Graduate School and Research Center In communication Systems

EURECOM

Sophia Antipolis

Semester Project Report

Attack Crawler for Modern Networked Systems

Group
TRUONG Quang-Huy
VO Huynh-Dan

Supervisors:
Yves Roudier
Ludovic Apvrille

Biot, June 29, 2015

CONTENTS

INTRODUCTION

1 Project Description

Numerous large scale attacks on networked systems have been conducted in recent years, like the Zeus/Zitmo attack on mobile banking systems, the Stuxnet and Havex malware that were discovered on SCADA systems, or even the unlikely botnet infection of smart fridges. Unfortunately, the study and analysis of those attacks is quite complex and requires gathering information from very diverse sources. In addition, documenting and disseminating the knowledge acquired in this process is often a largely manual process in which a lot of technical details get lost and which is difficult to keep up to date with new vulnerabilities. The objective of this project is to evaluate which relevant information can be automatically extracted from Internet sources to better detect, understand, and model such persistent threats. In particular this project will develop a crawler for assembling all such data and to integrate them into TTool, a modeling environment featuring the SysML-Sec framework[1] defined by EURECOM and Telecom ParisTech.

2 Work Description

Our works in this project include:

- We are provided with a few models of attacks captured manually with TTool and practice modeling with an additional known attack Stuxnet attack.
- We implement a modeling assistant.

1 Introduce Stuxnet

Stuxnet[2] is a computer worm discovered in June 2010. It was designed to attack industrial control systems or set of similar systems. The goal of the attack is to reprogram industrial control systems by modifying code on programmable logic controllers (PLCs). The whole process of attack is deeply hided and covered in the system. However, the purpose and the identity of attackers are unknown, yet they are skilled and well resourced; this wasn't something that was put together in a short period of time. Four zero-day vulnerabilities are exploited in order to achieve this goal.

2 Model Stuxnet's attack in TTool

To understand clearly about Stuxnet, we describe the attack of Stuxnet as SysML-Sec attack models by using TTool. The General model is used for a general understanding of Stuxnet. Furthermore, important components are also described in detail in distinct models such as Execute_Stuxnet, Inject_To_Project_Files, and Modify_PLCs.

2.0.1 Diagram's Components

- Tree Diagram: a diagram describles all possible attacks on the system with the interconnetion between attacks and blocks.
- Block: is an actor involving in the attack case.
- Attack: a malicious action performed in a block. There are two kinds attack: normal and root attack. Root attack is a final target in an Attack Tree
- Contraint: relations between attacks such as: OR, AND, SEQUENCE, BEFORE, AFTER.

2.0.2 Stuxnet Operation

In the Gerneral model 2.1 , we can easily and quickly understand the purpose and procedure of Stuxnet. In particular, the final target of it is to modify PLCS in Siemens supervisory control and data acquisition (SCADA) system. To achieve this, Stuxnet attacks the system step by step. It takes advantages of zero-day vulnerabilities to propagate itself. In addition, it also uses rootkits, advanced techniques to hide itself from users and anti-virus software, on both Windows and the control computers it targets. Then, Stuxnet install itself and infects Siemen SIMATIC Step7 sofware. Stuxnet spreads rapidly, but it

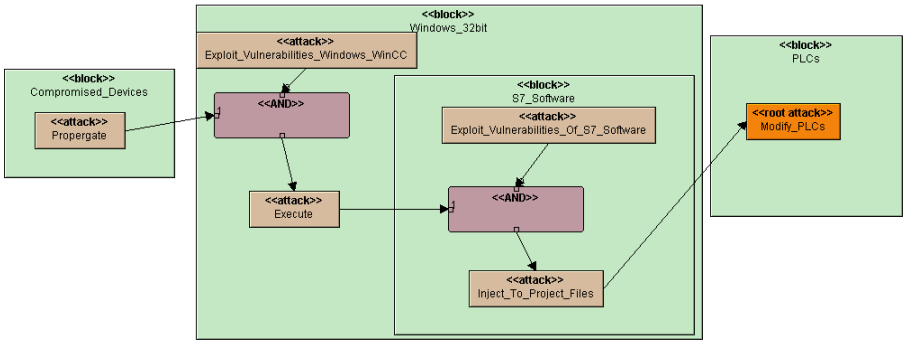


Figure 2.1: General Stuxnet diagram

also has mechanism to limit its spread. As illustrated in 2.2, Stuxnet employs numerous methods to spread itself[3]:

- Via USB flash drives, Stuxnet can use Windows LNK vulnerability or autorun.file vulnerability to spread. For LNK exploit, when .LNK files are displayed in windows explorer and the icon for a .LNK file is loaded. The malicious .LNK files contain an exploit code that will be excuted automatically . For autorun.inf file, Stuxnet inserts the malicious code into the file, along with with commands to excute this code when autorun feature is enable.

- Via Print Spooler zero-day vulnerability, this vulnerability allows Stuxnet to copy itself on remote computers, then execute the copy to infect targets.

- Via network shares vulnerability, the worm distributes itself over the network through shared folders by using either a scheduled job, and Windows Management Instrumentation. It copies itself on anyshares on remote computers, and schedule a task to execute it.

