MALT: Malicious Login Tracker

Online System for Tracking Malicious Login Attempts to the UA Little Rock Network in Real Time

8 February 2017

CPSC 5373 Software Engineering
University of Arkansas at Little Rock
Department of Computer Science

Project Team:
Zhenlin Jin (zxjin@ualr.edu)
Andrew Pyle (axpyle@ualr.edu)
Hanzhao Chen (hxchen@ualr.edu)

Mentoring:
Coşkun Bayrak, Professor of Computer Science, UA Little Rock (cbayrak@ualr.edu)
Veysel Erdag, Information Security Officer, UA Little Rock (overdag@ualr.edu)

Summary

Our proposed system will alleviate UA Little Rock Information Security Services's inefficient data-processing workflow for visualization of suspicious login data. The current workflow is required by the lack of an accessible automated data storage, processing, and visualization tool. Our solution will extract information from human-readable email alerts, save the data in a database, process the data, and visualize appropriate metrics with an online dashboard. Our proposed system supersedes the current commercial products as it integrates all the steps of the data analysis workflow, while avoiding the need to directly write code to gain understanding of the data. The project timeline includes a design phase (5 weeks), implementation phase (4 weeks), and testing phase (3 weeks). Upon project conclusion we hope to develop a scalable system that will be deployed by the school network which can be easily modified by later developers.

Table of Contents

List of Abbreviations

Google	Google Inc.
UA Little Rock	University of Arkansas at Little Rock
GUI	Graphical User Interface

List of Figures

List of Tables

Introduction

Motivation

The University of Arkansas at Little Rock network has been subject to a steady volume of uninvited, worldwide traffic. Login attempts are regularly detected from locations outside of central Arkansas, and even outside the United States. UA Little Rock's email system is hosted by Google Inc., which provides its own proprietary malicious login attempt alert system to customers. These alerts are delivered to Google's Gmail™ customers (UA Little Rock, in this case) via human-readable emails containing the account name, IP address (IPv4 or IPv6), geographical location, and date/time information.

Currently, UA Little Rock's Information Security team mines these emails using an automated script and stores the information in a Microsoft® Excel file. The Excel file is used for manual exploration of the data, for example, analysis of the location range for a given time of day, or the frequency of login attempt per account for a given date and time. This manual analysis, while helpful for security decision making, is currently a slow and inefficient process.

Significance

Monitoring connections to a computer network is an essential component of network security. If accounts exhibiting malicious patterns can be prevented from connecting to the network, the likelihood of a destructive attack or privacy breach is greatly reduced. System administrators possessing malicious login pattern can act proactively to prevent data theft, debilitating attacks, or simple breaches of privacy.

The current manual data analysis procedures yield critical information about potential malicious login attempts; however, they are time-consuming, fail to yield real-time information, and generate information only stored locally. The procedures in place currently rely on careful

attention by a human analyst to stay current, which drains the limited human resources of the Information Security Department of UA Little Rock. The result of the current network analysis methods is a tradeoff between up-to-date security information and use of human resources for the department's daily tasks, resulting in lagging analysis.

Approach

In light of the shortcomings of the current manual network traffic analysis system, we propose an automated system for treatment of the streaming network traffic information in real time. Our system will assimilate the Google-generated data from incoming email alerts into an online dashboard visualizing threat classification, origin of potential malicious login attempt, and descriptive metrics for accounts, locations, and times of day, etc. Alerts will be generated for critical security-related findings. This product will provide instant access to actionable security knowledge, and provide the information necessary for decision-making.

The proposed system is superior to the current manual model due to its automated workflow, constantly updated data source, online accessibility, and real-time network security analysis. The implementation of such a system would free human resources and result in more complete information for security decision-making.

The limitations of the proposed network traffic analysis system include reliance on static threat classification algorithms and UA Little Rock's server hosting infrastructure. First, the system's information visualization interface will rely on data processing methods predefined by UA Little Rock's Information Security department. The automated nature of the system will preclude the level of customization attainable with manual analysis. Second, for the system to both run in real time with constant accessibility and maintain data privacy for account data, it will need to be hosted by UA Little Rock in a secure server environment. Thus the system will require dedicated resources for the its implementation.

