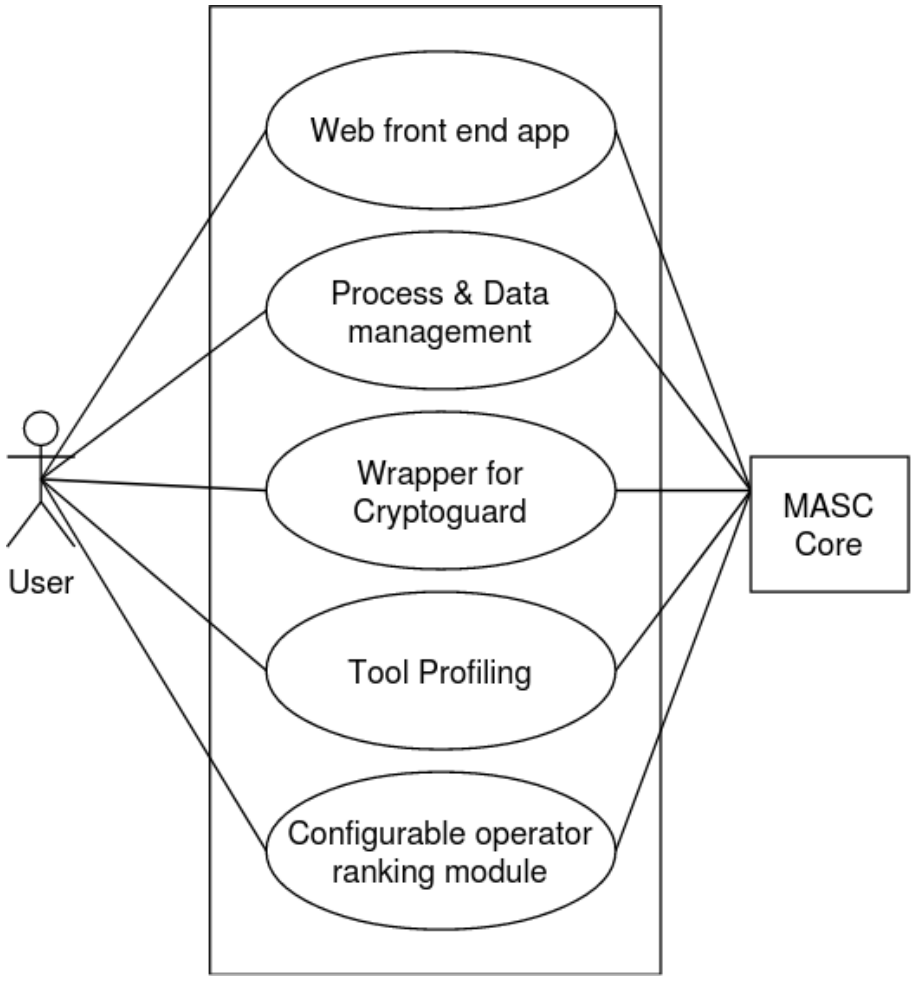
Secondary actors support the system so that primary actors can do their work. They either produce or consume information

5.2 Use Case Diagram

Use case diagrams for MASC Tool

Level: 1

Use Case Name: Masc Subsystems
Use case id: 1



Primary Actors: User
Secondary Actors: MASC Core
MASC shall include five sub systems in order to improve make MASC a completely usable tool

1.1 Web front end app: There shall be a web based front end application which shall enable the user to use different features and facilities of MASC.
1.2 Process & Data management: The user needs to configure various settings and upload different resources and it is necessary that this records and resources are stored properly.

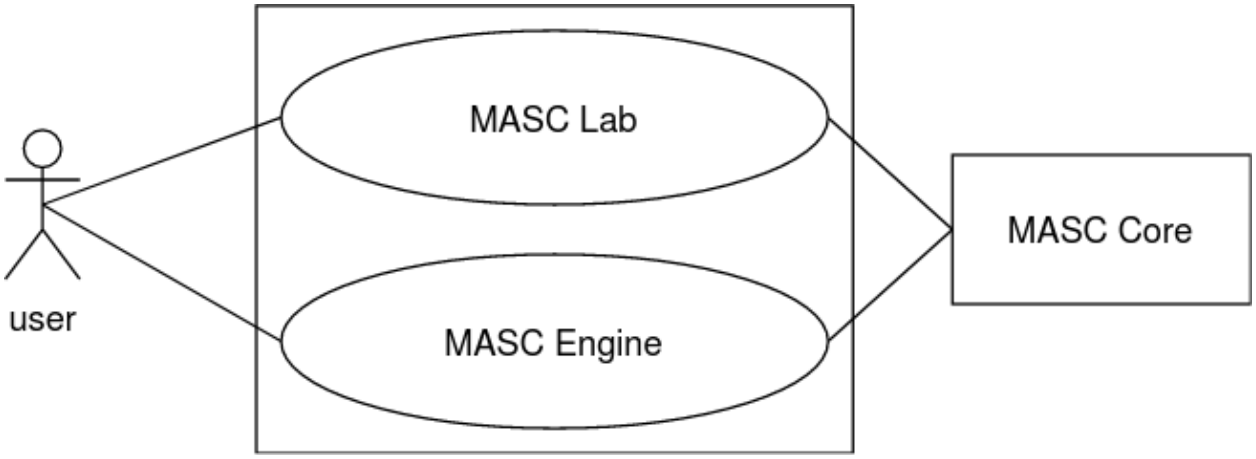
1.3 Wrapper for Cryptoguard: Cryptoguard[2] does not provide its outputs in standard SARIF format. So, it is essential to convert this into SARIF format for further utilization.

1.4 Tool Profiling: Users may run different tools in order to detect flaws in those tools. A tool profile shall be generated upon running it with the mutated code prep[ared by MASC.

1.5 Configurable operator ranking module: The user shall be able to configure the operator ranking while running mutations.

Level: 1.1

Use Case Name: Web front end
Use case id: 1.1



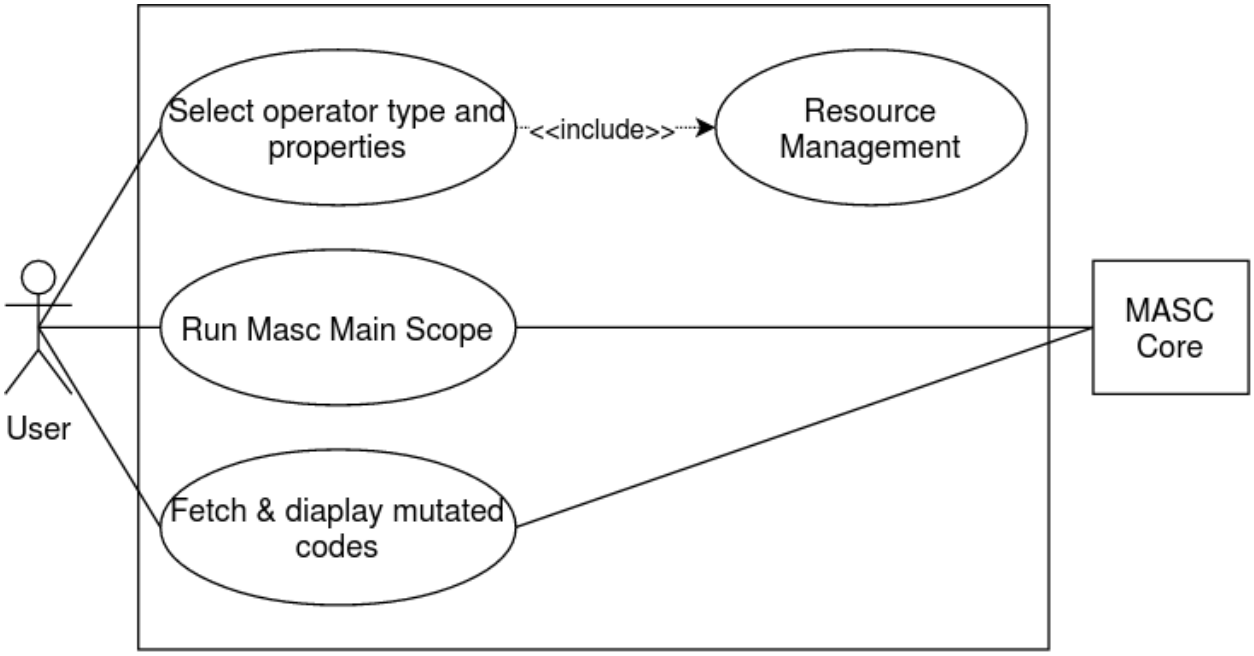
Primary Actor: User
Secondary Actor: MASC Core
Users shall be able to to mutate there codes in two ways.

