University of Nevada, Reno

Project Title: CYBEX Automation & Virtualization

By: Alexander Parr

Co-Authors: Kathleen Lowe & Timothy Muller

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science & Engineering and the Honors Program

Thesis Advisor: Dr. Shamik Sengupta

Graduation: May 2018

Signature Page

The Honors Program

We recommend that the thesis prepared under our supervision by

Alexander Parr

Entitled

CYBEX Automation & Virtualization

be accepted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering.

_____

Dr. Shamik Sengupta, Thesis Advisor

_____

Tamara Valentine, Ph.D., Director, Honors Program. May 2018

Abstract

Cybersecurity is an issue that affects virtually everybody in the digital age. Attacks such as the Equifax breach or the Wannacry ransomware attack are becoming more and more commonplace and are becoming harder to defend against as attackers become more sophisticated. Many organizations that are breached are often entirely unaware of the breach until after it has occurred, or slow to react during the event. CYBEX is a project to help mitigate this. CYBEX is an automated cybersecurity response system designed to import and analyze log files from many organizations, develop defensive rules against an attack on any one of them, and send the solution out to all other members in order to stop attacks before they can spread. CYBEX primarily targeted at businesses in order to facilitate faster and more effective incident response. CYBEX was successful in testing at being able to securely disseminate security rules out to a group of computers and preemptively stop attacks against them.

Table of Contents

List of Tables

List of Figures

Introduction

This project was to develop a cybersecurity automation system named CYBEX in a virtualized environment. In the environment, there are three types of computers. There are computers that perform attacks, computers that are attacked and need to defend themselves, and the CYBEX server. The CYBEX software, has two components, a remote client that is installed on each of the defending (member) computers, and the software running on the server. In order to automate the defense, the CYBEX remote clients parse logs generated by the intrusion detection system Snort on each of the member devices. If an attack is detected, the remote client sanitizes and parses the log before sending information about the attack to the central server. The central server then constructs an IPTables firewall rule, logs it, and sends the information out to all of the other members, including both the ones that were and were not attacked. In this way, all members are protected in real time from attacks, even without necessarily being attacked themselves. The end result of this is a much faster response time to an attack due to humans not being involved with the process and the ability to preemptively block ongoing attacks.

Literature Review

Cybersecurity is an issue that affects virtually everybody in the digital age. Attacks such as the Equifax breach, which exposed nearly half of the US adult population to identity theft, are becoming more and more commonplace [1]. This breach occurred despite the fact that three months prior to the incident, the compromised organization had been made aware of the patch that could have prevented their loss of sensitive information. In order to minimize damage due to implementation delays, the International Telecommunications Union (ITU) drafted a recommendation for a Cybersecurity information exchange framework (CYBEX) to automate sharing of immediately actionable cyber threat intelligence[3]. Ideally, CYBEX could be integrated with existing network security infrastructure to create an automated defense system that would not require action by a human. Such a defense would proactively protect an organization from attacks that have been perpetrated upon some other organization.

Because modern cyber attackers include groups with complex political and financial goals, reverse engineering intent from computer logs of events is a complex task [6]. In order to concretely communicate about cyber threats so that meaningful analysis can be performed, the Structured Threat Information eXpression (STIX) language was developed [4]. Although this language is intended to support automation, current core use cases include decision making steps by cyber security professionals. However, if the problem were focused down from the entire cyber kill chain to simply the attacker

reconnaissance and password cracking phases, an automated defense against this subset of attack behavior might be possible without the use of a specialized language.

Sharing of cybersecurity information faces multiple obstacles. Lack of trust has been cited as a primary impediment, particularly if sensitive information might be transferred to the government or competitors [2]. An attribute based sharing framework has been proposed that would effectively preserve confidentiality by encrypting sensitive fields [7]. The information provider could then supply access to selected entities by providing the decryption key. However, if sensitive information were sanitized from the shared information, the need for encryption by the provider might be avoided. For an automated defense that is designed purely to protect against reconnaissance and password cracking activities, sanitizing ought not to degrade defensive capability, because only information about the attacker is required to be shared.

Creating an independent CYBEX functionality that is specifically designed to defend against preliminary information gathering attack behaviors is consistent with the defense in depth model. In the defense in depth model, separate layers of protection are constructed to delay an attacker by forcing them to breach each individual defensive mechanism [5]. Using this model, the outer layer of defense would be a CYBEX system that protects network attack surfaces against attacker attempts at host discovery, network mapping, port scanning, and password cracking. This would allow decoupling from an

inner CYBEX system that could specialize on the more complex task of converting cyber threat indicators into attacker behavior models.

This project implemented and tested a proof of concept version of an outer layer CYBEX system as defined above.

Specification

Summary

For this project, different requirements and use cases were made to determine the scope of what CYBEX needed to accomplish and to help define what it meant for the project to be successful. For this implementation of the project, CYBEX needed to be able to sanitize and parse information about an attack from log files generated by the Snort IDS, send that information to a central server, have the server create and send a rule out to all other members of the CYBEX system, and log its activity. Additionally, there needed to be a UI for easier user access and the information needed to be secure while traveling over the network. Finally, for the virtualization component, development and testing needed to be in a controlled and isolated environment.

Functional Requirements

Functional requirements were created for this system in order to plan use cases and test whether or not it is successful. Some of these requirements are general requirements for the system as a whole while others are specific to this particular testing environment.

Functional requirements have been prioritized into two levels. Level 1 requirements were implemented by the end of Spring 2018. Level 2 requirements were originally planned but were deemed infeasible to finish in the time frame of the project.

Level 1

F1. Virtual networking shall allow mutual communication between Attackers, potential Victims (including Member organizations), and the CYBEX Server.

F2. (Sandbox) operator shall be able to simulate attacks from Attackers to potential Victims.

F3. Virtual Attacker capability shall include Nmap port scanning application.

F4. Victims shall be able to detect intrusion attacks that have been attempted.

F5. Virtualization Victim capability shall include Snort network intrusion detection system application.

F6. Victims shall be able to block known attacks.

F7. Victim capability shall include IPTables firewall application.

F8. Victims shall be able to join the CYBEX system as Member organizations.

F9. Member organizations shall be able to leave the CYBEX system.

F10. Member organization firewall shall block previously unknown attacks that are learned vicariously via new rules from CYBEX Server.

F11. CYBEX Remote shall be able to access and read Victim alert log files.

F12. CYBEX Remote shall sanitize alert log files of identifying or sensitive information.

F13. CYBEX Remote shall parse attack signature information from alert logs.

F14. CYBEX Remote shall send attack signature information to the CYBEX Server.

F15. CYBEX Remote shall receive IPTables rules from CYBEX Server.

F16. CYBEX Remote shall implement rules received from the CYBEX Server

F17. CYBEX Server shall be able to receive alert information from CYBEX Remote at member organizations.

F18. CYBEX Server shall be able to generate IPTables firewall rules based on alert information.

F19. CYBEX Server shall send IPTables firewall rules to CYBEX Remote.

F20. CYBEX Server shall protect system data by verifying System administrator intent before executing deletions.

F21. CYBEX Server shall be able to notify system administrator of attack events.

F22. CYBEX administrator shall be able to add member organization information.

F23. CYBEX System administrator shall be able to delete member organizations from system.

F24. CYBEX System administrator shall be able to access member organization contact information.

F25. CYBEX System administrator shall be able to access attack reports.

F26. CYBEX System administrator shall use secure login to access system.

F27. CYBEX System administrator should be able to add Security device information.

F28. CYBEX Remote shall initiate secure communication sessions with the CYBEX Server.

F29. CYBEX Server shall initiate secure communication sessions with the CYBEX Remote.

F30. CYBEX Server shall be able to send new firewall rules to security infrastructure of Member organizations.

Level 2

F31. CYBEX should be capable of successful operation when intrusion detection and firewall functionalities are implemented in separate host machines within a Member organization.

F32. CYBEX should be able to identify and monitor potentially malicious alert information from Member organizations.

F33. CYBEX should be able to exclude potentially malicious alert information from rule creation and distribution functions.

Non-functional Requirements

NF1. Organizations joining and leaving CYBEX should not cause any disruptions.

NF2. CYBEX should be able to generate and distribute new rules within minutes of receiving alert information.

NF3. Member organizations should not know anything about any other potential member organizations.

NF4. CYBEX system administrator should not be able to know anything about member organizations' infrastructure.

NF5. The sandbox environment should not affect, nor be affected by, any other systems.

NF6. CYBEX shall integrate with security applications that are employed by the majority of network administrators.

NF7. CYBEX shall incorporate industry standard data protection and security protocols.

Virtual Hardware Requirements

This project used a virtual, isolated network environment for simulation and testing. All computers in the virtual network used different versions of Linux. The network was run in host-only mode in Virtualbox in order to ensure that the machines running on it would not have internet access and accidentally contact uncontrolled systems. CYBEX leverages off of the existing capabilities of Snort and IPTables in order to automate defensive measures. Member computers thus all had Snort installed and configured. CYBEX was made in order to work directly with IPTables and Snort was configured to detect a variety of attacks from the attacking machines such as network mapping attempts and password brute-force attempts.