- Via WinCC. Stuxnet sends malicious SQL code to a system running with WinCC database using a hardcoded password that allows Stuxnet to be transfered to that system and excutes itself.

- Via Microsoft Server Message Block (SMB) vulnerability, Stuxnet sends a malformed path over SMB to excute arbitrary code on the remote system in order to

propagate itself.

After being executed, Stuxnet will try to attain the root privilege by using

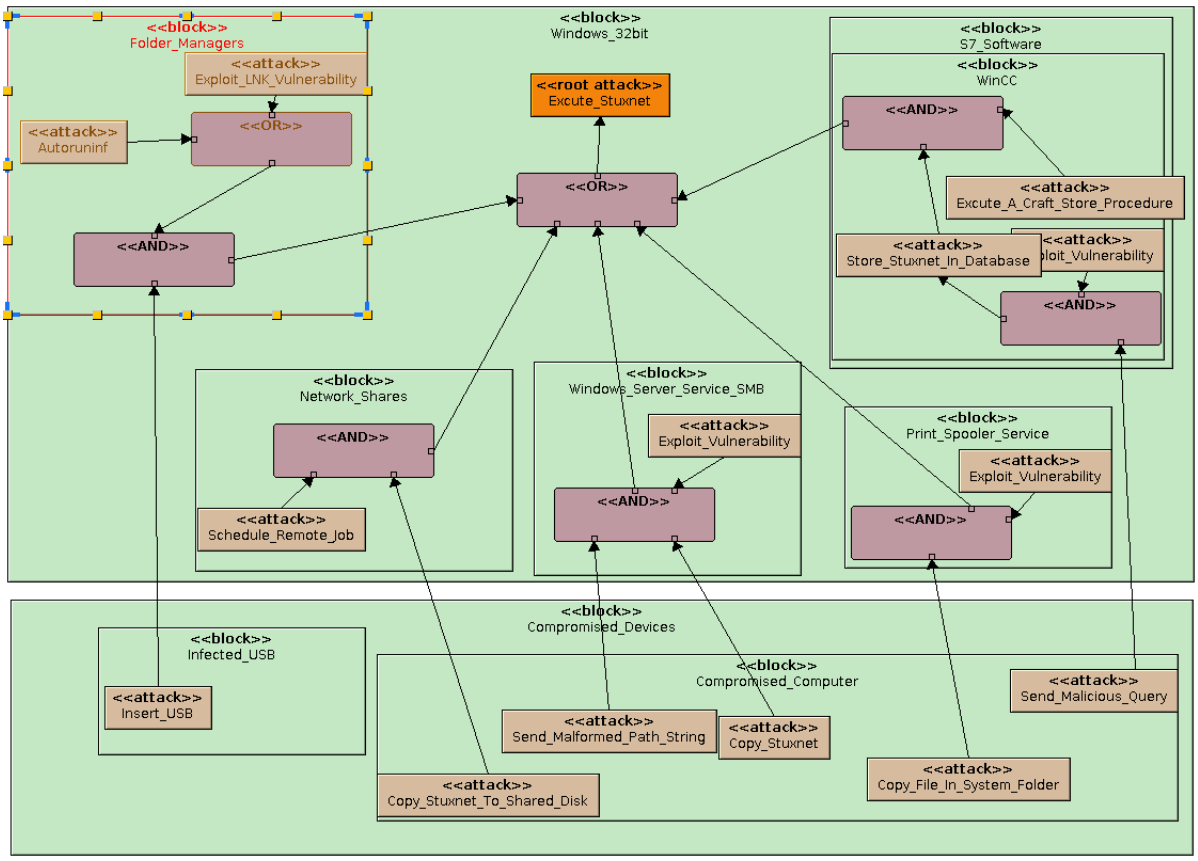


Figure 2.2: Execute Stuxnet diagram

one of two zero-day escalation of privilege attacks which are Win32k.sys and Task Scheduler Escalattion vulnerabilities. Together with advanced bypass antivirus techniques, Stuxnet tries to inject payload into the target process which infects directly to Step7 project files, and finally infects the project folder 2.3.

Once Stuxnet is executed, it will overwrite the original DLL file named *s7otbxdx.dll* that allows Stuxnet intercept any call between PLCs and Step7 Software 2.4. Eventually, Stuxnet will be able to perform actions: Monitor PLC blocks being written to and read from the PLC; Inserting and replacing or infecting blocks; And Hiding injected code.