1.1.1 MASC Lab: User gives property file as input and inspects the mutated codes for the given configurations in the property file.

1.1.2 MASC Engine: User upload source code zip for mutation and receives the mutated codes

Level: 1.1.1

Use Case Name: MASC Lab
Use case id: 1.1.1



Primary Actor: User
Secondary Actor: MASC Core

1.1.1.1 Select Operator type and properties: User selects operator type and previously uploaded property files or can upload new property files. This includes the resource management sub module.

Action: User selects properties file as input
Reply: System parses the selected file and collects value

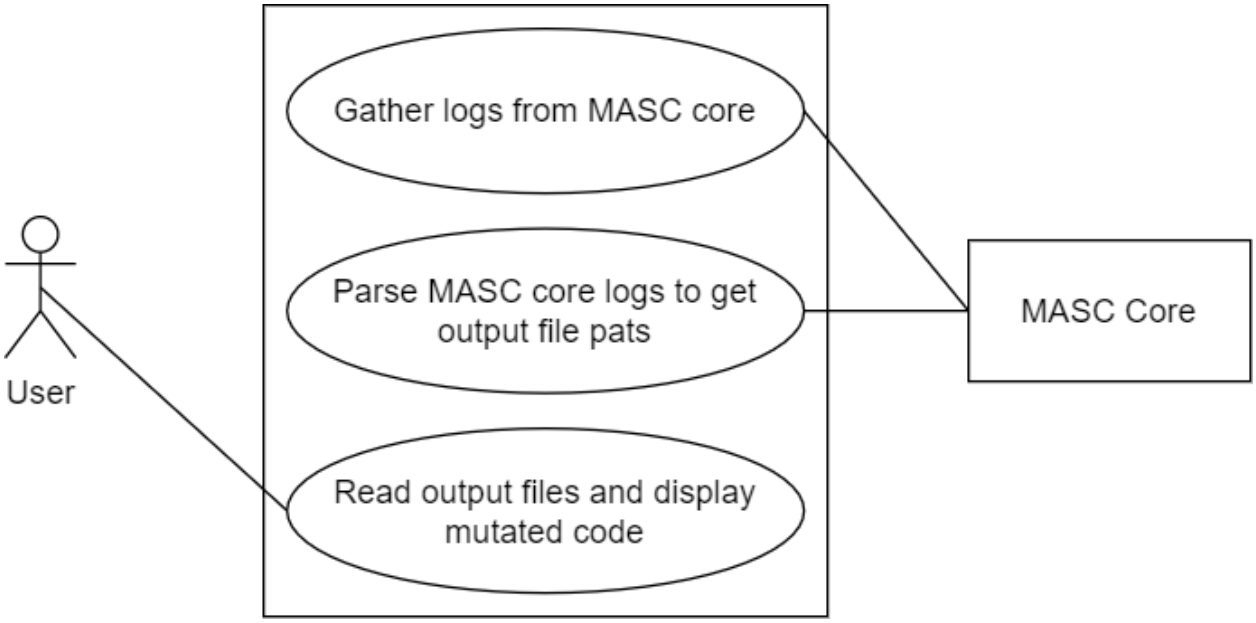
1.1.1.2 Run MASC Main Scope: User upon selection starts running the MASC process and mutated code for submitted configuration is generated.

Action: User initiates run command
Reply: MASC core is ran with given configuration

1.1.1.3 Fetch & display mutated codes: The codes generated by MASC main scope are parsed and displayed to the user.

Level: 1.1.1.3

Use case Name: Fetch & display mutated codes
Use case id: 1.1.1.3



Primary Actor: user
Secondary Actor: MASC Core

1.1.1.3.1 Gather logs from MASC Core: Loges are collected while running the MASC Core main scope for mutation.

Action: Run MASC core process
Reply: Get logs from MASC core

1.1.1.3.2 Parse MASC core logs to get output file paths: The collected logs are parsed in order to get the output paths for the mutated codes.

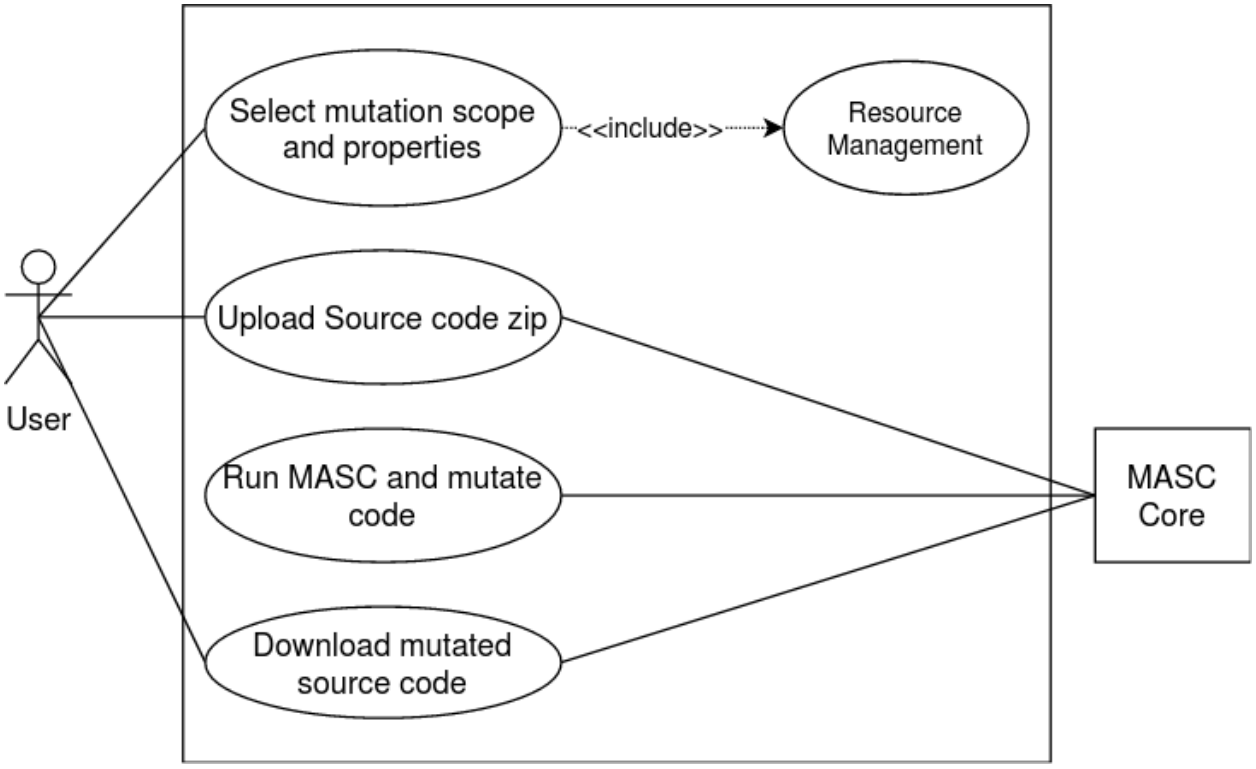
Action: Parse logs gathered from MASC
Reply: Output file paths of the mutated codes

1.1.1.3.3 Read output files and display mutated codes: The output files are then parsed the visual output is generated for the user to inspect.

Action: Parse files from the output paths
Reply: Combine output from the files and visualie to the user

Level: 1.1.2

Use case name: Masc Engine
Use case id: 1.1.2



Primary Actor: User
Secondary Actor: MASC Core
MASC engine will take source code as input and muate it

1.1.2.1 Select mutation scope and properties: User will first select the desired mutation scope and the uploaded properties file from resource management.