Use Case Modeling

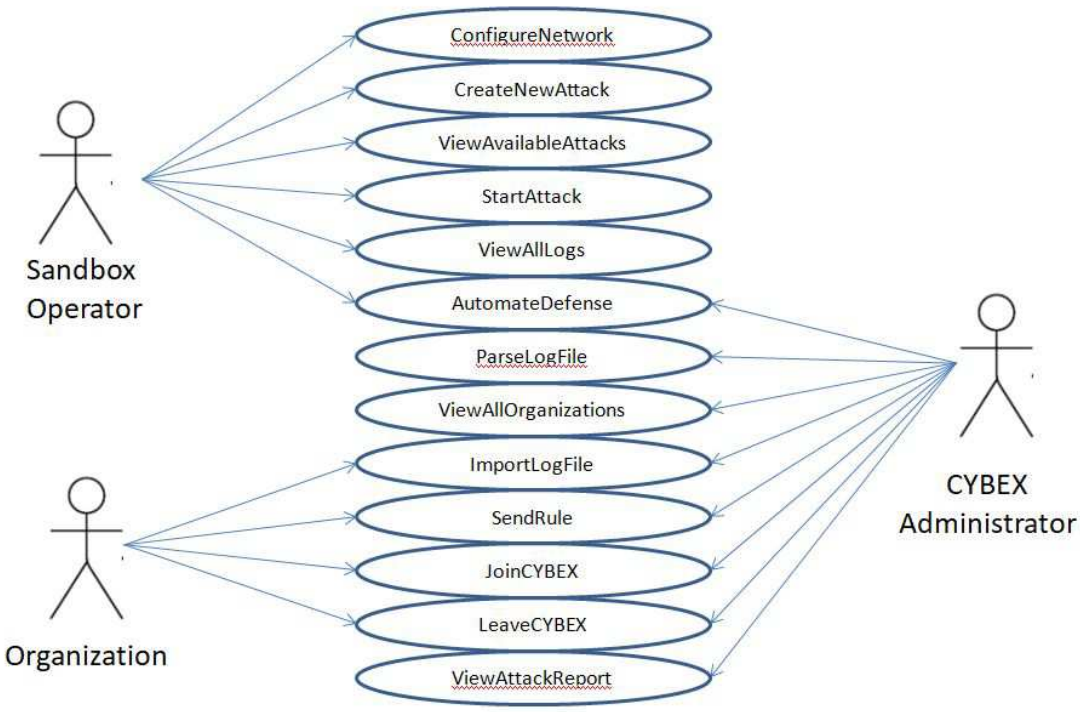

Figure 1: Use case model diagram

Case ID	Name	Description
UC-S01	ConfigureNetwork	The sandbox operator configures the systems in the sandbox environment.
UC-S02	CreateNewAttack	The sandbox operator prepares a new attack in the sandbox environment.
UC-S03	ViewAvailableAttacks	The sandbox operator views what attacks are available in the sandbox environment.
UC-S04	StartAttack	The sandbox operator executes a prepared attack in the sandbox environment.
UC-S05	ViewAllLogs	The sandbox operator views/exports all available logs in the sandbox environment.
UC-S06	AutomateDefense	The sandbox operator joins the victim computers in the sandbox to the CYBEX system for defense.
UC-C01	ViewAllOrganizations	CYBEX administrator views the information about organizations joined to CYBEX.
UC-C02	ImportLogFile	CYBEX imports a log file from a member organization.
UC-C03	ParseLogFile	CYBEX parses a log file for new alerts.
UC-C04	SendRule	CYBEX sends out a rule change to all relevant member organizations.
UC-C05	JoinCYBEX	A new organization joins the CYBEX system.
UC-C06	LeaveCYBEX	A currently joined organization leaves CYBEX.
UC-C07	ViewAttackReport	System administrator views report about attack rules generated by CYBEX.

Table 1: Use case details, including ID number, name, and description

Use Case: AutomateDefense
ID: UC-S06
Actor(s): Sandbox Operator, CYBEX
Precondition(s): 1. The sandbox network has been configured successfully. 2. At least one system on the network is configured to be a victim.
Flow of Events: 1. The victim machine(s) are registered to the CYBEX system. a. Any available defense measures on the victim machines begin receiving rule updates from the CYBEX system. b. The victim machine(s) begin sending logs to the CYBEX system.
Postcondition(s): 1. The victim machine(s) are registered to the CYBEX system.

Use Case: ImportLogFile
ID: UC-C09
Actor(s): CYBEX

Precondition(s): 1. A changed log is available for import.
Flow of Events: 1. CYBEX notices that a log file is available or has changed. 2. A copy of the log file is sent over the network to the central CYBEX machine. 3. A new checksum is computed for future change verification.
Postcondition(s): 1. CYBEX has a copy of the log file that is ready for parsing / sanitizing.

Table 2: Detailed use case templates

Requirements Traceability Matrix

	UC-S01	UC-S02	UC-S03	UC-S04	UC-S05	UC-S06	UC-C01	UC-C02	UC-C03	UC-C04	UC-C05	UC-C06	UC-C07
F1	X			X		X		X		X			
F2	X	X	X	X									X
F3	X	X	X	X									
F4	X					X							X
F5	X					X							X
F6	X					X							
F7	X					X							
F8	X										X		X
F9	X											X	
F10	X					X				X			
F11	X					X		X					X
F12	X					X			X				
F13	X					X			X				X
F14	X					X		X					X
F15	X					X				X			
F16	X					X							
F17	X					X		X					X
F18	X					X				X			
F19	X					X				X			

F20	X											X	X
F21	X					X		X	X				X
F22	X										X		
F23	X											X	
F24	X						X				X		
F25	X										X		X
F26	X						X				X	X	X
F27	X										X		
F28	X							X					
F29	X									X	X		
F30	X									X	X		
F31	X					X		X		X			
F32	X					X		X	X	X			
F33	X					X		X	X	X			X

Table 3: CYBEX Virtualization requirements traceability matrix

Design

Summary

CYBEX is broken into 3 primary components. The first is the remote client that runs on member machines. This client monitors and parses Snort logs, and sends malicious information to the CBYEX server. It also serves to receive IPTables rules from the CYBEX server and implements them on the member computer. This program just runs in a terminal as it is designed to be turned on and forgotten about.

The second major component is the actual CYBEX server. This receives malicious information from a client and creates an IPTables rule to defend against the attack. It then logs information about it and sends the rule to all members of the CYBEX system.

The final major component of CYBEX is the UI/database. This serves as the primary human interface to the program. It allows an administrator to add or remove members from CYBEX and view operational logs.

In order to test CYBEX in a controlled manner, we also developed a sandbox environment for it. The sandbox environment serves to simulate attacks on a system and connect potential victim machines to a CYBEX system for defense.

High-level and Medium-level Design

System-level Diagram: Context Model

Figure 2 shows the context diagram for the CYBEX system. It shows the structure of how different components of the project are related to each other and what they have access to.

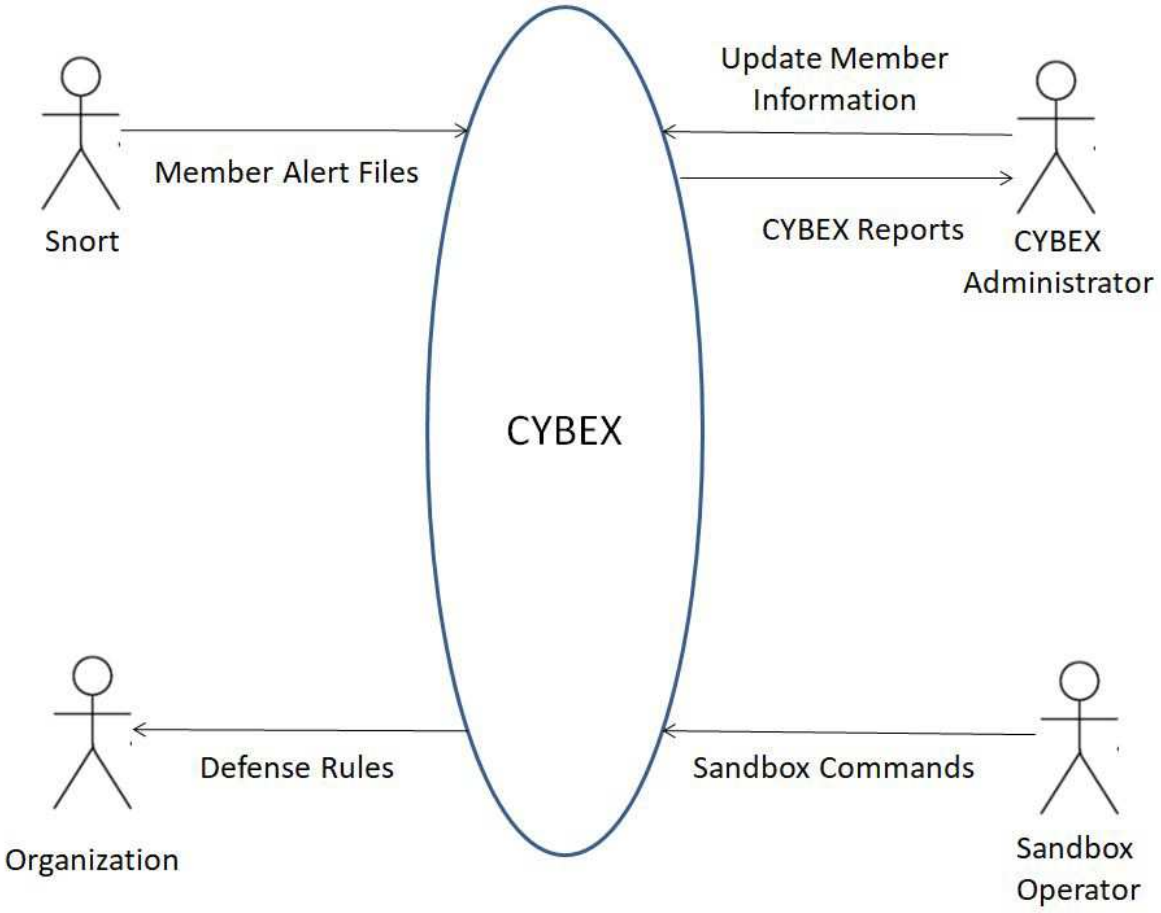

Figure 2: Context Model

Software Structure: Class Diagram

The class diagram in Figure 3 shows the list of all of the classes and their attributes. The connections between classes show both when the classes contain each other as members and when they call functions from other classes.