Literature Review

Many data visualization tools exist today. The operation of a successful internet enterprise depends on appropriate analysis of huge amounts of data. UA Little Rock is no exception. Specifically, the Information Security Services department has a critical responsibility to maintain the security of the university's internet system. Acquisition and understanding of login data information is central to the task of maintaining network security, and data visualization supplies an excellent cognitive aid in understanding data.

There are two major commercial data visualization approaches for which tools exist currently. Tools exist which import data from a static source and require user operation to construct a dashboard. These tools typically utilize a GUI, drag-and-drop editing, and don't require coding. Examples in this first GUI category include statistical software packages (SPSS [1], minitab [2]) and data visualization software (Tableau [3], Cognos [4], Microsoft® PowerBI [5]). The second type of data visualization tools do not utilize a GUI, and are usually available as language libraries or languages themselves. These tools are extremely versatile due to their basis in coding. Examples of such software include SAS [6], Plotly [7], D3.js [8], and many others.

Both of the above approaches to data visualization are insufficient for our purposes on their own. The GUI approach is currently in use (PowerBI), and as reported by our client, requires too much manual input to achieve the desired knowledge through visualization. This leaves the second approach, the code approach. However, to use one of these software packages alone to create visualizations on the fly from streaming data, significant time must be invested in learning the language. Each of these languages has a large learning curve.

This means that a single, integrated data processing system must be created, taking advantage of the flexibility of the code-based visualization approaches by using supporting software to simplify the required workflow and dataflow.

Our system will achieve many benefits for the client over existing systems. First, it will largely reduce the manual operation, since it saves Gmail™ supplied data into a database directly, and generate reports from our database. It saves a large amount of human resource, and avoids many possible mistakes. Additionally, since our aim is analyzing specific type of data, the user interface that we are going to build can be more condense and concise, comparing to other statistic visualization tools mentioned above. This feature can significantly reduce user's learning cost.

Beyond achieving the visualization functions that already existed in Microsoft Power BI, we are going to build an map that supplies the geographical origins of malicious behaviors. The user (admin of network system) can have a clearer idea about what operation need to be done after they see the report and reviewing the login attempts that are displayed on a map.

Research Design and Methodology

High-Level Requirements

Primarily, the system must be allow Mr. Erdag and the rest of UA Little Rock Information Technology Services to monitor suspicious login attempts on the network.

The system will extract suspicious login attempt information provided by Gmail™ and visualize all the malicious login attempts, providing easy understanding of the data and its trends over time to the client. The system should be reliable over time and scalable, providing the foundation for other developers to easily add functionality, include more diverse data sources, or modify the source code.

Logical Flowchart

The following flowchart represents the logical process of our system. There are four major stages in the process of the system. Starting with the original data in the form of human-readable emails, we will extract the relevant data (see Data Analysis below) into a database environment, request the data for processing with Python [9], and output the results of the analysis to an online data visualization dashboard using the Django Python web framework [10]. See Tools to be Used section below for more detail and alternative system design possibilities.

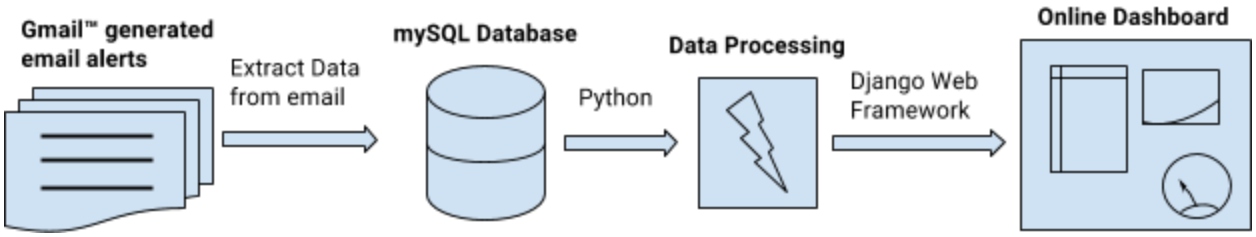

Figure 1. Logical flowchart of the software system.