Action: User selects mutation scope and properties file as input
Reply: System parses the selected file and collects value

1.1.2.2 Upload Source code zip: User will uploaded the source code zip file for mutation.

Action: User uploads source code as zip
Reply: Zip file is saved and prepared for mutatio

1.1.2.3 Run MASC and mutate code: MASC core will be run using the selected scope and status will be saved as running.

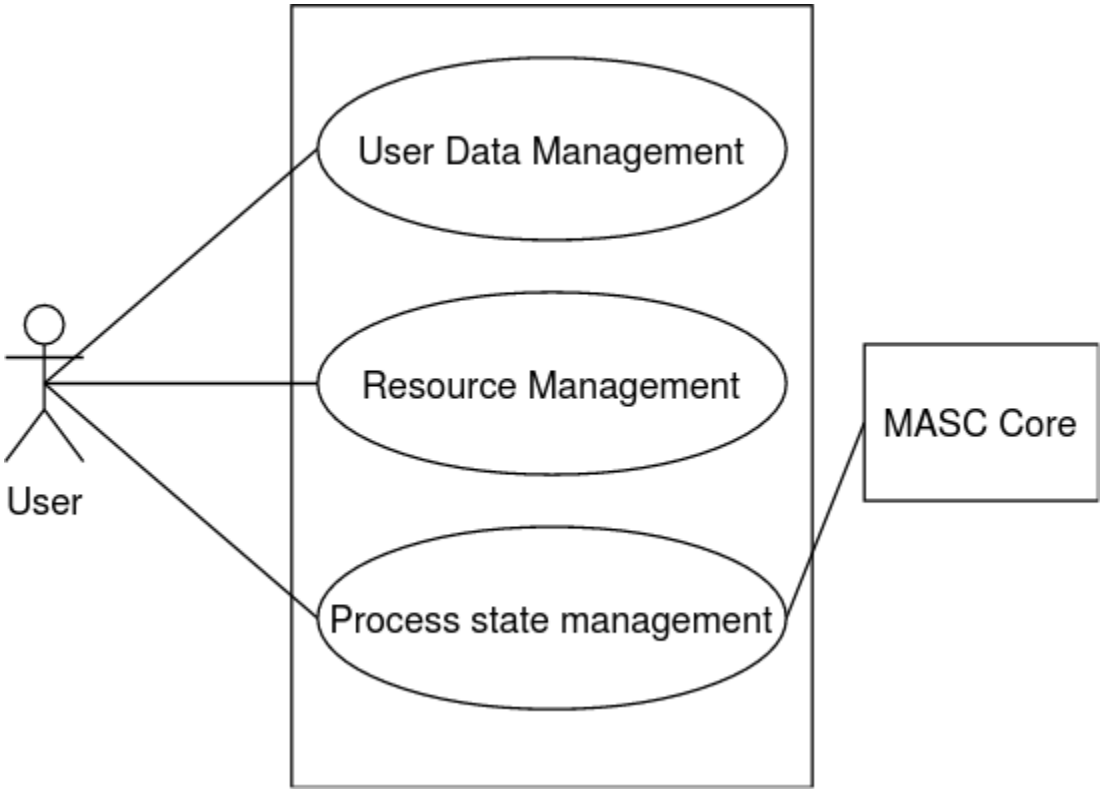
Action: User initiates run command
Reply: MASC core is ran with given configuration and selected scope

1.1.2.4 Download mutated source code: Once the process of mutating codes is finished , the user will be able to download the mutated codes.

Action: MASC core finishes mutation process
Reply: User can download mutated source code

Level: 1.2

Use case name: Property & data management
Use case id: 1.2

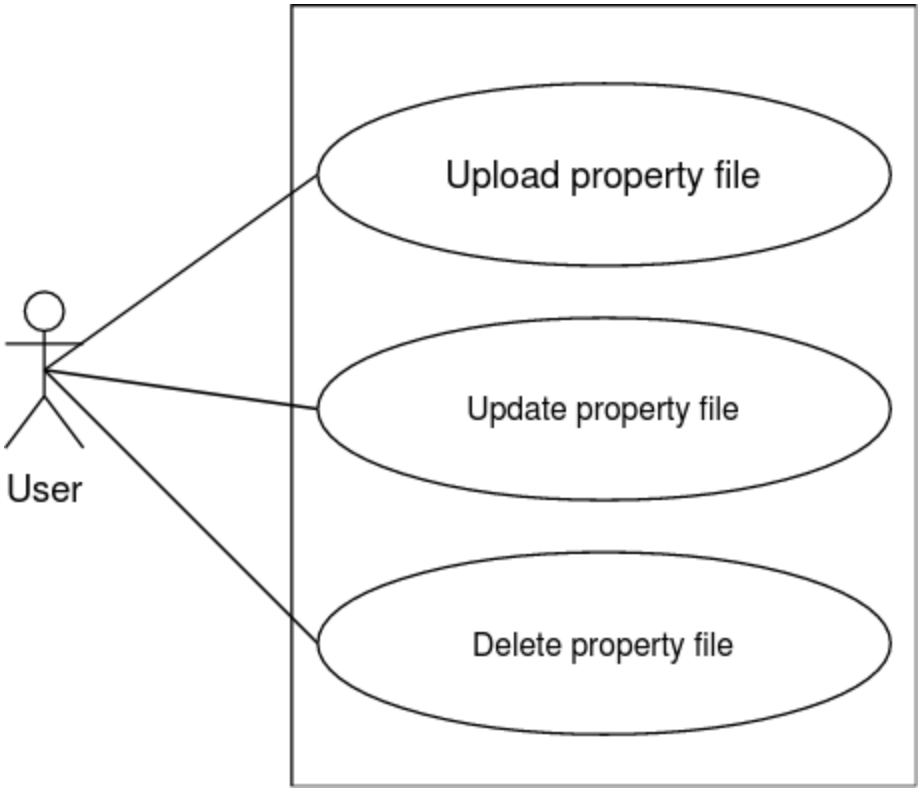


Primary Actor: User
Secondary Actor: MASC Core

1.2.1 User Data Management: Save the user data that is user profile, history of the processes run etc.
1.2.2 Resource Management: Store and manage property file and provide them when necessary.
1.2.3 Process State Management: Keep track of the running process, update their status when their state changes.

Level: 1.2.2

Use case name: Resource Management
Use case id: 1.2.2



Primary Actor: User
All property files and output files are stored, updated and deleted.

1.2.2.1 Upload property file: User uploads a property file and it is stored properly for further use.

Action: User uploads new property file
Reply: Save current file along with necessary datas

1.2.2.2 Update property file: User updates a property file and all changes are saved to the appropriate file.

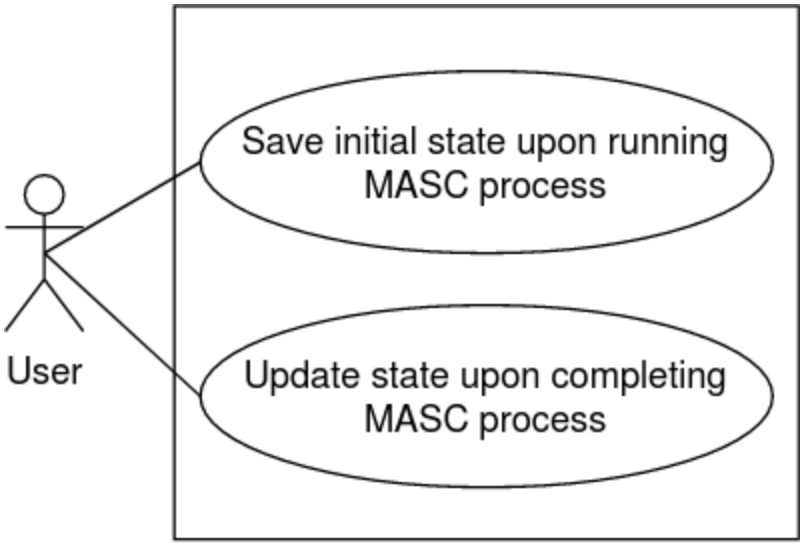
Action: User updates property file
Reply: Updates are saved to the property file

1.2.2.3 Delete property file: User can delete any of the property files. The file is removed along with its related datas.

Action: User deletes a property file
Reply: The file along with respective data is deleted

Level: 1.2.3

Use case name: Process state management
Use case id: 1.2.3



Primary Actor: User
All running processes are saved accordingly.

1.2.3.1 Save initial state upon running MASC: Save data of the current process once the user initiates a process

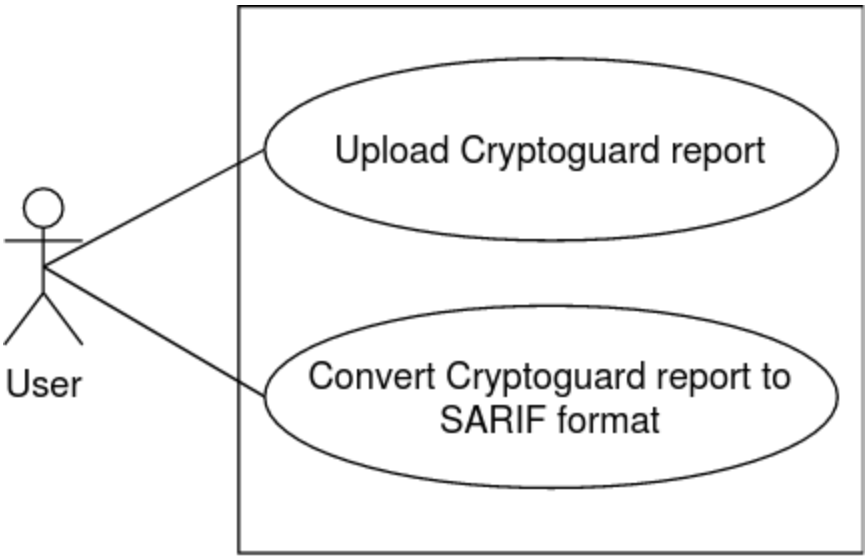
Action: User initiates a process run for MASC core
Reply: Initial information for the process run is saved

1.2.3.2 Update upon completing MASC process: Once the process is finished update its status and let the user know

Action: A running process changes state or finishes the process
Reply: The information for the process run is updated

Level: 1.3

Use case name: Wrapper for Cryptoguard
Use case id: 1.3



Primary actor: User
Wrapper to convert crypto guard report in standard SARIF format.