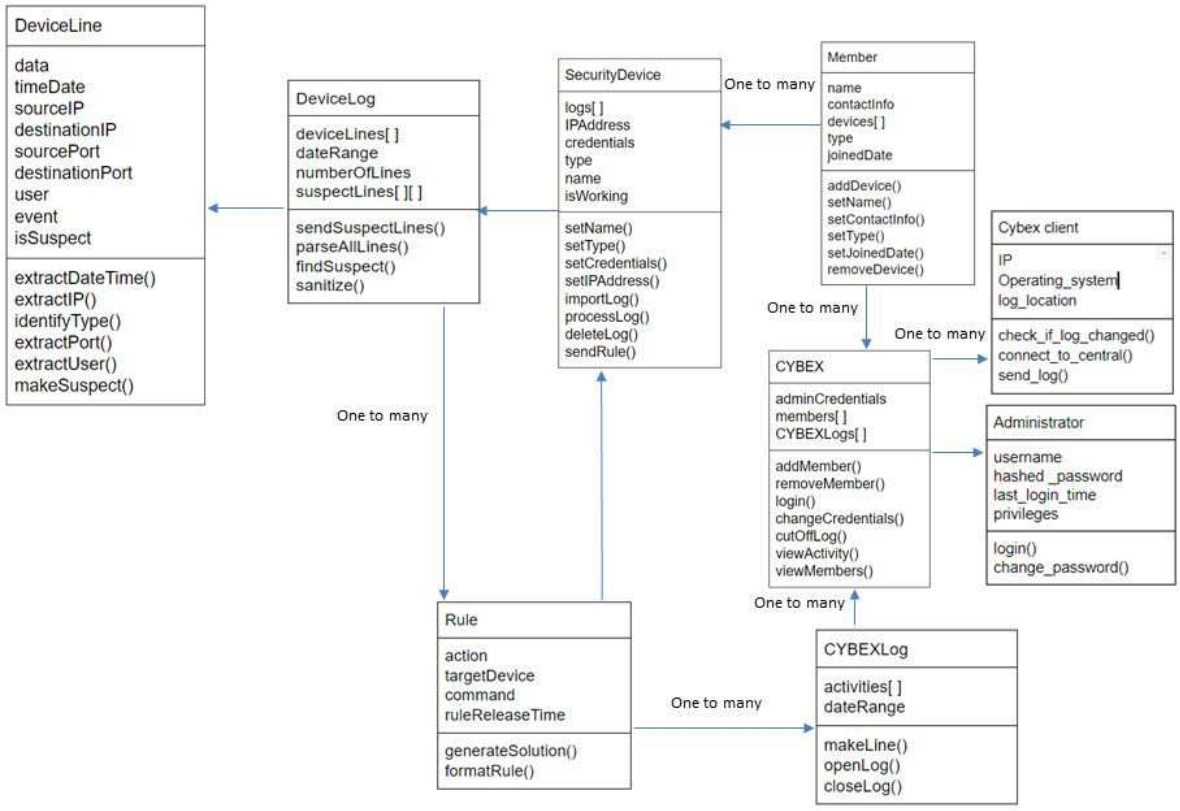

Figure 3: Class Diagram

Class Attribute and Method Descriptions

DeviceLine - this class contains either a single line or a most a couple lines pertaining to a single event in a log from a device.

Variables

data - this variable contains the line of the log file that contain the event.

timeDate - this variable holds the date and time that the event happened.

sourceIP - this variable holds the source IP address in the event, if applicable.

destinationIP - this variable holds the destination IP address in the event, if applicable.

sourcePort - this variable holds the source port number in the event, if applicable.

destinationPort - this variable holds the destination port number in the event, if applicable.

user - this variable holds the user or users involved in the event, if applicable.

event - this variable holds the type of event that happens on this line, ie user logon, connection established, etc.

isSuspect - this flag says whether a suspicious activity like an attack occurred on this line.

Methods

extractDateTime() - this method grabs the date and time information from the log file line and stores it in the date/time variable.

extractIP() - this method grabs the source and destination IP addresses and puts them in the relevant variables.

identifyType() - this method identifies what happened in the line and assigns to the event variable.

extractPort() - this method extracts the source and destination ports from the log, and puts those in the sourcePort and destinationPort variables, if applicable.

extractUser() - this method extracts any users that were involved in the even and puts them in the user variable, if applicable.

makeSuspect() - if the there is suspicious activity that is detected in that event, then this method sets the variable suspect to true.

DeviceLog - this class contains all the lines from a log file for a single device, as well as information about the log file like the date range and the size.

Variables

deviceLines[ ] - this variable holds a list of deviceLines.

dateRange - this variable specifies the date range that is covered in this log file

numberOfLines - this variable says the number of lines that are in the log file.

suspectLines[ ][ ] - this is a list that holds all the deviceLines that were considered suspicious for multiple possible attacks.

Methods

sendSuspectLines() - this method sends the suspectLines variable to the Rule class.

parseAllLines() - this method goes through all the variables in the deviceLines list and has each line run its methods to pull out the relevant information, such as the time, the source/destination IP addresses, and so on.

findSuspect() - this method looks through all of the log lines after they are parsed and adds them to the suspectLines list, if they are marked as suspicious.

sanitize() - this method removes any identifying or sensitive information from log files.

SecurityDevice - this class holds all the information about a device connected to the CYBEX network.

Variables

logs[ ] - this variable holds a list of deviceLog variables that have been generated by this device.

IPAddress - this variable contains the IP address of this device.

credentials - this variable contains the credentials for this device.

type - this variable contains what type of device this is, ie server, IDS, firewall, etc.

name - this variable contains the name of the device

isWorking - this variable indicates whether the device is up and working.

Methods

setName() - this method sets the name variable.

setType() - this method sets the type variable.

setCredentials() - this method sets the credentials variable

setIPAddress() - this method sets the IP address variable.

importLog() - this method receives a log file from the security device so that the log can be processed.

processLog() - this method processes logs that are received. It turns the log file into a deviceLog variable and splits the log file up into logLines.

deleteLog() - this method deletes a specific log in the Logs list.

sendRule() - this method sends the rule that was generated by the rule class to the security device so that it can be implemented.

Member - this class contains all the relevant info about a member organization that has joined CYBEX.

Variables

name - this variable contains the name of the member.

contactInfo - this variable contains the contact information for a member organization such as email, telephone number, etc.

devices[ ] - this variable is a list of all the securityDevice variables that a member organization has connected to the CYBEX network.

type - this variable contains what type of organization the member is, such as a bank, hospital, school, etc.

joinedDate - this variable contains the date the organization joined.

Methods

addDevice() - this method adds a securityDevice to the devices list.

setName() - this method sets the name variable.

setContactInfo() - this method sets the contactInfo variable.

setType() - this method sets the type variable.

setJoinedDate() - this method sets the joinedDate variable.

removeDevice() - this method removes a securityDevice variable from the devices list.

CYBEX - this class contains information about the CYBEX network. It manages the list of member organizations as well as a list of rules that CYBEX has pushed out.

Variables

adminCredentials - this variable contains the admin credentials for CYBEX.

members[ ] - this variable is a list of the members that are a part of the CYBEX network.

CYBEXLogs[ ] - this list contains logs from the CYBEX system, such as what rules were sent out to which members.

Methods

addMember() - this method adds a Member variable to the Members list.

removeMember() - this method removes a Member variable from the Members list.

login() - this method logs the user into the system

changeCredentials() - this method changes the adminCredentials.

cutOffLog() - this method ends the current log, add it to the CYBEXlogs list, and start a new log.

viewActivity() - this method returns the list of recently deployed rules for the system administrator to view

viewMembers() - this method displays all the current Members in the Members list.

CYBEXLog - this class holds the list of logs about the rules that the CYBEX system has generated and sent.

Variables

activities[ ] - this variable is a list that contains the logs of all the rules that the CYBEX system has sent out.

dateRange - this variable contains the range of dates that encompass all the rules sent out.

Methods

makeLine() - this method makes a new line in the activities list.

openLog() - this method opens a new log file to store CYBEX logs in.

closeLog() - this method closes and saves the currently open CYBEX log.

Rule - this class holds the rule that was generated from a suspicious part of the log file. The rule that was generated will be able to prevent that attack.

Variables

action - this variable contains a human readable version of what the command is for the security device that this rule applies to.

targetDevice - this variable contains information for the device that this rule applies to.

command - this variable contains the command that the device needs to run in order to implement the rule.

ruleReleaseTime - this variable contains the time that the rule was sent out to the device.