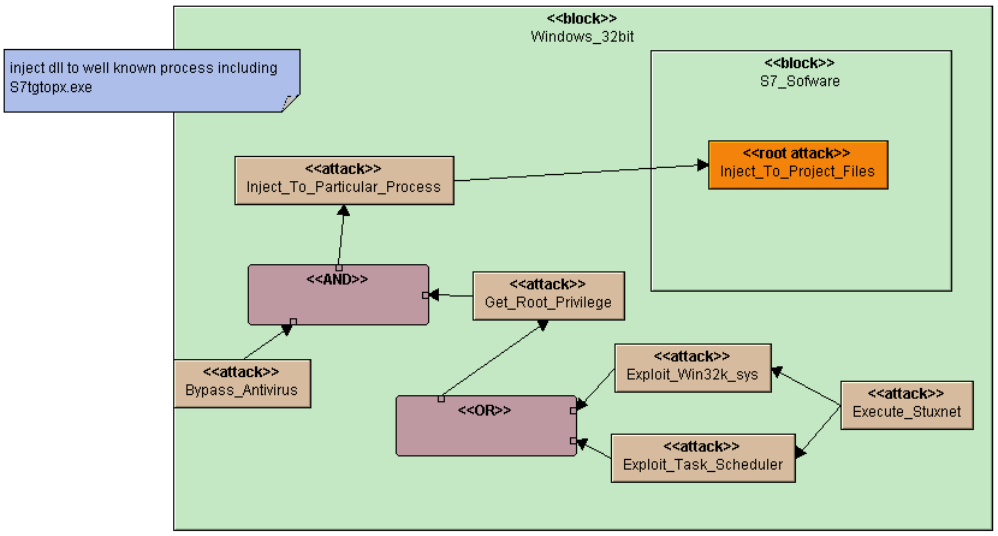


Figure 2.3: Inject Stuxnet to project files diagram

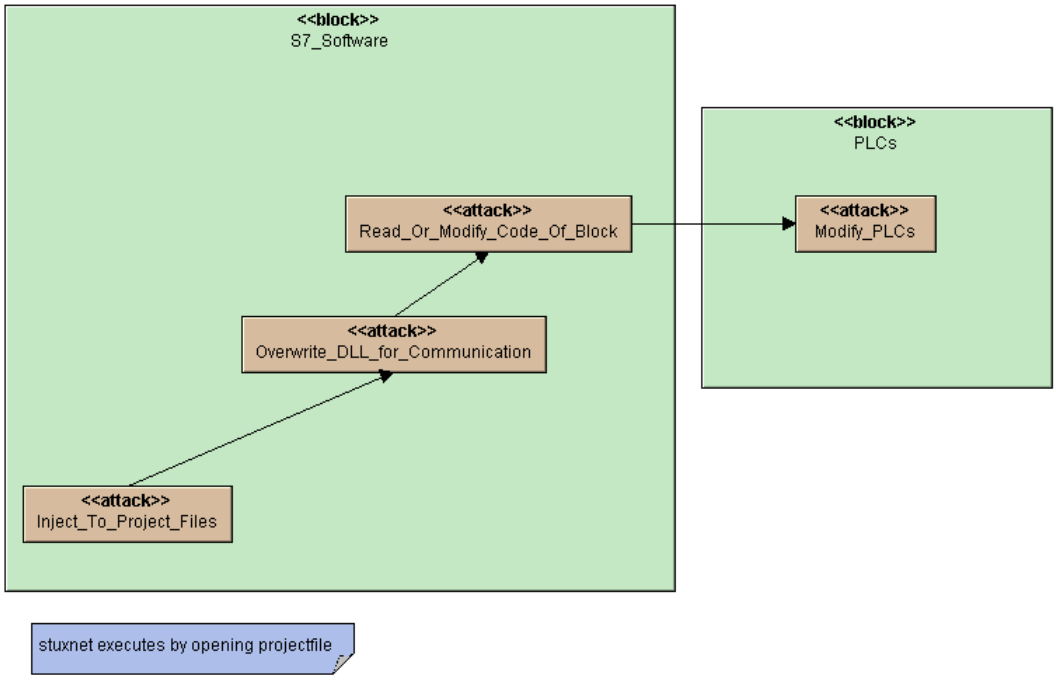


Figure 2.4: Modify PLCs diagram

1 Requirement

Currently, TTool has a function to find components inside a SysML-Sec model according to an input string provided by users. The main purpose is to support for locating components in the model. There is a concern that users would like to inquire more information about graph components. In particular, it could be the technical detail of components, the other relationship between components and the variant of the component. However, this piece of information could not be embedded in these components.

The approach in this part of the project is to extract the data from outside TTool; Especially, on the internet and databases. From the internet perspective, thanks to search engine such as Google and Google Scholar, the finding of relevant information can be performed by an URL to these services. Another way of retrieving information is to access databases, the other part of this project. Therefore, main requirements in this part includes:

- Implementing a GUI allowing users to perform extended search from outside resources by strings and following options.
- Retrieving results from the internet, namely Google and Google Scholar.
- Designing and implementing a protocol to communicate with the database in order to get information and display to users.

2 Environment

As an extention for Ttool's User Interface, this part of project involes directy to source code of TTool, which is competable with Java 7.