1.3.1 Upload Cryptoguard report: User uploads report of cryptoguard to the the system for further processing

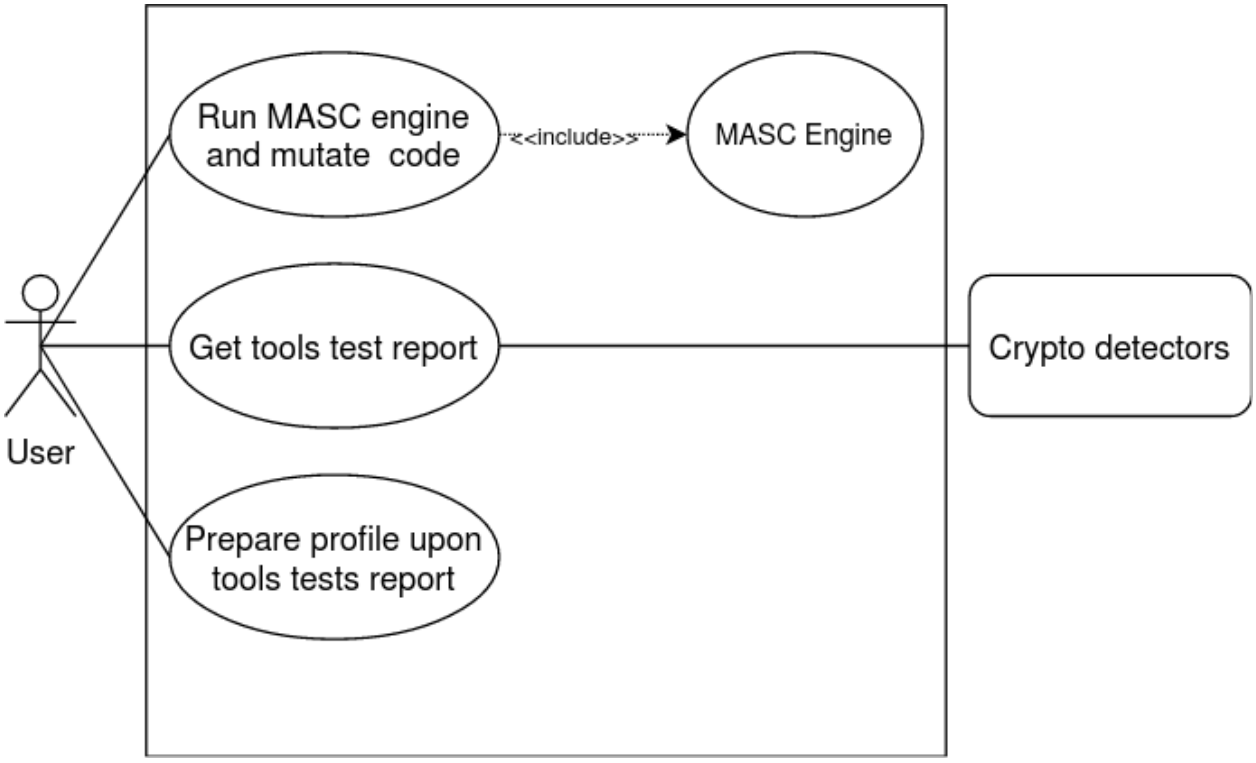
Action: User provides the generated report by Cryptoguard
Reply: The report is saved to the system for further processing

1.3.2 Convert Cryptoguard report to SARIF format: Convert the user uploaded report to SARIF format in order to properly parse it.

Action: User starts the conversion process
Reply: Report is converted to SARIF format

Level: 1.4

Use case name: Tool Profiling
Use case id: 1.4



Primary actor: User
Secondary actor: crypto detectors

1.4.1 Run Masc engine and mutate code: Upload source code in order to mutate it so that crypto detectors can be tested.

Action: User provides source code for mutation
Reply: Code is mutated for further use

1.4.2 Get tools test report: Run respected tools on the mutated code and get a test report.

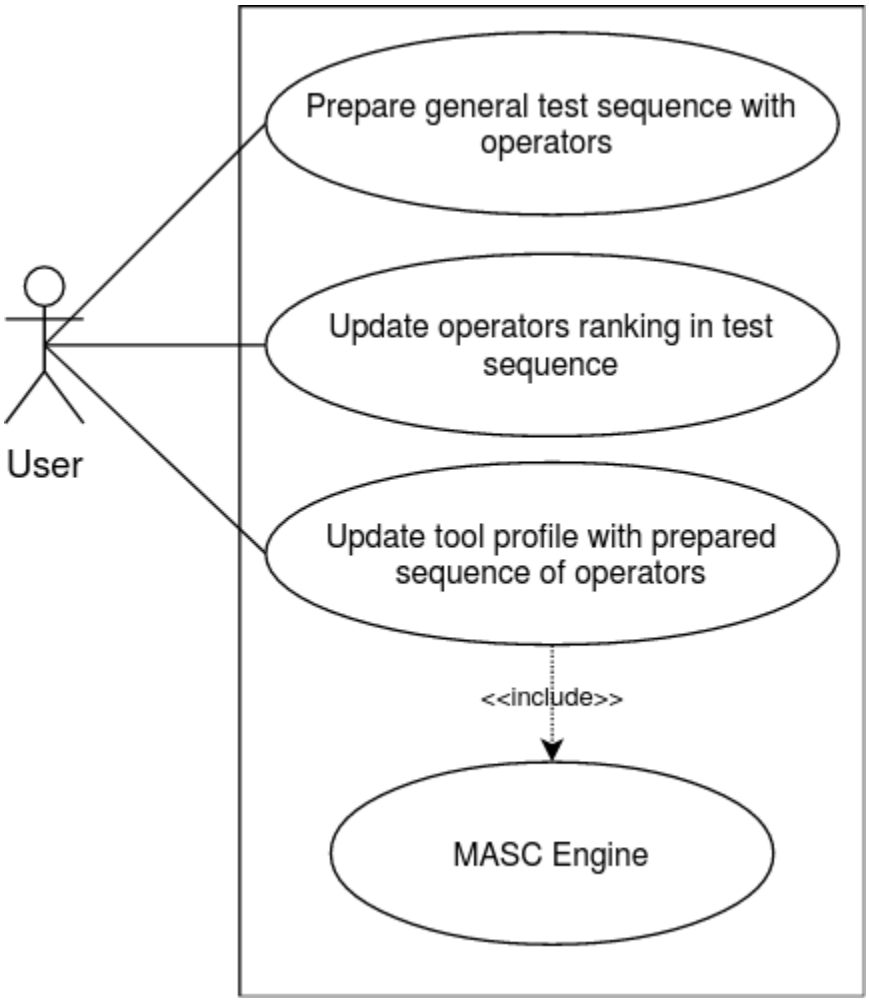
Action: User runs a tool on the mutated codes
Reply: Test reports are generated by cryptodectors

1.4.3 prepare profile: Prepare tool profile analyzing the previously uploaded report for the app.

Action: Test reports are submitted to MASC core.
Reply: Profile is created based on the report analysis

Level: 1.5

Use case name: Configurable Operator Ranking
Use case id: 1.5



Primary Actor: User

1.5.1 Prepare general test sequence with operators: The general sequence will be needed in order to analyze.

Action: A general sequence is prepared for profiling

Reply: Sequence is saved to the system

1.5.2 Update operators ranking in test sequence: Update the ranking of operators for mutating codes.

Action: User updates prepared ranking

Reply: Changes to the sequence is saved to the system

1.5.3 Update tool profile with prepared sequence of operators: The tool profile needs to be updated if operator rankings are updated.