Methods

generateSolution() - this method takes in the deviceLine variables that were marked as containing suspicious activities, determines the type of attack that took place, and assigns a rule to the command and action variables that could have prevented the attack.

formatRule() - this method converts the generated solution from an internal format to the required format for the given security device.

Administrator - this class holds the data for the administrator of CYBEX

Variables

username - this variable contains the username for the administrator

hashed_password - this variable contains the password for the administrator, stored as a hash

Last_login_time - this variable contains the last time the administrator logged in

privileges - what degree of access the administrator has

Methods

login() - This method checks the supplied password of an administrator logging in and grants access to the system if successful

change_password() - This program updates the hashed_password of an administrator with a new password.

Cybex_Client - this class is in the client program that runs on devices in the CYBEX network.

Variables

IP - This contains the IP address of the client machine producing logs.

operating_system - This variable contains the operating system information of the client for use in sending rule updates

log_location - this variable holds the file path for the location of the log file being monitored.

Methods

check_if_log_changed() - this method checks to see if the log file has changed at all, indicating an attack.

connect_to_central() - this method makes a network connection to the main CYBEX server

send_log() - this method sends the specified log to the main CYBEX machine for processing.

Database: Data Structures

This program has an implementation of a MySQL database for storing large amounts of information about member organization devices.

Rules	
date	Date and time that the rule was created.
rule	The rule generated for users.

contactInfo	
name	The name of the contact person.
email	The email of the contact person.
phone	The phone number of the contact person.

adminCredentials	
username	The username of the admin.
password	The password of the admin.

Virtual Hardware Design and Integration

The CYBEX system does not require any hardware design for the software itself. As it is an information sharing system, it requires an internet connection but no actual hardware. CYBEX also leverages other software programs such as Snort and IPTables.

For testing and demonstration, CYBEX runs in a virtualized sandbox environment. The environment is a set of networked computers running on Oracle Virtualbox in host-only mode, allowing the virtual machines to communicate within the sandbox but without having access to the internet. There will be three types of computers running in the sandbox, the attackers, the member computers, and the central CYBEX server.

The attacking computers run Kali Linux as the operating system and use Nmap for network scanning, Hydra for launching a brute-force attack against an ssh server, and Metasploit for other exploits.. Their IP's will change over time and they will be connected to some or all of the member computers.

The member computers run various types of Linux, such as Ubuntu, Kali, or Metasploitable. These computers also run Snort and IPTables as well as other services such as Apache for a web server and an SSH server. They will have fixed IP addresses and will be connected to the attackers and the central CYBEX computer.

The central CYBEX computer runs Ubuntu. It hosts the primary CYBEX program and is connected to all of the member computers.

Virtualization Structure: Component Diagram

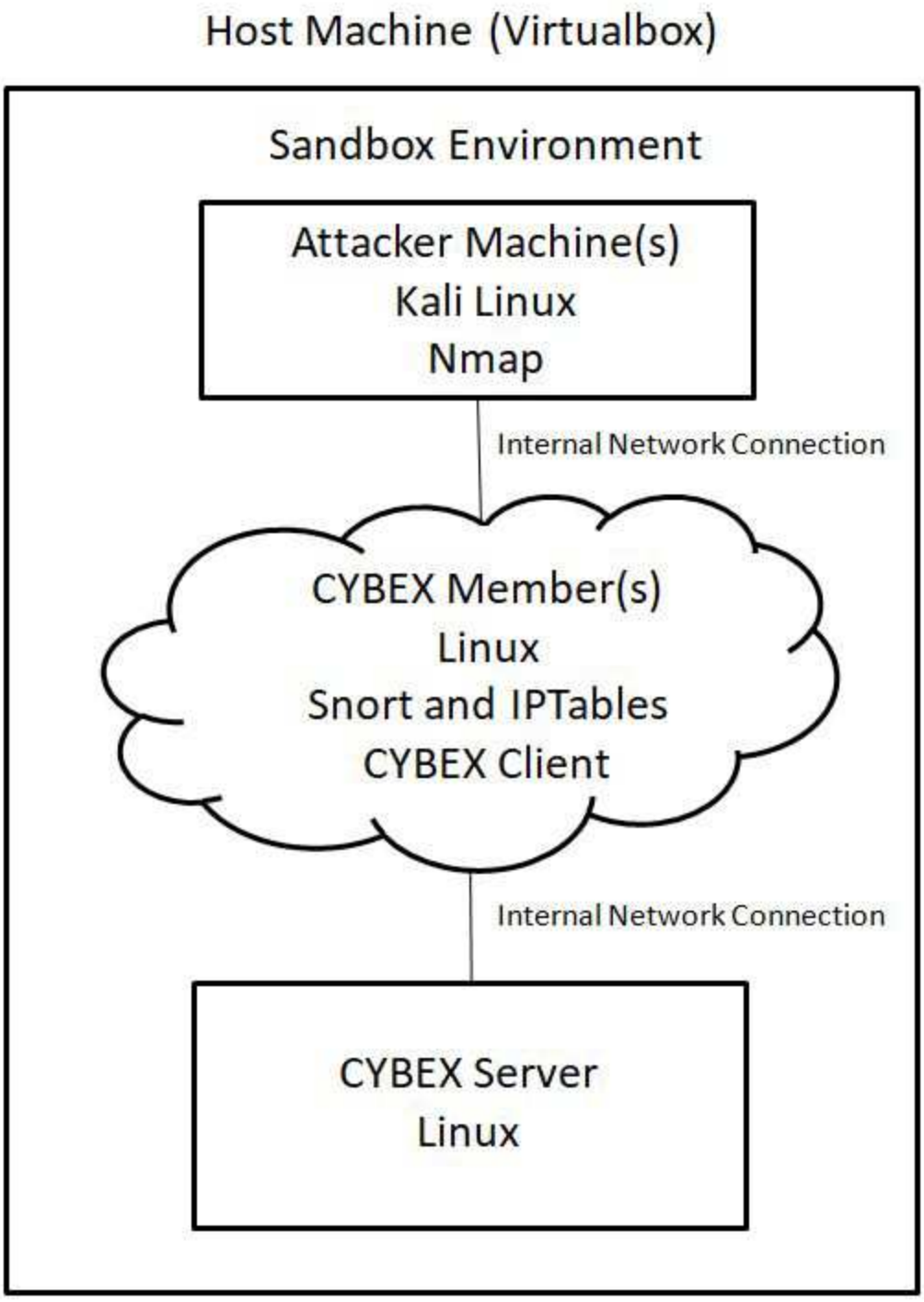

Figure 4: *Virtualization Structure: Component Diagram*

User Interface Design

Nine user interfaces for the CYBEX system administrator are shown below. These interfaces demonstrate the choices to be made in each interface, as well as the clear, efficient experience that will be created for the CYBEX system administrator. This minimalist experience is consistent with the fundamental goal of CYBEX automation: to eliminate humans from the first response component of the cybersecurity information exchange process. It should be noted that no user interface will be implemented for the sandbox operator, as this human will be directly accessing low level computer functionalities and other programs in order to develop and test the CYBEX system. This person also only exists for testing and would not be required in an actual deployed system.

CYBEX System Administrator Login

Figure 5 shows the login screen for the CYBEX system administrator. This interface provides access control for the CYBEX system. By entering and submitting a valid ID and password combination, the system administrator gains access to the CYBEX system main menu.

Figure 5: Login interface for the CYBEX system administrator provides access control.

Change System Administrator Credentials

Figure 6 shows the interface that the CYBEX system administrator will use to periodically change their login credentials. The administrator can access this screen from the main menu. This screen will automatically pop up if the system administrator has not changed credentials in a minimum acceptable time interval.