Tools to be Used

Our system will leverage several open-source technologies. We plan to accomplish both the initial data-extraction and core data-processing using the Python programming language, due to its usefulness with data-manipulation and database interface. mySQL™ Community database-management system [11] will be used to store and serve data for analysis. Plotly open-source Python libraries will provide the essential data visualization functionality directly to Python. Hosting of the Python backend into an online environment will be done with the open-source Django Python web framework.

In the event that Plotly for Python on the Django framework is not flexible enough to create an effective solution, the open-source BootStrap JavaScript web framework [12] combined with jQuery or other libraries will serve as the front-end platform with backend connection to the mySQL™ database.

The development environment will consist of several open source tools as well. PyCharm Python IDE [13] will be used for development and testing of the data processing components, while Brackets text editor [14] will serve as the front-end html and CSS development environment, if needed.

System Environment

The system will be composed of a backend data extraction and storage solution, as well as a frontend data-processing and visualization solution. These two solutions will be hosted by UA Little Rock's private server environment, allowing secure access to the network traffic information.

All the record from G-mail will be sented to our server and saved into our mySQL™ database. Our application will be running in an server offered by UALR IT service, which usually runs a linux operating system.

Data Analysis

The proposed system for analysis of network traffic is based on four data points provided by Gmail™: account name, IP address (IPv4 or IPv6), geographic location, and time/date. See the below table for sample data. These four data points will inform the proposed data analysis.

Account Name	IP Address	Location	Date and Time
xxxxxxxxxxx	108.64.45.187	Benton, Arkansas, United States	Sunday, January 8, 2017 at 7:46:32 AM Central Standard Time
xxxxxxxxxxx	24.144.24.103	Conway, Arkansas, United States	Sunday, January 8, 2017 at 2:08:42 PM Central Standard Time
xxxxxxxxxxx	166.173.62.123	Oklahoma City, Oklahoma, United States	Sunday, January 8, 2017 at 3:38:24 PM Central Standard Time
xxxxxxxxxxx	73.133.196.202	North Bethesda, Maryland, United States	Sunday, January 8, 2017 at 8:08:30 PM Central Standard Time
xxxxxxxxxxx	73.133.196.202	North Bethesda, Maryland, United States	Sunday, January 8, 2017 at 8:08:31 PM Central Standard Time
xxxxxxxxxxx	73.133.196.202	North Bethesda, Maryland, United States	Sunday, January 8, 2017 at 8:08:30 PM Central Standard Time
xxxxxxxxxxx	2601:381:8101:1b0f:d066:4f15:899b:dc49	Little Rock, Arkansas, United States	Sunday, January 8, 2017 at 8:52:47 PM Central Standard Time

Table 1. Sample network traffic data.

The proposed system will use the above four data points for analysis. Account names have been redacted to preserve data anonymity.

Essential results of the proposed system's data analysis include the following five tasks. The first four tasks will be combined to create an aggregate threat measure with one of three values: mild, severe, or no threat.

1. Mapping the geographical origin of login attempts
2. Visualizing the most frequent potentially malicious account and location
3. Logging entrance of a new account into the database
4. Logging presence of accounts in the database which do not exist as active UA Little Rock accounts
5. Development of an aggregate threat measure for each login attempt event

The analysis will be accomplished by accessing data stored in a local database and processing the records appropriately. The system will display up-to-the-minute streaming data as it is saved to the database. We intend to provide a view of data for a limited historical timeframe, as decided by the client. Time-permitting, we will integrate interactivity, allowing a user to specify their own timeframe for investigation of login attempts from the database.

Validation/Testing

The goal of our testing is to discover as many faults as possible such that they can be repaired before the delivery of the system, which is to be deployed by the campus network administrator and modified by software developer in the future.

In the case of this project, we are dividing the testing into three major parts: unit testing , integration testing and system testing. During the unit testing, we we will be using the sample input data sets, the historical data to passed to the front-end and back-end systems, respectively. During the integration testing, combinations of front-end and back-end systems are integrated together and compared. During the system testing, typical and exception cases are run through the system and compared with the requirements model. The requirements model are proposed by Mr Erdag assessed by Professor Bayrak.

Specifically in the stage of system testing, we will run functional testing, performance testing, acceptance testing, and installation testing.