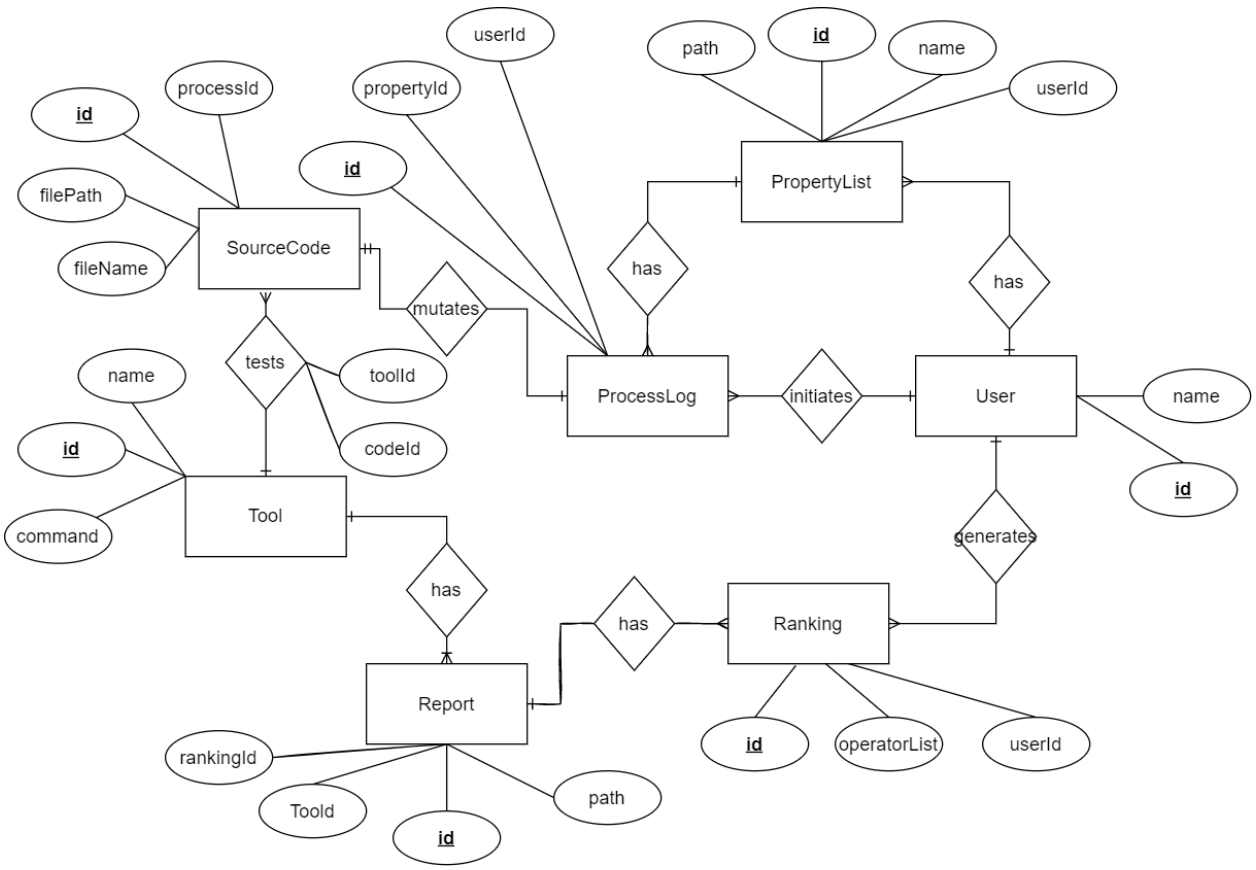
Action: User runs a new analysis with updated sequence

Reply: Tool profile is updated accordingly

6. DataBase Modeling

The systems uses and generates some critical data which needs to properly managed in order to run the whole process smoothly that is why we have generated a database model

6.1 ER Diagram



6.2 Schema

The following tables are the schemas of the proposed database.

PropertyList		
Attribute	Type	Size
id	string	8
name	string	50

fileName	string	100
path	string	200
userId	string	8

User		
Attribute	Type	Size
id	string	8
name	string	50

Ranking		
Attribute	Type	Size
id	string	8
userId	string	8
operatorList	list	500

ProcessLog		
Attribute	Type	Size
id	string	8
userId	string	8
propertyId	string	8
status	int	8
startedAt	time	100

SourceCode		
Attribute	Type	Size

id	string	8
processId	string	8
fileName	string	200
filePath	string	200
outputFilePath	string	200

Tests		
Attribute	Type	Size
id	string	8
codeId	string	8
toolId	string	8

Tool		
Attribute	Type	Size
id	string	8
name	string	100
command	string	100

Report		
Attribute	Type	Size
id	string	8
toolId	string	8
rankingId	string	8
fileName	string	100
filePath	string	200

7. Class Based Diagram

Class-based Modeling represents the object. The system manipulates the operations. The elements of the class-based model consist of classes and objects, attributes, operations, class – responsibility - collaborator (CRS) models.

7.1 General Classification

To identify the potential classes, we have first selected the nouns from the solution space of the story. We have used seven general characteristics for this. The seven general characteristics are as follows:

1. External entities that produce or consume information to be used by a computer based system
2. Things that are part of the information domain for the problem
3. Events that occur within the context of system operation
4. Roles played by people who interact with the system
5. Organizational units that are relevant to an application
6. Places that establish the context of the problem and the overall function of the system.
7. Structures that define a class of objects or related classes of objects

Now running general classification on the identified nouns

1. Cryptography 2
2. Softwares 2
3. APIS 2,7
4. Threat 3
5. Flaws 3
6. Stakeholders 1,4
7. Crypto-detectors 1,2,5,7
8. Framework 2
9. Analysis 2,3
10. Android 2,7
11. Java 2,7
12. Operators 1,2,7
13. Code 2,7
14. Upload 3
15. Status 2,3
16. Process 3,5,7
17. Track 3,7

18. Profile 2,5,7
19. Tools 1,2,5,7
20. Knowledge 2
21. Web APP 2,5.7
22. MASC Lab 2,3,5,7
23. MASC Engine 2,3,5,7
24. File 1,2,7
25. Environment 2,7
26. Management 3,5,7
27. History 2,7
28. User 4
29. Wrapper 2,7
30. Cryptoguard 1,2,7
31. SARIF 5,7
32. Configurable Ranking 2,3,7
33. Module 2,7
34. Resource 2,5,7
35. Properties 2,5,7

Based on general classification of the nouns, the ones fulfilling three or more classification criteria can be considered as potential classes. The only exceptions are the nouns *wrapper*.
Although they only cover two criteria, they may have a lot of operations involved with them. So they are also being considered potential classes.
The list of potential classes after general classification -

- Crypto-detector
- Operators
- Process
- Profile
- Tools
- Web APP
- MASC Lab
- MASC Engine
- File
- Management
- Cryptoguard
- Configurable Ranking
- Resource
- Properties
- Wrapper

7.2 Selection criteria

The candidate classes are then selected as classes by six Selection Criteria. A candidate class generally becomes a class when it fulfills around three characteristics. These characteristics are -

1. Retain information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Now running selection criteria on the nouns, that is selection criteria of the candidate classes

- Crypto-detector 1,2,3
- Operators 3,4
- Process 2,5
- Profile 1,2,3,4
- Tools 1,2,3,4
- Web APP 2,3,4,5,6
- MASC Lab 1,2,3,4,5,6
- MASC Engine 1,2,3,4,5,6
- File 1,2
- Management 1,2,5
- Cryptoguard 1,3
- Configurable Ranking 1,2,3,4,5
- Resource 1,3,4
- Properties 1,3,4
- Wrapper 2,5,6

Fulfilling nouns - The nouns fulfilling three or more criteria can be selected as finalized classes. In this case, the fulfilling nouns are -

- Crypto-detector
- Profile
- Tools
- Web App
- MASC Lab.
- MASC Engine,
- Management,

- Configurable Ranking,
- Resource,
- Properties
- Wrapper.

Further analysis shows that -

- Profiles and tools are expressing the same entity in our systems. They refer to the results of a crypto detector. That is why we can combine them and create new class as ToolProfiling
- CryptoDetectors act as a source of data. Mutations will be ran on them and they will act as a source for information. That is why it deserves a separate class.
- Resource and properties both this class are primitive and management mainly deals with their operation. That is why we can combine this classes and create a separate board named ResourceManagemer.
- Besides there shall be another type of resources which are uploaded zip files for mutations. For those we introduce another class that is SourceProduct.
- Configurable Ranking refers to modifying the general ranking criteria for crypto detectors as all crypto detectors shall necessarily not strongly meet every operator.
- CyprtoGuardWrapper class deals with wrapper as this will convert the report of cryptoguard to SARIF format.
- SARIF is a standard report format. It will be used for parsing the reports of the crypto detectors and prepare for profiling. We will name this class SARIFReport
- MASCLab is a part of the web app. It will take input using resource manager and gives mutated code as output
- MASCEngine is also a part of the web app. It will take input using resource manager and a source code zip from the user and gets mutated code as output
- And over all for proper management of MASC Engine and ToolProfiling we will introduce a class as ProcessManager. It will deal with the states of the processes and keep track of the process and also update them as necessary.