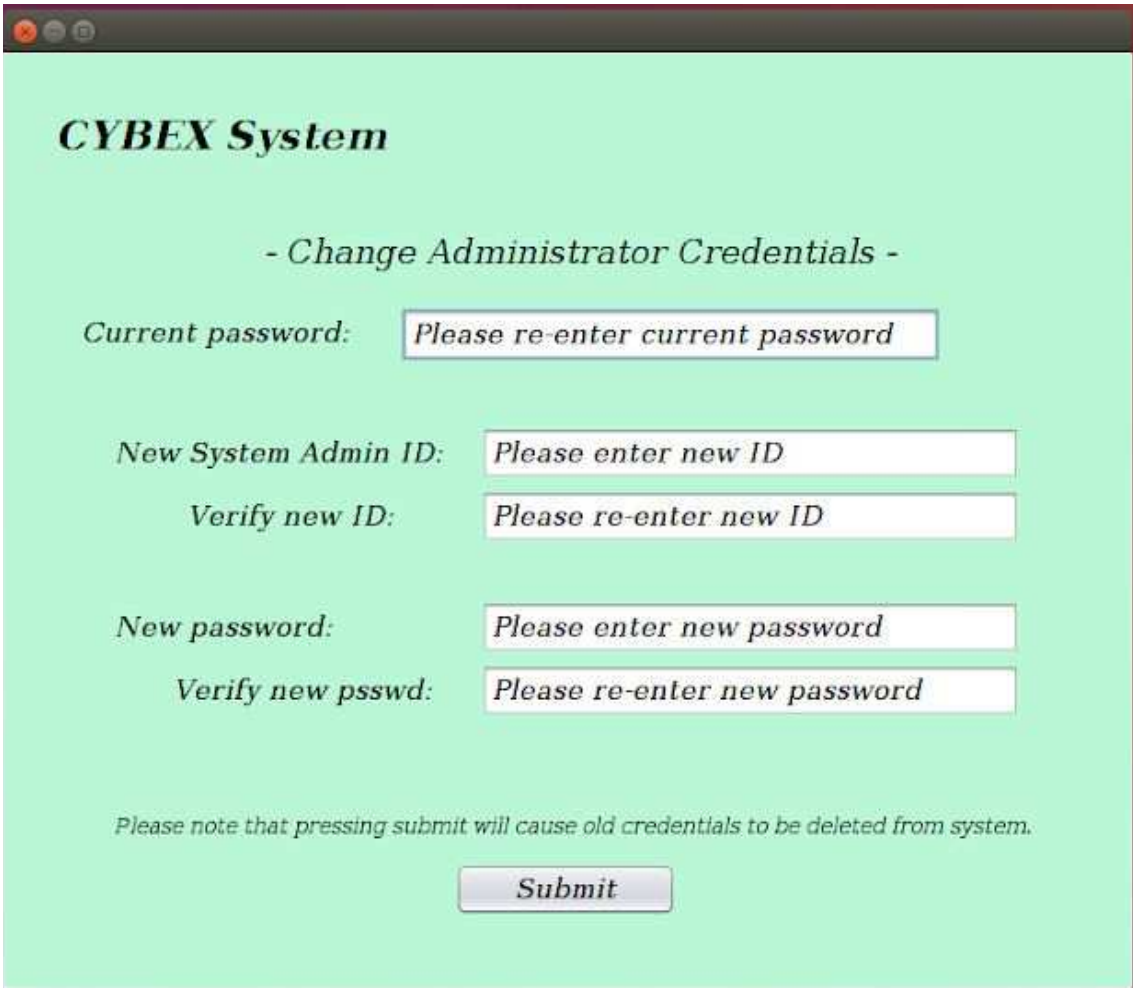

Figure 6: Interface for the CYBEX system administrator to change credentials.

CYBEX System Main Menu

Figure 7 shows the system administrator main menu which allows the system administrator to perform the minimum set of functionalities necessary to support the CYBEX system. These include viewing member organizations, adding new members, adding and deleting member devices, deleting members, viewing system activity reports, and changing system administrator login credentials. Since CYBEX is an automated system, the system administrator will not be involved in the process of acquiring logs, reviewing logs, generating solutions, or dispensing solutions. Instead, the system administrator will maintain membership functionalities and monitor system activity, in order to determine when system maintenance might be appropriate.

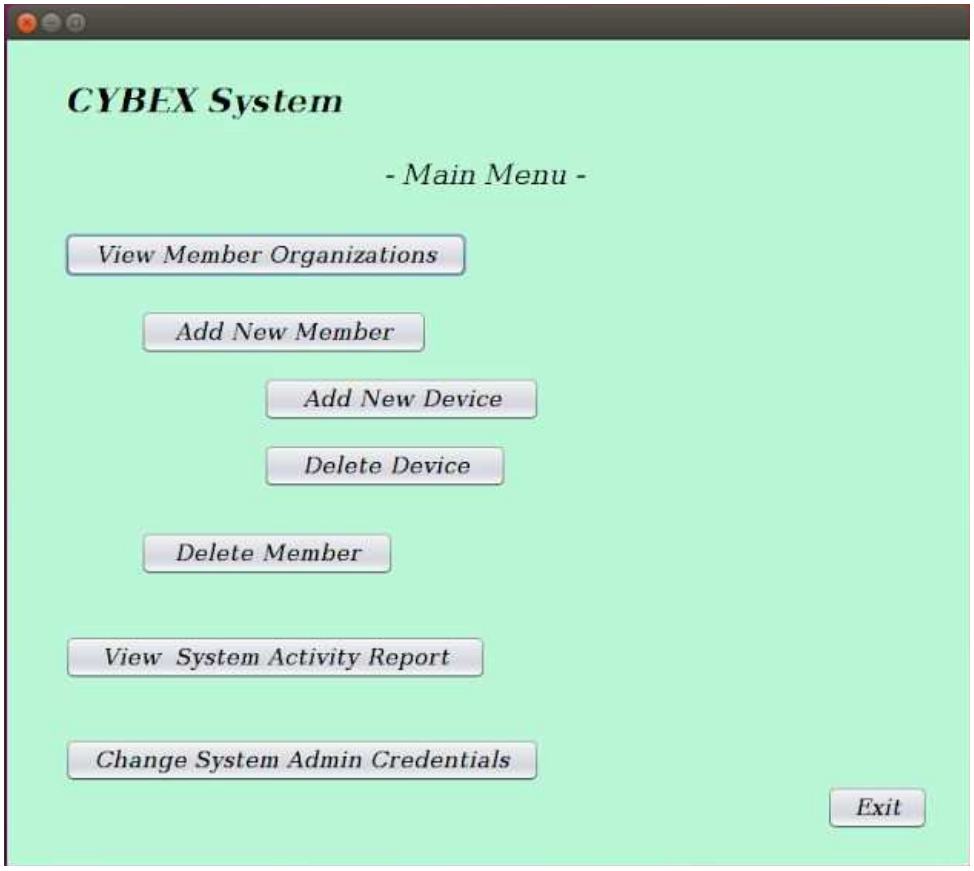

Figure 7: The main menu allows the system administrator to perform the minimum set of activities necessary to support the CYBEX System.

CYBEX System View Member Organizations

Should it become necessary to contact the designated representative of a member organization, the system administrator will acquire needed details from the interface shown in Figure 8. Example members are included to demonstrate the types of organizations that might join and benefit from CYBEX.

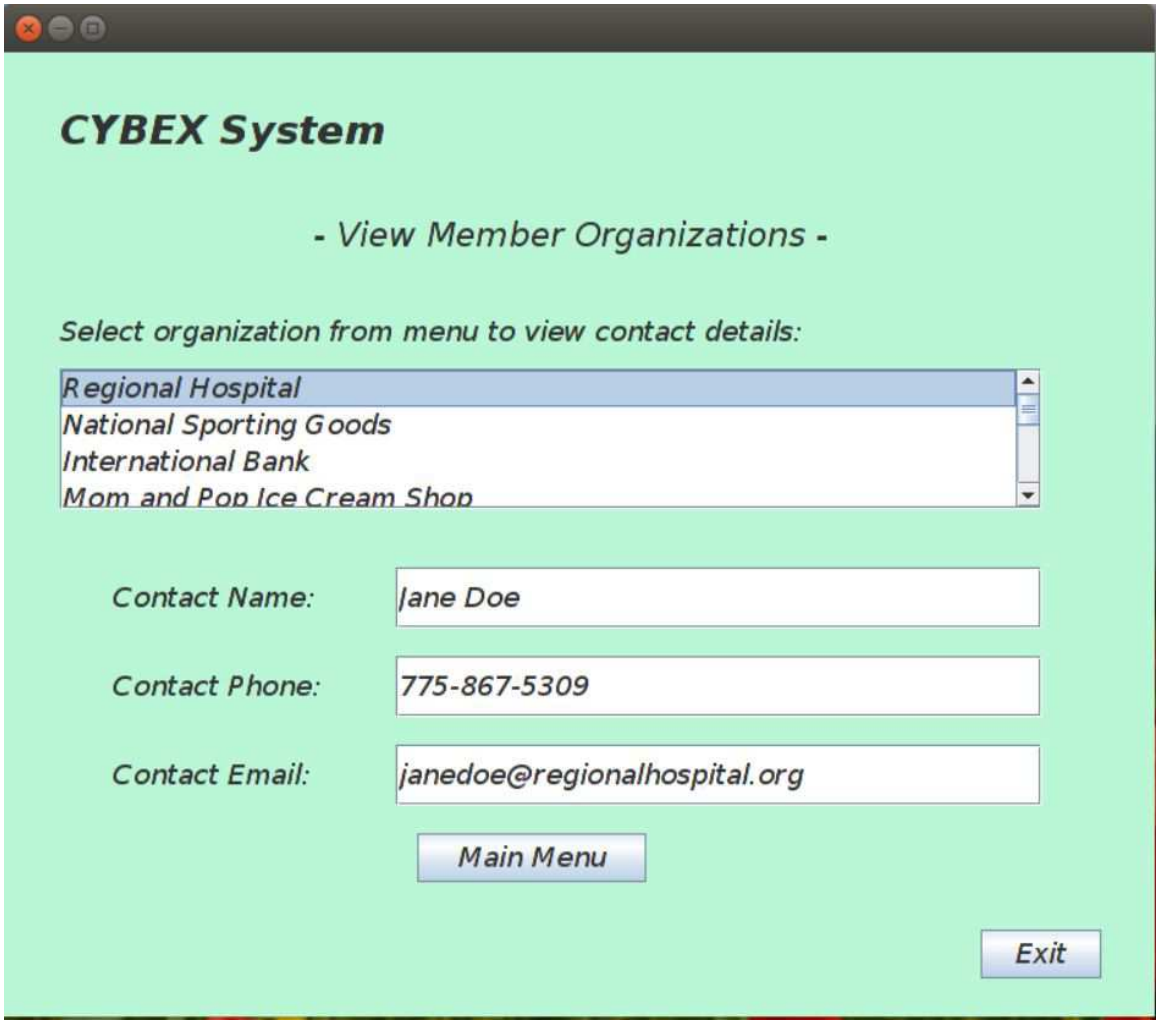

Figure 8: Interface for the system administrator to view member organization details.