There is an external library named JSoup[4] is taken as advantages in order to manipulate

the request, response to Search Engine, and parse the HTML-format response. JSoup is an open source project distributed under the liberal MIT license. It can be found at http://jsoup.org/. Jsoup jar jsoup-1.8.1.jar is put in ./bin directory. Then, the Makefile is modified to ensure TTool will load the library.

```
1 basic:
  $(JAVAC) $(SOURCEPATH) $(TTOOL_SRC) $(CLASSPATH) $(TTOOL_BIN)/jsoup
    -1.8.1.jar $(TTOOL_SRC)/*.java
3
```

3 Graphic interface

As requirements, users desire to retrieve more information which either directly or indirectly relevant to elements in a working model. The task is to create an interface allowing users to interact with search engine.

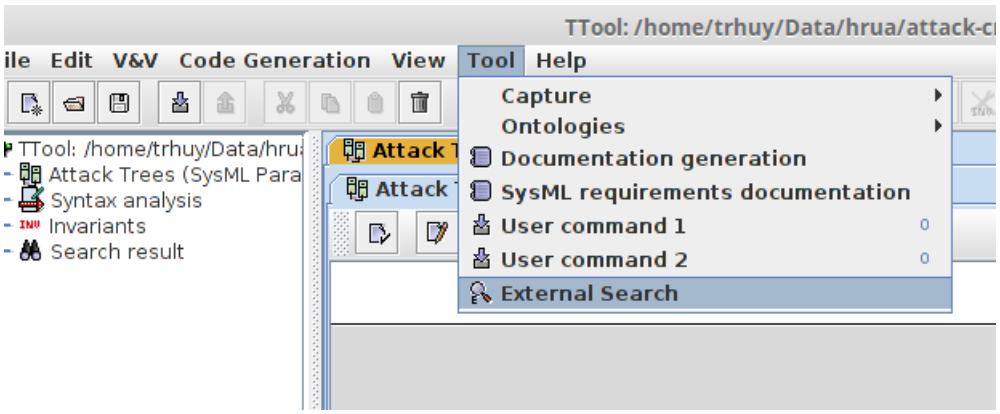
The graphic interface needs keywords from users as inputs which are obtained through several ways. In specific, there are two techniques to input keywords. Users can input them directly or simply, keywords can be obtained from component names.

For the convenience, users have four options to compose a desired keyword by:

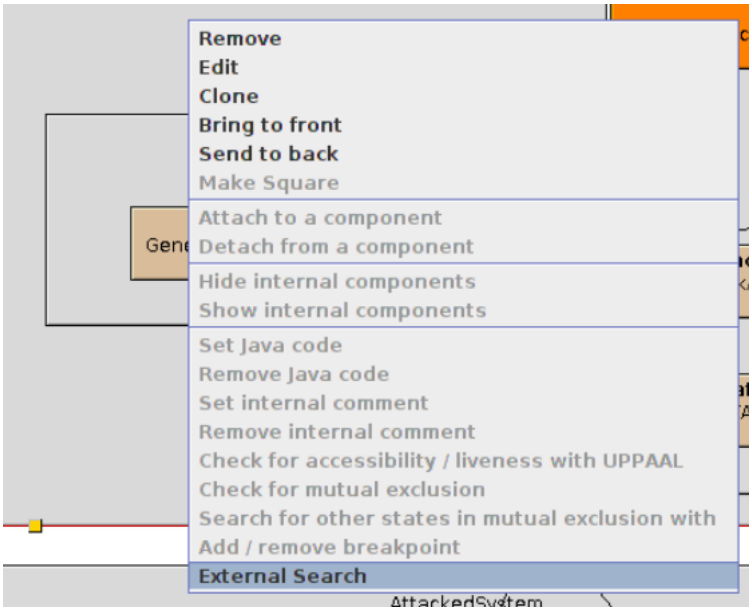
- Typing keywords into the search field on the menu bar of TTool.
- Pressing Ctrl and left-click on a component to get a name as keyword.
- Pressing Ctrl and left-click on components consequently, then right-click on the model and select option "External Search". The list of names of selected components will be combined as a keyword.
- Pressing Ctrl - Alt and left-click on a component. The form will be displayed with the name of selected component.
- When the form is shown, users can modify a keyword by pressing Ctrl and left click on that component.
- In addition, users modify a keyword directly on the search field of the dialog

To open the External search dialog, users can perform those actions:

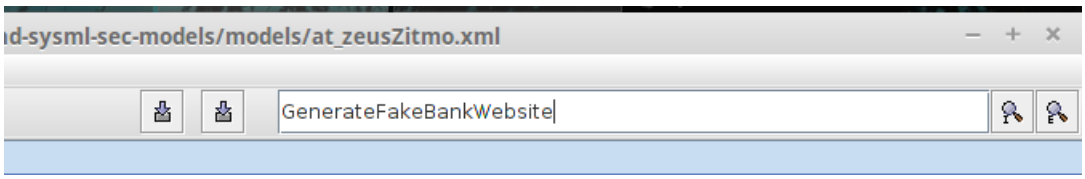
- From the main menu Tool → External Search



- From working graph, through the context menu (Right click → External Search)