- Functional testing tests the requirements from RAD and the user manual.
- Performance testing checks the non-functional requirements and additional design goals from Mr Erdag and Professor Bayrak.
- Acceptance testing and installation testing check the system with the project agreement. We will have Mr Erdag to assess the system and have the approval from the developers and network administrators.

Ultimately, there could be non-optimal behaviors, as mentioned by Mr. Erdag during the meeting. We will make an effort to eliminate them through testing and make them as few as possible.

Timeline

Our timeline is divided into three phases: design, implementation, and testing. Each phase includes activities and tasks as detailed below. The timeline is given in tabular and Gantt chart form as well.

Design Phase | Five Weeks

The system design, including the data processing model, visualization techniques, web framework, text scraping program, and database design will be researched and designed.

- Data Processing Model
    - Research streaming data analysis
    - Design appropriate analysis methods
    - Research tools for data analysis implementation and visualization
        - Python Web application using Plotly and Django
        - Javascript visualization library (D3.js, etc.)
        - Other options
- Web Framework Environment

    - Research web frameworks for chosen data analysis and visualization tools
    - Research web hosting environment on UA Little Rock's virtual server system
- Database Design
    - Design a relational database that store all records
    - Design an mySQL™ algorithm to extract specific types of records
- Text Scraping
    - Research appropriate text processing tool for email scraping

Implementation Phase | Four Weeks
All systems will be coded and prototyped for review by Mr. Erdag in an iterative process. All systems will be operational by the end of this phase.

- Data Processing Model will be coded in chosen tool
- Web framework will be created and deployed in a UA Little Rock virtual server environment
- Visualization dashboard will be created using the chosen tool
- All systems will be submitted to Mr. Erdag for feedback

Testing Phase | Three Weeks
The testing phase and Implementation phase will not be discrete phases, as the iterative prototyping process will gradually shift from a feature implementation focus to a testing focus. Historical data will be initially used to test both the front-end and back-end systems, followed by testing with live data. All systems will be tested individually and in conjunction with other systems. Furthermore, the system has will be easily adopted and modified by the school network administrator.

Deliverables

All of the following deliverables will be provided to the client by the conclusion of the testing phase, and delivery of the final deliverable will mark the final conclusion to the project.

1. Software source code
2. User manual detailing
    a. Data processing model
    b. API for linking data sources to the web service
    c. Database design
3. Demonstration of prototype visualization dashboards and feedback
4. Demonstration of final, real-time online visualization

Task Mode	Task Name	Duration	Start	Finish	Predecessors
	CPSC 5373 Project	**59 days**	**Mon 2/6/17**	**Thu 4/27/17**	
	Design Phase	**5 wks**	**Mon 2/6/17**	**Fri 3/10/17**	
	Data Processing Model	80 hrs	Mon 2/6/17	Fri 2/17/17	
	Web Framework Analysis	10 days	Mon 2/6/17	Fri 2/17/17	
	Database Design	16 days	Fri 2/17/17	Fri 3/10/17	
	Text Scraping	16 days	Fri 2/17/17	Fri 3/10/17	
	Implementation Phase	**4 wks**	**Fri 3/10/17**	**Thu 4/6/17**	
	Database Coding	15 days	Fri 3/10/17	Thu 3/30/17	
	Creating Web Framework	15 days?	Fri 3/10/17	Thu 3/30/17	
	Creating Visualization Dashboard	5 days?	Thu 3/30/17	Wed 4/5/17	
	Submission of System	1 day	Thu 4/6/17	Thu 4/6/17	9
	Testing	**3 wks**	**Fri 4/7/17**	**Thu 4/27/17**	
	Unit Testing	6 days?	Fri 4/7/17	Fri 4/14/17	
	Integration Testing	5 days?	Mon 4/17/17	Fri 4/21/17	12
	System Testing	4 days?	Mon 4/24/17	Thu 4/27/17	13

Figure 2. Project timeline broken down by subtask.