CYBEX System Add New Member Organization

The system administrator adds new member organizations using the interface shown in Figure 9. Stored information is limited to details about the member organization that are necessary to effectively participate in CYBEX. These details enable contact with the designated representative of the member organization, in order to acquire details from the organization that will allow successful transfer of log files from the CYBEX Remote at the organization and delivery of solutions to the organization. These details include the specific device(s) that will communicate, as well as network address information for the source(s) of attack alert information and the destination for solutions. Device specific details are input using the add security device interface which is detailed next.

Figure 9: Interface for the system administrator to add new member organizations.

CYBEX System Add Security Device

The system administrator adds new member devices using the interface shown in Figure 10. Stored information shall be limited to details necessary to successfully transfer parsable log files from the organization and to successfully deliver solutions to the organization. Because each device will create log files with a particular data set and formatting, device definition must sufficiently specify the device(s) that will provide log files and receive solutions, as well as all necessary network connection information.

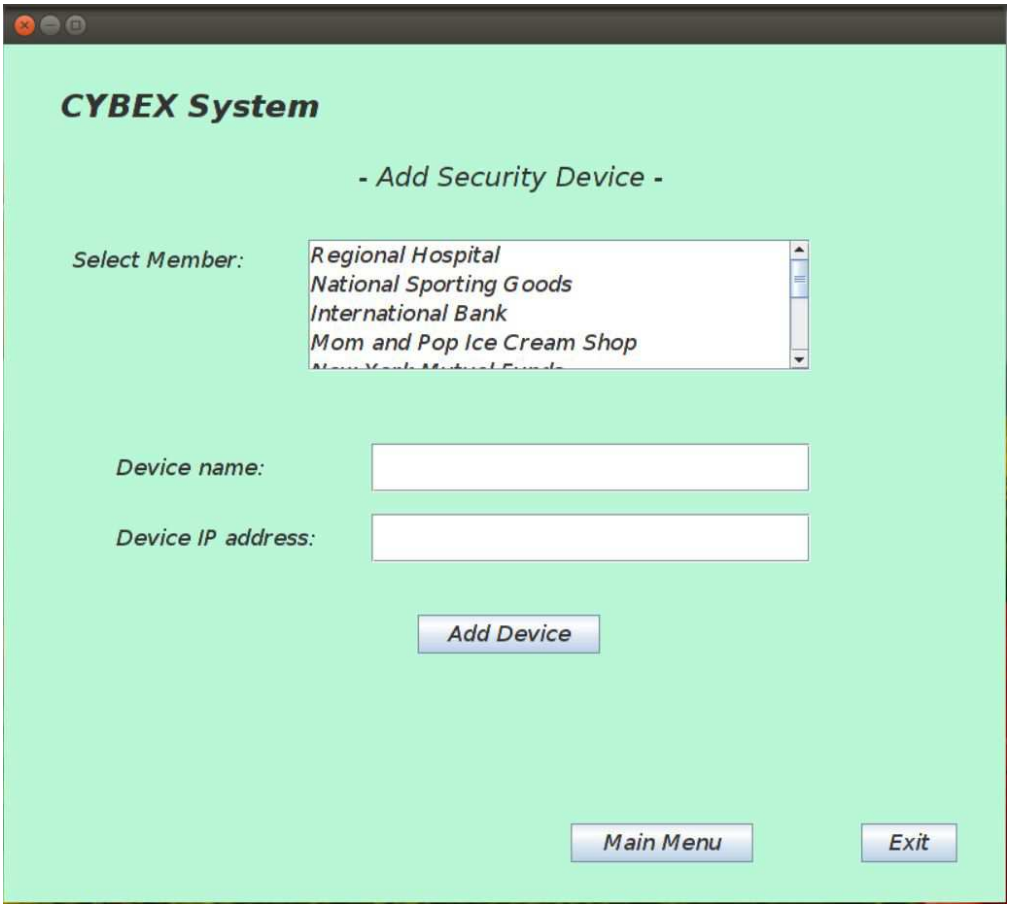

Figure 10: Interface for the system administrator to add new member security devices.

CYBEX System Remove Security Device

The system administrator removes member security devices using the interface shown in Figure 11. To minimize the possibility of accidental removals, the administrator will be asked to verify the request before it is processed.

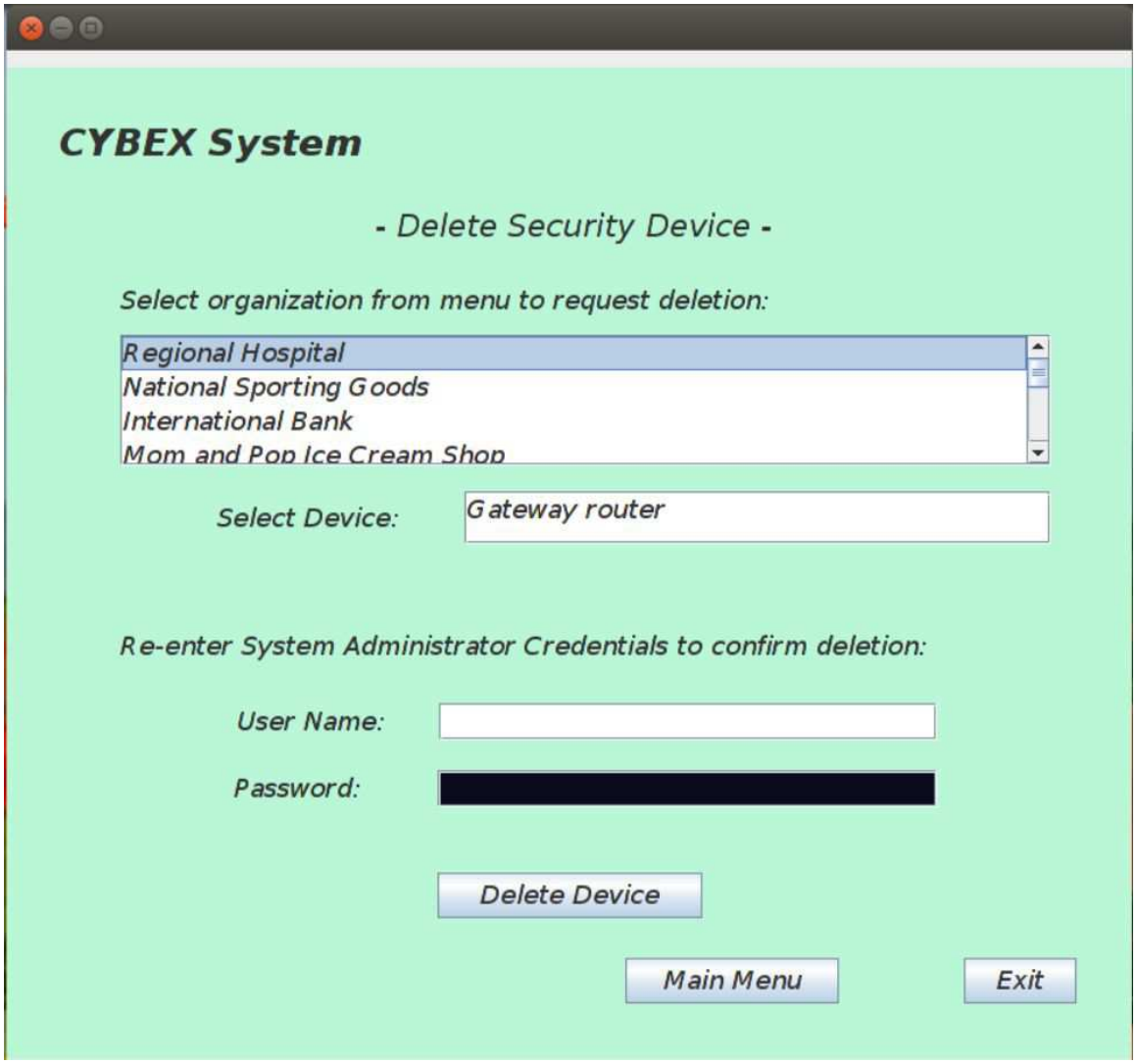

Figure 11: Interface for the system administrator to remove member security devices.

CYBEX System Remove Member

The system administrator removes member organizations using the interface shown in Figure 12. To minimize the possibility of accidental removals, the administrator will be asked to verify the request before it is processed.