So the final identified classes are

1. MASCLab
2. MASCEngine
3. ProcessManager
4. ResourceManager
5. SourceProduct
6. ToolProfiling
7. ConfigurbaleRanking
8. CyprtoGuardWrapper
9. SARIFReport
10. CryptoDetectors

7.3 Class Cards

MASCLab	
Attributes	Methods
- operatorType - ResourceManager - mascCommand - outputFilePath - outputCode	+ prepareConfigurationFromresource() + runMASCmainScope() + getoutputFilePath() + collectOutput() + prepareOutput()
Responsibilities	Collaborators
● Gather user selected property file ● Run MASC main scope with given configuration ● Gather output and display to user	● ResourceManager

MASCEngine	
Attributes	Methods
- mutationScope - ResourceManager - ProcessManager - SourceProduct - outputFilePath	+ prepareConfigurationFromresource() + runMASCscope(string mutationScope) + saveProcessState() + getSourceProducts() + getProcessHistory()
Responsibilities	Collaborators
● Gather user selected property file ● Run MASC with selected scope and with given configuration ● Save process state ● Get history of all ran processes ● Gather output files paths	● ResourceManager ● Processmanager ● SourceProduct

ProcessManager	
Attributes	Methods
- processid - processStatus - startTime - processlogs	+ getProcessStateById() + saveprocessState() + updateProcessState(string id) + getAllProcesses() + getProcesslogs()

Responsibilities	Collaborators
● Keep track of running processes ● Update status when process completed ● Manage datas of all processes	

ResourceManager	
Attributes	Methods
- Id - name - fileName - filePath - file - operatorType	+ saveNewProperty() + uploadNewPropertyFile() + updateProperty(string id) + deleteProperty(string id) + getListOfAllproperties()
Responsibilities	Collaborators
● Store, update and delete all property files	

SourceProduct	
Attributes	Methods
- id - fileName - filePath - MASCEngine	+ storeZipFile() + getMutatedCode() + saveMutatedCode() + deleteMutatedCode() + downloadMutatedCode() + getSourceproductByID(string id)
Responsibilities	Collaborators
● Store user uploaded zip files for mutations ● Store Mutated codes for further use	● MASCEngine

ToolProfiling	
Attributes	Methods
- CryptoDetector - ToolId - operatorRanking - SarifReport	+ getToolInfo() + getALLSarifReportofTool() + updateOpertorRankingForToolProfile() + updateProfile()
Responsibilities	Collaborators
● Keep track of the respective crypto detectors ● Get all reports for the corresponding tools ● Get the ranking criteria for the tool ● Update profile upon new report	● ConfigurableRanking ● SarifReport ● CryptoDetectors

ConfigurableRanking	
Attributes	Methods
- Id - operatorList - rankingList	+ prepareNewRanking() + updateExistingRanking() + addOpertorToList() + removeoperatorFromList()
Responsibilities	Collaborators
● Store general ranking of operators for mutation and testing ● Update operator ranking on demand	

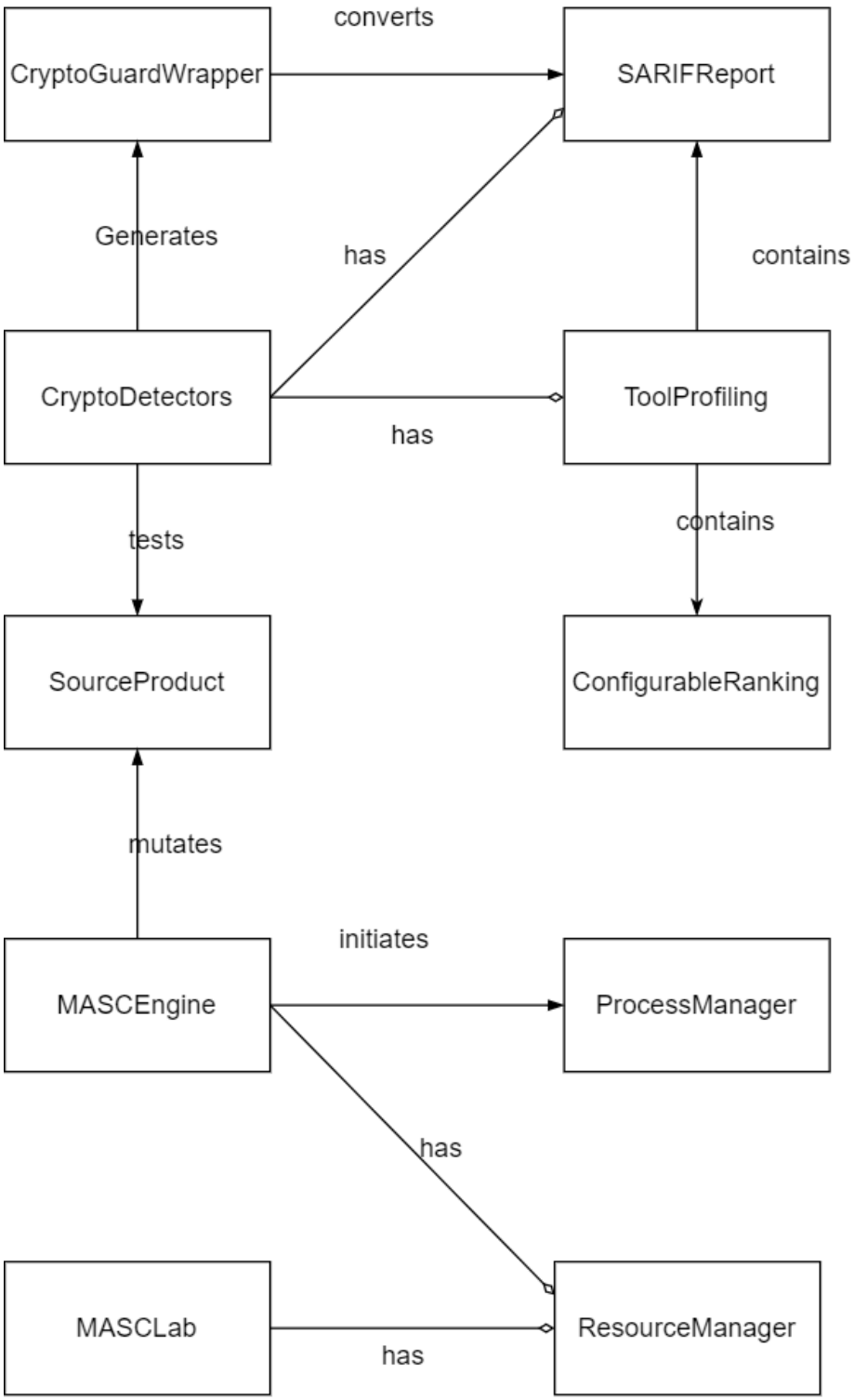
CryptoguardWrapper	
Attributes	Methods
- directory - fileName - tokens - Report - SARIFReport	+ uploadNonSarifReport() + tokenizeReport() + extractVulnerabilities() + generateSarifReport()

Responsibilities	Collaborators
● Convert report to standard Sarif format ● Extract Vulnerabilities	● SARIFReport

SARIFReport	
Attributes	Methods
- filepath - fileName - cryptoDetector - Contents - operators	+ getReportContent() + exxtractMutantedOperators() + reportAnalysis()
Responsibilities	Collaborators
● Store SARIF reports ● Analysis report ● Prepare report for tool profiling	● CryptoDetectors

CryptoDetectors	
Attributes	Methods
- toolName - toolDirectory - SARIFReport - outputReportFilePaths - runCommands - SourceProducts	+ getToolName() + setToolName() + getToolOutputFilePaths() + getToolRunCommand() + setToolRunCommand() + getSARIFReportForTool() + saveSARIFReportagainstTool() + getSourceProductById(string id)
Responsibilities	Collaborators
● Store and fetch related datas of a tool that is a crypto detector	● SARIFReport ● SourceProducts

7.4 Class Modeling Diagram



8. Architectural Design

As architectural design begins, the software to be developed must be put into context—that is, the design should define the external entities that the software interacts with and the nature of the interaction. This information can generally be acquired from the requirements model and all other information gathered during requirements engineering.