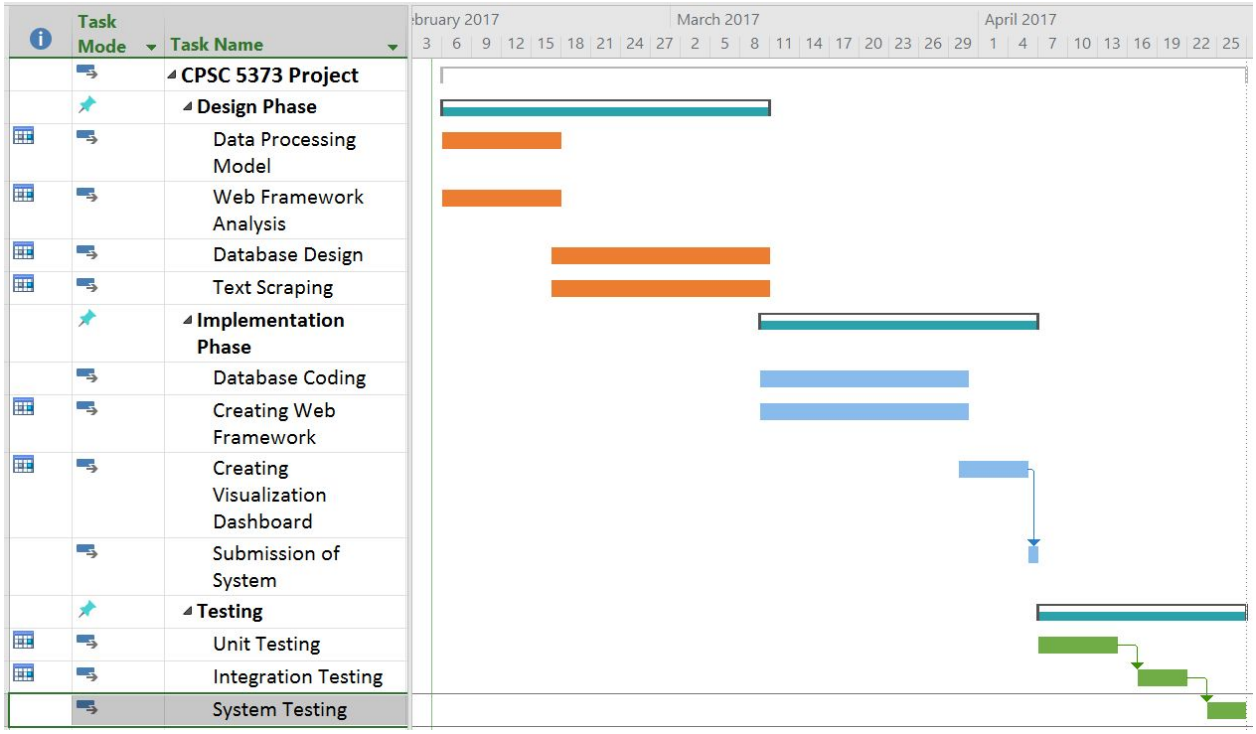

Figure 3. Gantt chart of project timeline.
Each entry corresponds to the timeline in figure 2 above.

Bibliography

[1] IBM SPSS Statistics. IBM <http://www.ibm.com/analytics/us/en/technology/spss>

[2] Minitab 17. Minitab, Inc. <http://www.minitab.com/en-us/products/minitab/>

[3] Tableau Desktop. Tableau Software <https://www.tableau.com/products/desktop>

[4] Cognos Analytics. IBM <http://www-03.ibm.com/software/products/en/cognos-analytics>

[5] Microsoft PowerBI. Microsoft® <https://powerbi.microsoft.com/en-us/>

[6] SAS 9.3. SAS Institute Inc. <http://www.sas.com/en_us/software/sas9.html>

[7] Plotly API Libraries <https://plot.ly/python>

[8] D3.js. Open Source <https://d3js.org>

[9] Python Software Foundation <https://www.python.org>

[10] Django Software Foundation <https://www.djangoproject.com>

[11] mySQL™. Oracle Corporation <https://dev.mysql.com>

[12] Bootstrap <http://getbootstrap.com>

[13] PyCharm IDE © JetBrains s.r.o. <https://www.jetbrains.com/pycharm>

[14] Brackets © 2012 Adobe Systems Incorporated. All rights reserved. <http://brackets.io>