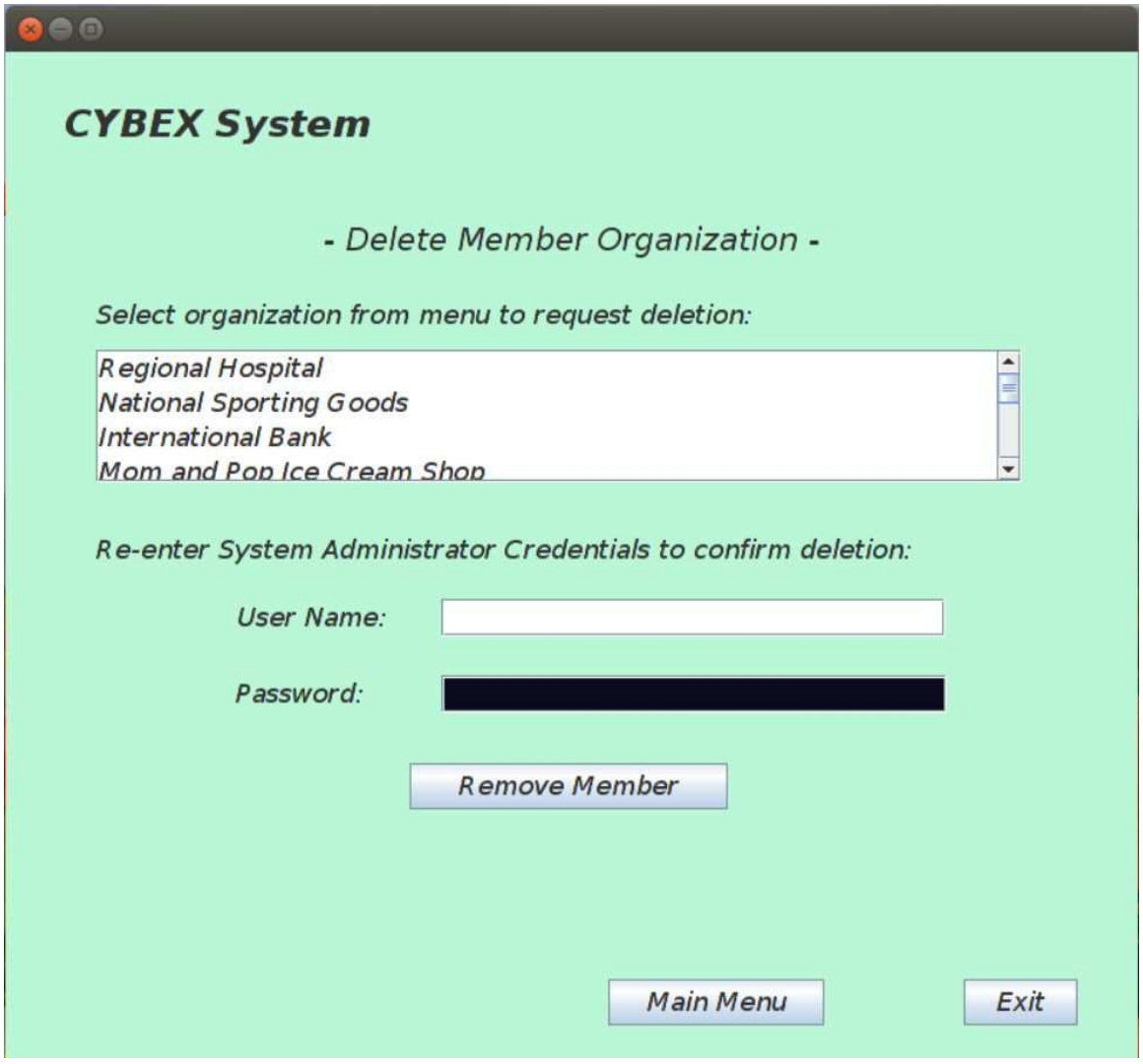

Figure 12: Interface for the system administrator to remove member organizations.

CYBEX System Activity Report Screen

The system administrator will monitor CYBEX system activity via the report shown in Figure 13. This monitoring will allow verification that the system is successfully identifying attacks, sanitizing log files, and creating and dispensing responses.

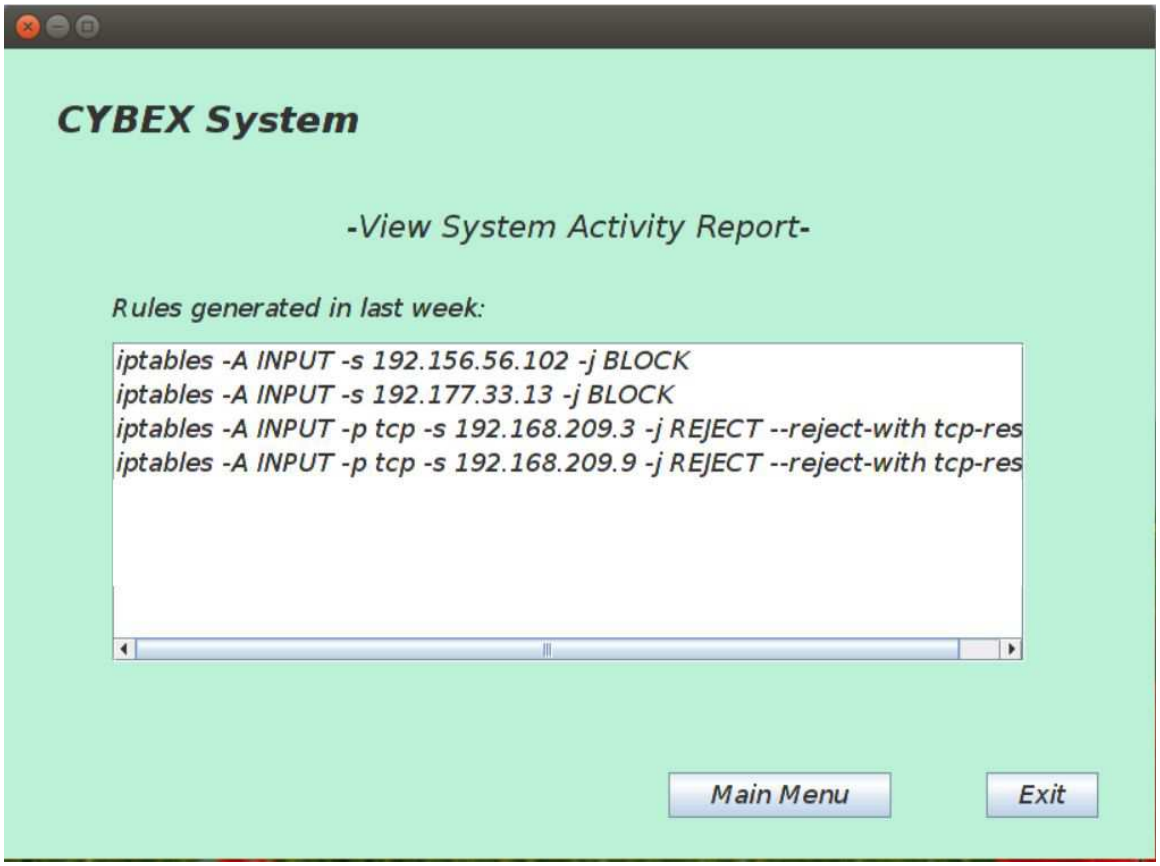

Figure 13: System administrator CYBEX system activity report screen.

Results

CYBEX was developed successfully in our testing environment. For final testing, there were 5 virtual machines configured in a host only network hosted by Oracle Virtualbox. The network configuration is shown below in Figure 14.

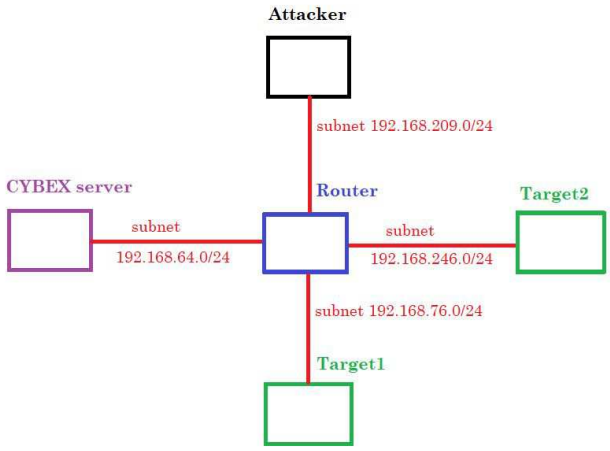

Figure 14: The configuration of the final CYBEX testing environment.

The central computer was configured to simply be a router. It forwarded traffic between each of the subnets and served no other purpose in the test.

One computer was configured to be an attacker but its IP address was regularly changed throughout and between testing, allowing it to emulate being multiple computers. This computer was running Kali Linux and was running nmap, a network scanning tool used for reconnaissance, hydra, a password brute-forcing tool, and metasploit, a vulnerability exploitation tool.

Two of the computers were configured to be members. These computers were each running Kali Linux but could have been running any other Linux operating system. They were set up running Snort, IPTables, and the CYBEX remote client. Snort was configured to detect nmap scans, SSH brute-force attempts, and metasploit exploits. These were used for testing but any other rules could have been used. This implementation of CYBEX has the potential to block legitimate traffic if Snort triggers a false positive so all other rules were disable for testing.

The final computer was set up to serve as the CYBEX server. It was running the CYBEX server program, the SQL database, and the UI. This computer ran Ubuntu as it was easier to run the UI and the SQL database.

In testing, CYBEX was able to successfully automate the defense of the two member computers. Figure 15 below shows the central server constructing a rule received after a networking mapping attack on one of the members and storing information about it to the SQL database. Figure 16 shows the rule having been applied to the member that was not originally attacked. This rule also appeared on the member that was attacked meaning that the same attack would not continue to be successful.