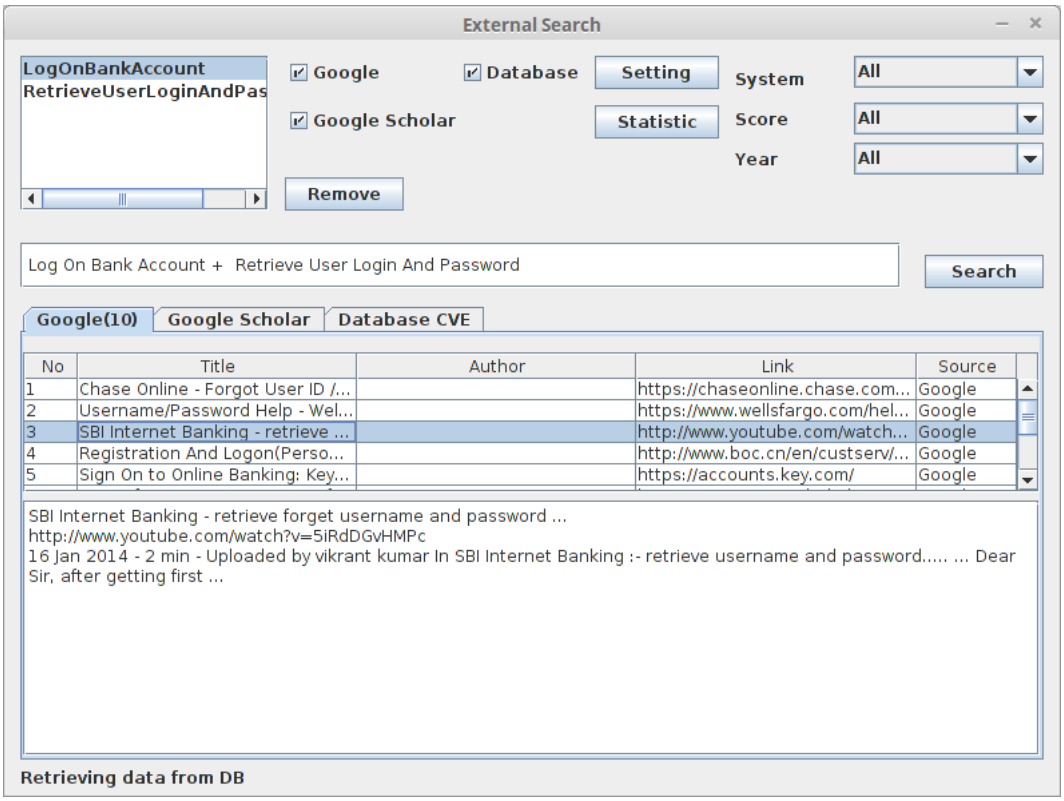


- From the menu bar, through the icon External Search.



- Pressing Ctrl-Alt and left-click on a component. The form are displayed with the name of selected component.

On the dialog, a query is constructed by collecting the keyword, options, target data resources. As well as, there are tables and text fields to represent searching results.



To optimize the keyword for searching, name of selected components will automatically split into distinct words.

For each target data resource such as Google, Google Scholar, and Database, there are tabs to display those results separately.

The related source code for the External Search GUI :

- *./src/ui/window/JDialogSearchBox.java* : This is the main implimentation of GUI for Searching Dialog.
- *./src/ui/TDiagramPanel.java* : Menu context is added into the option for External Search.
- *./src/ui/TDiagramMouseManager.java* : A few lines of code for caching the event of mouse when clicking on components with special keys (Ctrl, Alt)
- *./src/ui/MainGUI.java* : The instance of JDialogSearchBox belongs to MainGUI which allows the search function can be accessed by every context of TTool. An

added code supports to reuse the search field in menu bar for both Internal Search and External Search. It also allows to open a External Search Dialog by a button beside the search field.

- *./src/ui/TGUIAction.java* : register new actions for Internal and External, including icons and description.

4 Internet Search

The goal of this function is to parse a returned searching result in HTML format from Google search engine for specific string queries. For Google and Google Scholar, the query string is sent to servers by GET method. Two parameters applied here are "&q=" and "&num=" to supply the query and number of result for that query, respectively.

The URLs are:

http://www.google.com/search?hl=en&q=<string>&num=<number>
http://scholar.google.com/scholar?ht=en&q=<string>&num=<number>

Thanks to JSoup library, DOM objects in HTML content are parsed and displayed in GUI. Structure of HTML content from Google with selector syntax of CSS includes:

- List of search results : *li.g*
- Each search results has:
  - Link: *a* tag with *href*
  - Description: the element with *span.st*

Structure of HTML content from Google Scholar with selector syntax of CSS:

- List of search results : *div.gs_ri*
- Each search results has:
  - Link: the element with *h3.gs_rt > a*
  - Description: the element with *div.gs_rs*
  - Author: the element with *div.gs_a*
  - Cites: the element with *div.gs_fl > a*

The related source code for the Google Search

- ./src/myutil/GoogleSearch.java: a class represents for an article in result of google search. It has two functions to fetch data from Google and Google Scholar. 3.1
- ./src/myutil/CheckConnection.java: functions to check the connection to servers.3.2

Functions	Descriptions	Returned Type
getGoogleResult	execute the search query to Google, the result is stored in a list of GoogleSeach objects	ArrayList<GoogleSearch>
getGoogleScholarResult	execute the search query to GoogleScholar, the result is stored in a list of GoogleSeach objects	ArrayList<GoogleSearch>

Table 3.1: Functions of GoogleSearch class

Functions	Descriptions	Returned Type
checkInternetConnection	check if client can connect to default address and url.	Boolean
checkConnectionWithAddr	check if client can connect to a specific address.	Boolean

Table 3.2: Functions of CheckConnection class

5 Database Search

Similarly, this feature connects to a database created by Crawler in order to extract relevant information. Furthermore, users also have general view of vulnerabilities in the database through visualization.