Software architectural design represents the structure of the data and program components that are required to build a computer-based system. There are 4 steps in architectural design:

1. Representing the system in context

2. Define archetypes

3. Refine the architecture into components

4. Describe instantiations of the system

8.1 Representing the system in context

At the architectural design level, a software architect uses an architectural context diagram (ACD) to model the manner in which software interacts with entities external to its boundaries. While representing the system in context, systems that interoperate with the target system are represented as,

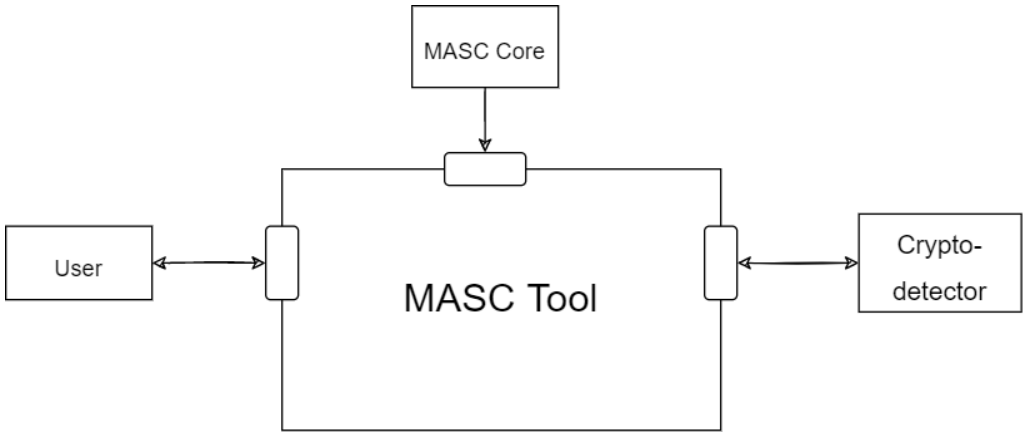
☐ Superordinate systems—those systems that use the target system as part of some higher-level processing scheme.

☐ Subordinate systems—those systems that are used by the target system and provide data or processing that are necessary to complete target system functionality.

☐ Peer-level systems—those systems that interact on a peer-to-peer basis (i.e., information is either produced or consumed by the peers and the target system.

☐ Actors—entities (people, devices) that interact with the target system by producing or consuming information that is necessary for requisite processing. Each of these external entities communicates with the target system through an interface (the small shaded rectangles).

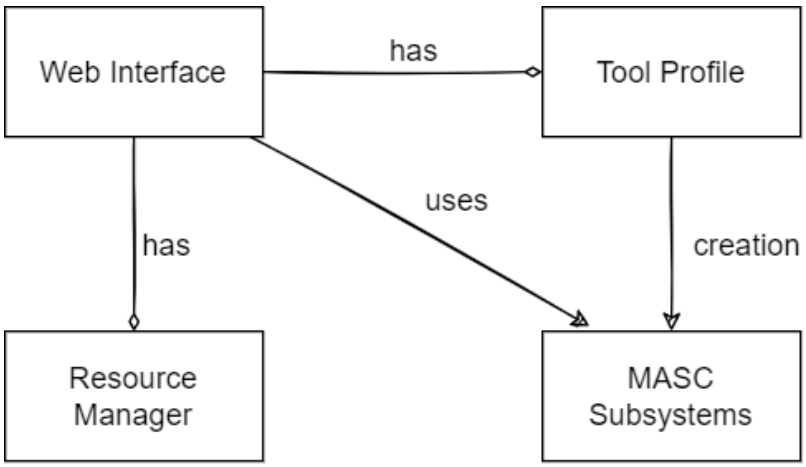
Representation of MASC Subsystems in Architectural Context Diagram (ACD) :



8.2 Define Archetypes

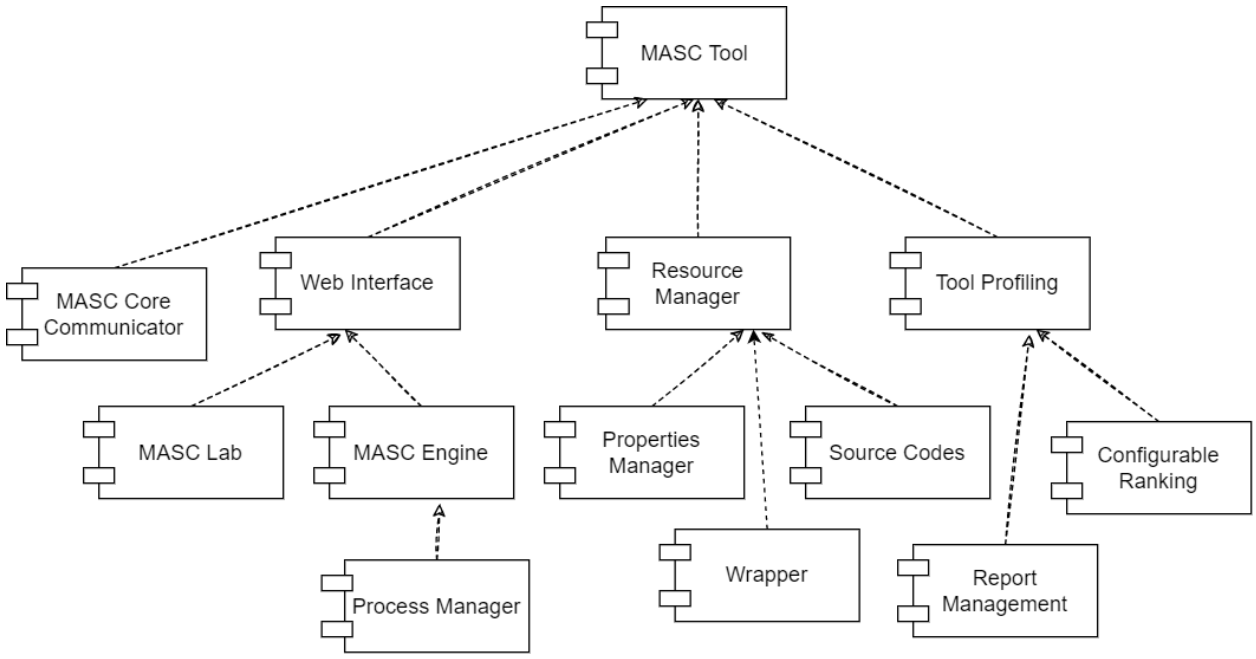
The target system architecture is composed of these archetypes, which represent stable elements of the architecture but may be instantiated many different ways based on the behavior of the system.In many cases, archetypes can be derived by examining the analysis classes defined as part of the requirements model. In this cases the archetypes are

- ☐ Web Interface
- ☐ Resource Manager
- ☐ Tool Profile
- ☐ MASC Subsystem



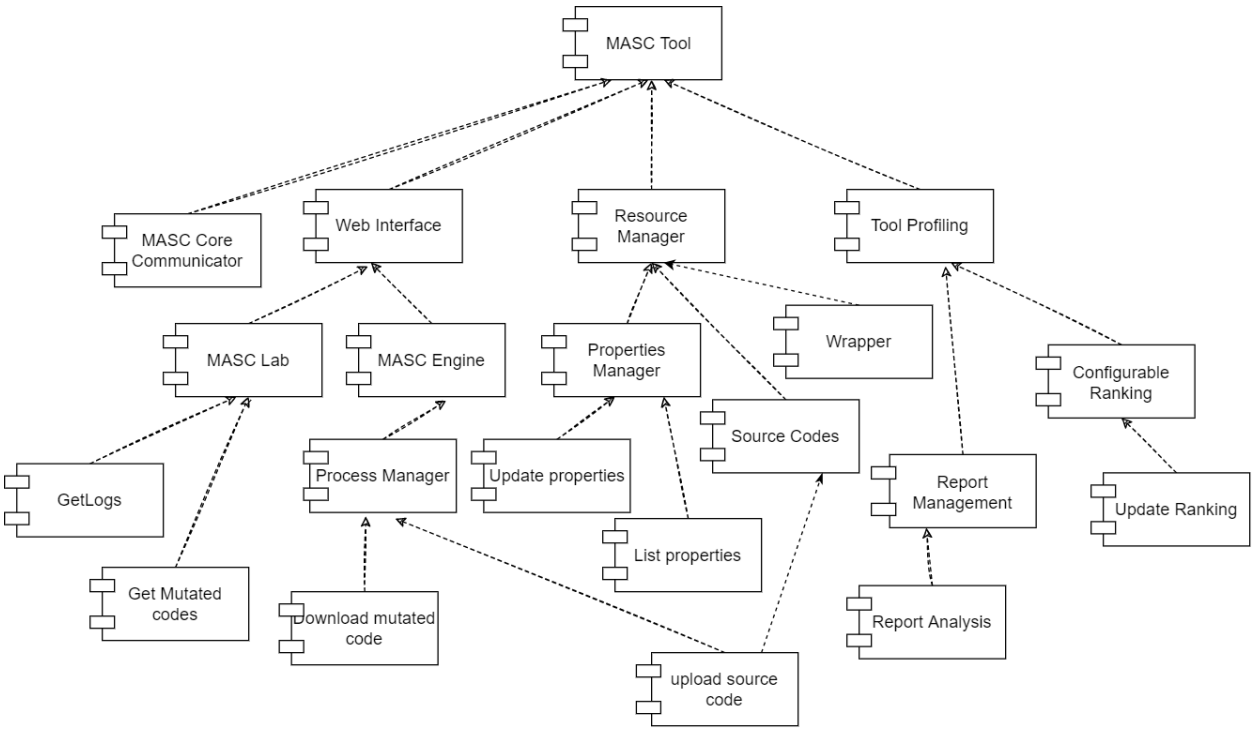
8.3 Refine the Architecture into Components