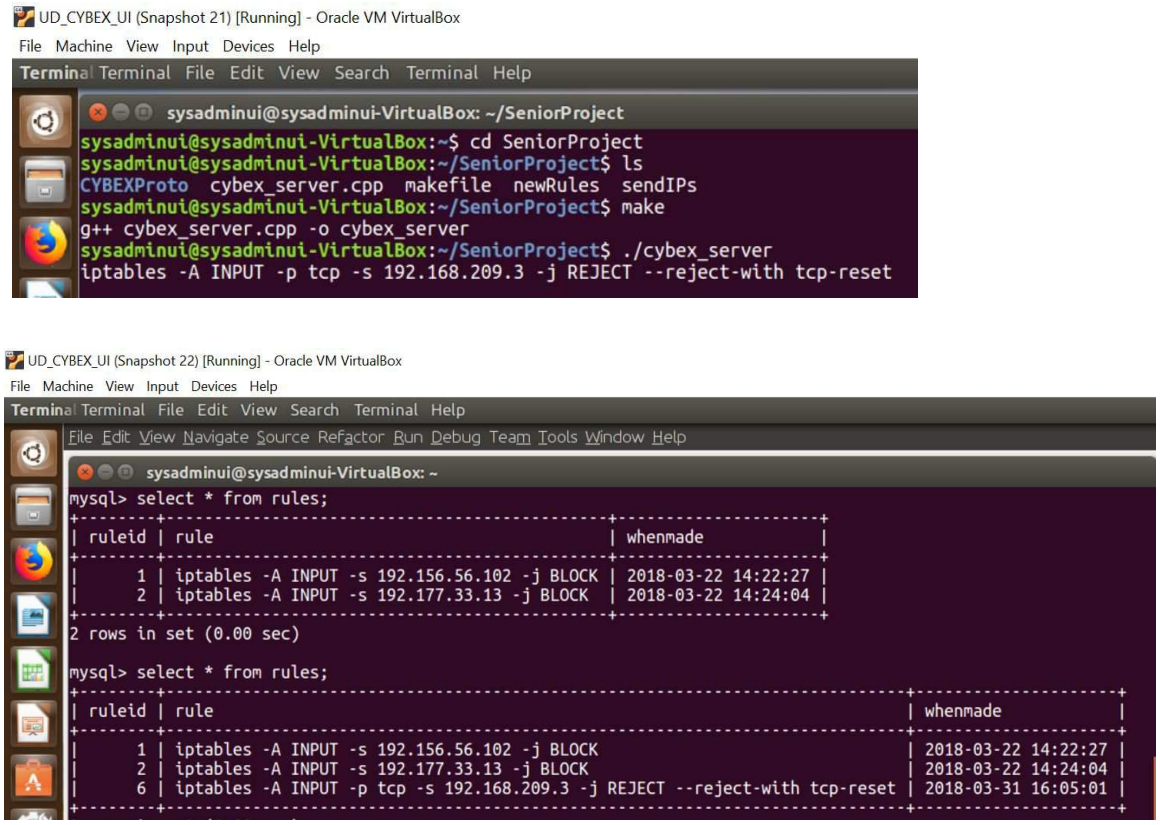

Figure 15: The server generating a rule and storing it in the database.

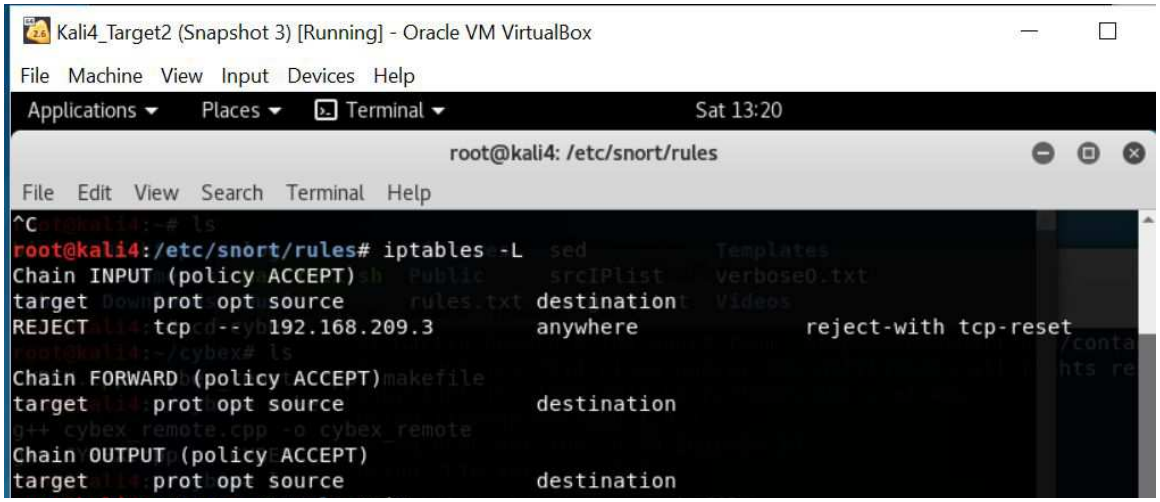

Figure 16: The rule implemented on the second (not attacked) member computer.

The time from the start of the attack to the rule being in place on all of the machines was approximately 15 seconds. This delay was mostly the time it took for Snort to detect the attack as the time from when the remote client sent information to the server to the rule being in place on all of the member computers was approximately one second.

Conclusion

These results show that automating basic security response is both practical and feasible. CYBEX is able to lower response time from minutes under a human operator to just seconds. This implementation of CYBEX is not perfect and should not be taken as a stand-alone solution. It depends on the detection and defensive capabilities of other programs and is susceptible to false positives, requiring human oversight. Having a rapid response to threats in real time will however, make entities much safer from the rapid proliferation of cyber attacks.

Glossary

Attack - An exploit on a vulnerability in a system.

Attacker - An entity initiating an attack.

Brute Force - A method of attack where a large number of passwords or login attempts are made against a system until a successful one is found.

Credentials - The required information to log into a system.

CYBEX - The name of this project. It is an automated system for generating defensive rules in response to an attack.

Cybex Log - The logs produced by CYBEX containing information about what rules have been generated and when they were sent to member organizations.

CYBEX System Administrator - The person(s) in charge of monitoring and maintaining the CYBEX system.

Device Log - A log produced by a security device that is given to CYBEX.

DOS - Denial-of-Service is an attack where the system resources of a victim are completely consumed by the attacker in order to deny service to legitimate users.

Error - The state that CYBEX will enter if something goes wrong or something unexpected happens. This initiates recovery procedures.

Event - Different classifications of actions that a log reports. These include actions such as user logins, connections established, or new rule generated

Exploit - An action taken against a vulnerability that results in improper access of information..

Firewall Attack - A process where an attacker tries to bypass firewall traffic blocking rules.

IPTables - A firewall / network control program that is part of the Linux operating system.

Mapping Attack - A process where an attacker tries to map or determine information about a victim's infrastructure in preparation for a future attack.

Member - Any entity and their associated infrastructure that may or may not be connected to CYBEX.

Sandbox - Our environment for developing, testing and demonstrating the CYBEX system. Will consist of attackers, defensive security systems, victim, and assets connected in a virtual network. It will be isolated from the outside environment.

Sandbox Operator- The person(s) operating the sandbox. Activities include system design, test and maintenance.

Sanitize - The process CYBEX will go through to remove any personal information from provided logs.

Security Device - A security device such as a firewall or an IPS that is owned by and operated by a member organization of CYBEX.

Snort - An open-source intrusion detection system.

SSH - A protocol used for accessing and using a computer remotely in a secured manner.

Suspicious Lines - Lines of a log file are suspicious if they could be part of an attack. Rules are generated from collections of suspicious lines.

Virtual Computer - A computer that is being emulated by another computer. This can be done to have multiple different and isolated systems hosted on one piece of hardware

Victim - Systems that could potentially be attacked.

References

[1] Berghel, Hal., *Equifax and the Latest Round of Identity Theft Roulette*. In IEEE Computing Edge, January 2018.

[2] de Fuentes, J.M. et al, *PRACIS: Privacy-preserving and aggregatable cybersecurity information sharing*. In Computers & Security 69, 2017, pp 127-141.

[3] International Telecommunication Union, Study Group 17, Recommendation ITU-T X.1500 [X.cybex] *Cybersecurity information exchange framework*, December 2010.

[4] MITRE Corporation, *Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX)*, https://www.mitre.org/publications/technical-papers/standardizing-cyber-threat-intelligence-information-with-the Accessed April 21, 2018.

[5] Messier, R., All In One, GSEC GIAC Security Essentials Certification Exam Guide. McGraw Hill Education, 2014.

[6] Oehmen, C. and Peterson, E., *Behavior Models to Express and Share Threat Information*, In IEEE IT Pro, September/October 2015.

[7] Vakilinia, I. et al, *Attribute Based Sharing in Cybersecurity Information Exchange Framework*, Society for Modeling & Simulation International, SummerSim-SPECTS, July 2017.

[8] Patrick Engebretson. *The Basics of Hacking and Penetration Testing : Ethical Hacking and Penetration Testing Made Easy* Ed. 2. 2nd ed. Syngress.

[9] A. Rutkowski, M. Hird, S. Adegbite, Y. Kadobayashi, I. Furey, D. Rajnovic, R. Martin, T. Takahashi, C. Schultz, G. Reid and G. Schudel, "CYBEX", *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 5, p. 59, 2010.

[10] R. Garrido-Pelaz, L. González-Manzano, and S. Pastrana, “Shall We Collaborate?,” P*roceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security* - WISCS16, 2016.

[11] McJunkin, Jeff. "Building your own home lab", *Files.sans.org*, 2017. [Online]. Available: https://files.sans.org/summit/pen_test_hackfest_2016/PDFs/Building-Your-Own-Kickass-Home-Lab-Jeff-McJunkin.pdf. [Accessed: 31- Oct- 2017].

[12] "Virtualbox Manual", *Virtualbox.org*, 2017. [Online]. Available: https://www.virtualbox.org/manual/ch01.html. [Accessed: 31- Oct- 2017].