5.1 Protocol

5.1.1 Objective

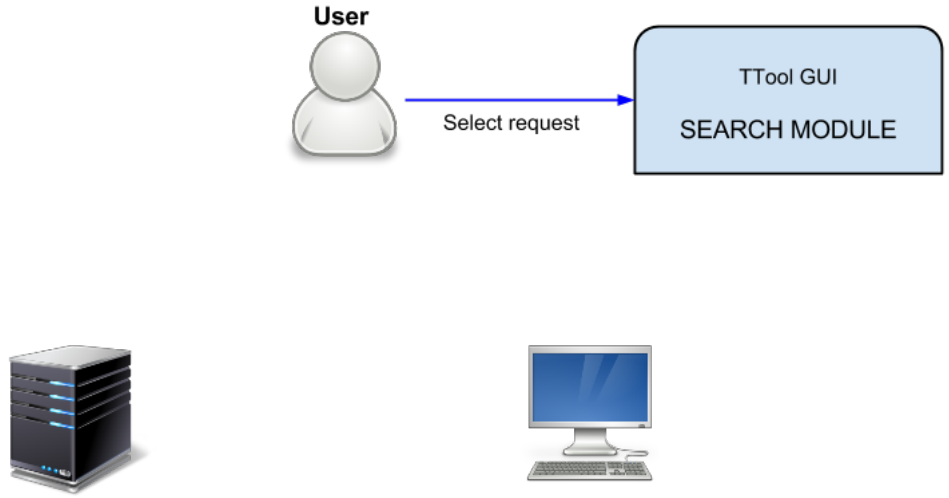
The aim of the protocol is to connect TTool and Web crawler. In particular, TTool users can retrieve relevant information from databases that needs help from Web crawler, so they need to unify a protocol to exchange information. In addition, we also need to secure the information.

5.1.2 Design

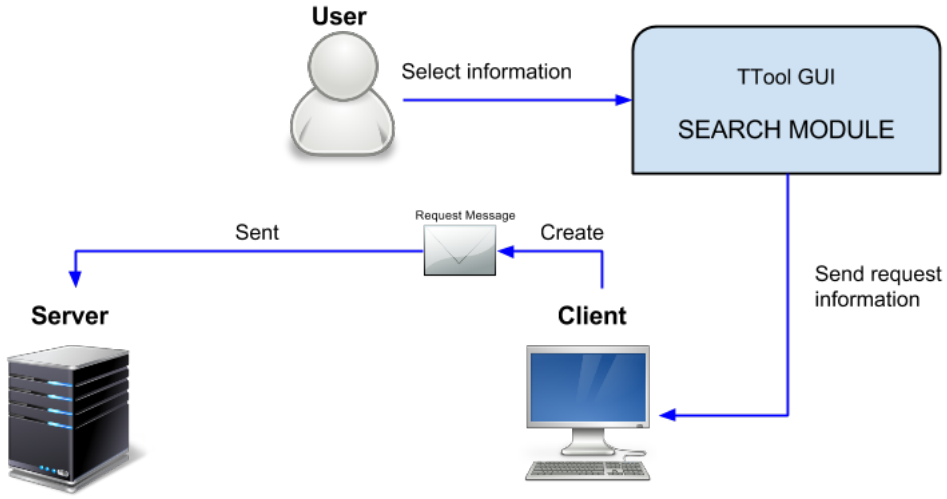
The protocol is divided into two directions, client side (TTool users) and a multithreaded server side (Web crawler). A new class named Message which is a unified structure used to supply functions for both clients and server. In addition, the protocol implements "Serializable" in order to exchange "Message" between clients and server.

Because of the confidentiality, data integrity of the message, and client-server authentication , SSLSocket technique which is a cryptographic protocol for secure Internet data transmission is implemented to make sure these properties. Exchanged messages between clients and server need to be secured. We want to make sure that clients can talk to the right server and vice versa. Furthermore, nobody has the permission to modify messages while they are being exchanged.