The interfaces depicted in the architecture context diagram (ACD) imply one or more specialized components that process the data that flows across the interface. In some cases (e.g., a graphical user interface), a complete subsystem architecture with many components must be designed.
Architectural Structure with top-level Components:



8.4 Describe instantiations of the system

To accomplish this, an actual instantiation of the architecture is developed. By this we mean that the architecture is applied to a specific problem with the intent of demonstrating that the structure and components are appropriate.



8.5 Mapping requirements to software architecture

The considerations that were made during the designing process to cover the requirements are discussed below

Normal Requirements

- Web App

  MASC Lab and MASC Engine this two deals with the web app providing the user an interface for using MASC.

- Process & Data Manager

  Resource manager deals with the uploaded property files and source codes and Process manager under the MASC engine keeps track of the processes.

- Wrapper for Cryptoguard

  The wrapper component under resource management will act as the main wrapper component for converting cryptoguard generated reports to SARIF format.

Expected Requirements

- Profiling for tools

  Tool profile component from the top level overview directly interacts with this requirements which can be further divided into subsystems

- Configurable Operator Ranking module

  Under tool profiling, configurable ranking modules and update ranking modules will be the solution for this requirement.

- Resource Management

  The resource management module will directly handle any resources uploaded to the or generated via the system.

- Logging Mechanism

  MASC Core communicator will handle this section of log generation and management.

Exciting Requirements

- Research and developments of new operators and schemas
- Research and development of new mutation scopes
- Enhancement and improvement of MASC Core

Fulfilling these exciting requirements include the MASC core which is a super ordinate system to the tool.
Overall, it can be said that this architecture successfully satisfies all the requirements.

9. Preliminary test plan

The above mentioned sub systems of MASC will go through Black-box testing. That is the features will be tested top to bottom.

9.1 Summary of Items & Features To be Tested

The following expected behaviors will be tested:

Web Front End

1. MASC Lab
    a. Users can select properties (configurations) from previously uploaded files or can upload new files if necessary.
    b. Visualize the properties file and make them editable
    c. Run MASC main scope with the given properties
    d. Visualize outputs that is mutated codes
    e. Display system errors or outputs if any
2. MASC Engine
    a. Users can select properties (configurations) from previously uploaded files or can upload new files if necessary.
    b. Users can select mutation scope
    c. User uploads source code for mutation
    d. Keep track of the running processes and once finished the user can download the mutated source code.

Resource Management

3. Properties Management
    a. Upload properties
    b. Edit properties
    c. Delete properties
    d. Fetch all available properties
4. Process Management
    a. Fetch all processes
    b. Update process state once process (MASC Engine) finished

Wrapper for Cryptoguard

5. MASC shall take the Cryptoguard generated report and convert it to SARIF format.

Tool Profiling

6. Run crypto detectors on mutated codes and generate report
    a. Mutate apps using MASC engine
    b. Build Profile on reports
    c. Report analysis, find undetected mutants
7. Update operator ranking for report analysis and building profile
8. MASC core logging Mechanism

9.2 Validation criteria and testing plan

Behavior	Validation criteria	Test plan	Expected Result
1a, 2a, 3a	Properties are selectable and list is updated on uploading new property	❖ Upload new properties and inspect list of properties ❖ Upload files with various extensions	❖ List of properties are available for user to select ❖ User can upload new property file
1b, 3b	Property file outputs are property visible and editable to the user. _____ Updates are saved to proper files	❖ Upload property file with data ❖ Upload various files with different extensions ❖ Upload empty property file ❖ Add lines to uploaded file ❖ Remove lines from file ❖ Inspect updated files and ensure changes	❖ User can view and update property files and changes are saved ❖ User can upload property files with proper extension
1c, 6a, 8	MASC main scope mutates code and delivers a log file	❖ Give various mutation files with different configurations and run masc lab	❖ A log file from upon running MASC core
1d	Parse log files and get all mutated code snippets for the selected configuration	❖ Manually visualize if mutated codes for operators of given type is present	❖ Mutated codes for selected operators are displayed
1e	Provide system outputs for running process	❖ Submit wrong configuration and see if system reports the error ❖ SSubmit proper configuration and see if system gives standard outputs	❖ Errors are provided to the user if any

2c	Source code is properly uploaded and accessible to the system	❖ Upload unzipped files ❖ Upload zip files ❖ Upload file without source codes in java	❖ Source code zip is uploaded and ready for mutation
2d. 4a, 4b	Initiate process and once finished status is updated	❖ Run MASC engine and inspect process ❖ Check status when process finished ❖ Check history of all process run	❖ Process status is updated once process is completed
3c	Property file is deleted with respective information	❖ Delete property files	❖ File is not on the property list
5	Report converted to SARIF format _____ Error thrown when report is not convertible	❖ Upload report provided by Cryptoguard ❖ Upload report from other detectors ❖ Upload report already converted to SARIF format	❖ SARIF report is delivered ❖ Error upon not providing proper report for conversion
6b	Tool profile built with proper information	❖ Provide tool info along with report ❖ Run profiling with updated ranking	❖ All reports, analysis and tool infos are visible
6c	Detect undetected mutants from the report	❖ Upload reports with specific mutants which are not detected	❖ All undetected mutants are detected from the report
7	General ranking criteria update	❖ Make change in ranking criteria	❖ Ranking update saved

10. Conclusion

The major objective of MASC is to assess how well the most advanced crypto-detectors work in spotting misuse of the crypto-API. Although MASC functions effectively as a system right now, there are some possible areas where its usability and extensibility might be enhanced. MASC tool works to make MASC a fully operational ready to use system for end users. The document carries technical details of this endeavor. The report starts with an introduction to MASC following a user story and QFD. Then the use case diagrams for the given tool have been added. Data management is a crucial part of this system hence the ER diagram and schemas have been introduced in the database modeling section. The following section carries the class based design, class cards and class modeling diagram. The architectural design of the system has also been added and it has been mapped with the requirements.The high-level explanation of the testing objectives, the list of objects and features that will be evaluated, and the validation criteria have all been supplied. Various diagrams have been added to the document for better understanding of the whole system. This analysis will surely help and enhance the efficiency in further development process.

References

[1] A. S. Ami, N. Cooper, K. Kafle, K. Moran, D. Poshyvanyk, and A. Nadkarni, “Why Crypto-detectors Fail: A Systematic Evaluation of Cryptographic Misuse Detection Techniques,” in 2022 IEEE Symposium on Security and Privacy (S&P), Los Alamitos, CA, USA, 2022, pp. 397–414. Available at: https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00024

[2] Sazzadur Rahaman, Ya Xiao, Sharmin Afrose, Fahad Shaon, Ke Tian, Miles Frantz, Danfeng (Daphne) Yao, Murat Kantarcioglu. CRYPTOGUARD: High Precision Detection of Cryptographic Vulnerabilities in Massive-sized Java Projects

[3] Veracode, “Veracode’s 10th State of Software Security Report Finds Organizations Reduce Rising ‘Security Debt’ via DevSecOps, Special Sprints,” https://www.veracode.com/veracodes-10th-state-software-security-report-finds-organizations-reduce-rising-security-debt, Oct. 2019, accessed May, 2020.