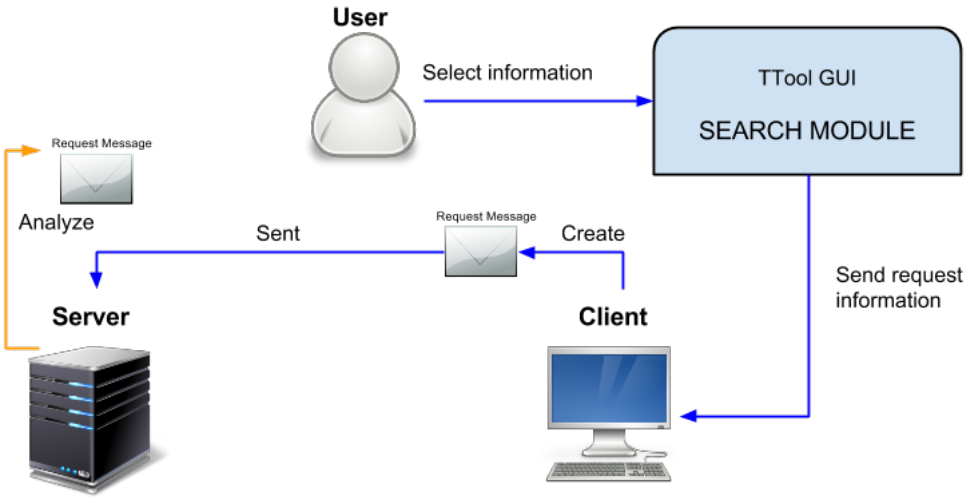
- First of all, GUI of TTool allows users to select and choose information they need to retrieve from the database.



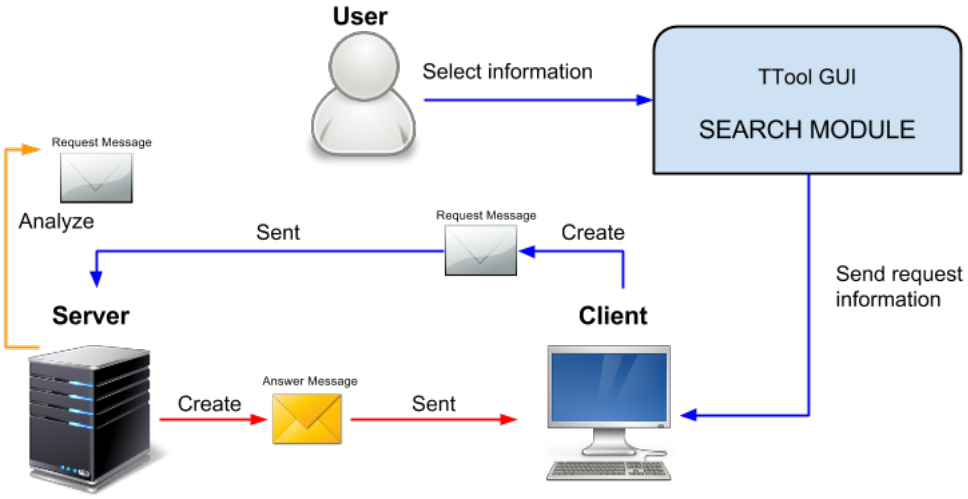
- Secondly, information is sent to client side which then is created and stored in a request message by client and sent to the server.



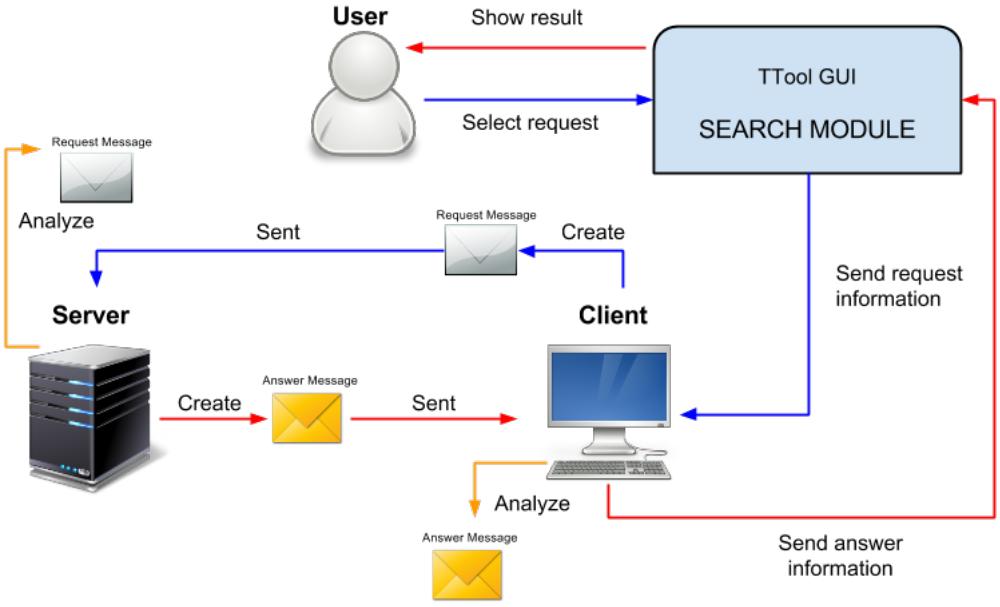
- Next, the server is responsible for analyzing the request message and retrieve, search for relevant information which matches with client requirements.



- In the next phase, relevant information is stored in an answer message and sent back to the client.



- Finally, client will analyze information and show via TTool GUI to interact with users from this message.



5.2 Implementation

5.2.1 Message

5.2.1.1 Objective Message is a unified structure between clients and server which supplies useful functions in order to exchange information between them.

5.2.1.2 Syntax

Parameters	Description	Type
cmd	Used for both client and server, to tell which type of message is. From client side, it could be a request message for searching, details, and statistic. From server side, it could be an answer message for searching, details and statistic.	String
content	Used for server to store results in order to send back to clients. It can be a string of xml, a byte array for transferring an image	ArrayList <Object>
options	Used for client side only. It will contain the name of all values in the *values* variable. The order of values in *options* will follow: **Keyword - Year - System - Score - NumberOfRecord**	ArrayList <Object>
values	Used for client side to store all the real data respectively with values in *options*.	ArrayList <Object>

Table 3.3: Variables of a message

- The used variables of a message is described in Table 3.3
- Both options and values are ArrayList<String> type and used for clients to let the server know what information the user wants to retrieve. for options, it could be an arraylist of string which contains strings both clients and server unified before. Beside that, values represents for real values respectively along with options. Take a look at one example for options and values in Table 3.4

<options>	Keyword	Year	System	Score	NumberOfRecord
<values>	Buffer-overflow	Last year	Linux	7-8	10

Table 3.4: An example of how to assign options and values

5.2.1.3 Message Types

- Search message structure is described in Table 3.6
- Detail message structure is described in Table 3.7
- Statistics message structure is described in Table 3.8
- Histogram message structure is described in Table 3.9

Function	Description	Returned Type
addOptionValueMessage	Add an element for both options and values lists	void
addKeywordMessage	Add an element into values list with "keyword" to option list	void
createRequestMessage	Create a request message from client	void
createAnswerMessage	Create an answer message from server	void
convertImageToByte	Used by server to convert an image to a byte array	byte[]
convertByteToImage	Used by clients to convert a byte array to an image	void

Table 3.5: Functions of Message

5.2.1.4 Implementation The source code is implemented in file *./src/myutils/externalSearch/Message.java* 3.5

CLIENT						**SERVER**
cmd	Search					SearchResult
content						Result from the server in XML format
options	Keyword	Year	System	Score	NoR	
values	Stuxnet	This year	Linux	7-8	10	

Table 3.6: Structure of Search message and SearchResult message

5.2.2 Client

5.2.2.1 Objective Receive request information from the Search Module, then create a request and send to server. Then, client analyze an answer message from server. Depends on which type of the answer message, client will call right functions to interact with users via TTool GUI. In specific, the type of the answer message is stored in the variable cmd (command), if command is result for search or detail, client will read and show information via the Database tab. If the command is result for details, it will show an image.

CLIENT		SERVER
cmd	Detail	DetailResult
content		Result from the server in XML format
options	Keyword	
values	CVE ID	

Table 3.7: Structute of Detail message and DetailResult message

CLIENT		SERVER
cmd	Stat	StatResult
content		Binary String
options	Keyword	
values	This can be several keyword in a string "Linux Sql Windows"	

Table 3.8: Structure of Statistics message and ResultStatistics message

5.2.2.2 Implementation Related source code relating to Client in TTool:

- ./src/myutils/externalSearch/Client.java 3.10
- ./src/myutils/externalSearch/Record.java : Used to parse the content of return messages

5.2.3 Server

5.2.3.1 Objective Server receives requests from the client through a request message, then it will analyze and search for relevant information. Finally, it will extract all information as an answer message and send back to the client.

5.2.3.2 Implementation Related source code relating to Server in Crawler *./src/web/crawler/MultiThreadSerer.java* 3.11

CLIENT		SERVER
cmd	Histogram	HistogramResult
content		Binary String
options	Keyword	
values	One key word "Linux"	

Table 3.9: Structure of Histogram message and HistogramResult message

Function	Description	Returned Type
parserAnswerMessage	Parse the message from the server depends on the type of the message	Object
send	Send a message to the server via SSL Socket and waits for response	Message

Table 3.10: Functions of Client

Function	Description	Type
analyseRequestMessage	Analyze the request message sent from the server	void
createAnswerMessage	Create an answer message with relevant information from searching or databases	Message

Table 3.11: Functions of Server

CONCLUSION

In summary, we successfully succeeded in describing Stuxnet attack by a SysML-Sec model. In addition, we designed a friendly user interface which allows users interact with databases for retrieving relevant information. Finally, we also implemented a protocol for exchanging messages between users and the web crawler database.

BIBLIOGRAPHY

[1] Ludovic Apvrille, *AVATAR-TTool A SysML Environment for the Proof of Safety and Security Properties*, available at `http://ttool.telecom-paristech.fr/docs/slidesAvatar.pdf`.

[2] Liam O Murchu Nicolas Falliere and Eric Chien, *W32.Stuxnet Dossier*, available at `https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf`.

[3] Eugene Rodionov Aleksandr Matrosov David Harley, *Stuxnet Under the Microscope*, available at `http://www.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf`.

[4] Jonathan Hedley, *jsoup cookbook*, available at `http://jsoup.org/cookbook/`.