Graphical User Interface for Anomaly Detection in DBOS

by

Robert Redmond

B.S. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2021

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

© Massachusetts Institute of Technology 2022. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by .....
Michael Stonebraker
Adjunct Professor
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Graphical User Interface for Anomaly Detection in DBOS

by

Robert Redmond

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes a graphical user interface which can aid in managing and visualizing detected anomalies within Database Operating System (DBOS). Because web applications can be built atop DBOS, it needs a security system to counteract incoming attacks from online. One of the cornerstones of a full security pipeline is a strong interface so system experts can monitor and react to incoming threats. While command line interfaces and graphical user interfaces are both means to monitor incoming/potential threats, the latter confer stronger advantages within the context of anomaly detection in DBOS. User studies were used to evaluate the interface's features throughout the development process.

Thesis Supervisor: Michael Stonebraker
Title: Adjunct Professor

Acknowledgments

I'd like to thank Professor Michael Stonebraker for his guidance over the duration of this project. Çağatay Demiralp and Deeptaanshu Kumar provided feedback and advice throughout this projects development and were invaluable to its success. Additionally, I'd like to thank the greater DBOS team, with special thanks to Qian Li and Peter Kraft whose previous work on DBOS and continued support made the project possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the internet has matured, it has taken a crucial role in all aspects of modern society. Whether it be in connecting communication world wide or serving as the basis for financial systems the world over, never before has the technology we use been more crucial. Because of the importance of online systems, they are often attacked by malicious actors for their own gain. These actors might leak users' confidential data or compromise users' accounts and steal their data, which takes an obvious toll on these users. As such, it is vital that these systems be protected from attacks.

Any online system today needs to be able to recognize and stop incoming malicious attacks. However, with novel attack methods being found regularly this requires constant vigilance on the part of system administrators to ensure that no malicious actors gain access. Security systems can quickly become outdated as attack methods evolve to overcome them. In addition to protecting against all the possible known threats, a security system also needs to allow the administrator to quickly recognize when a new threat has emerged.

Stopping known attacks from impacting a system requires recognizing what can uniquely identify an attack and rejecting any interactions which match the pattern. However, it can be difficult to extract exactly what identifies an attack. Compounding the issue, attackers actively try to disguise their traffic to appear benign to unsuspecting systems. Additionally, identifying novel attacks is very difficult; often it is not until a system has been breached that novel attack methods become known. While

new methods relying on machine learning are becoming more common, most systems still rely on security experts to identify and investigate potential attacks. It requires knowledge and skill to distinguish a potential attack from normal traffic. As such, security systems need a suite of tools designed to help experts sift through incoming traffic in the hopes of identifying possible attacks.

DBOS [1], is a system under development which can serve as the backend to web applications. DBOS is a novel design which places the database as the fundamental unit of an operating system instead of the file system. In order to serve as the backend for online systems, it will need the same industry standard security features to protect its users online.

This thesis details the design of a graphical user interface for a DBOS anomaly detection system. The goal is to allow security experts to quickly identify potential attacks in their system and conduct rapid investigation of these attacks.

Chapter 2

Related Work

2.1 Database Operating System

DBOS is an ongoing effort to develop an OS stack which rests on a modern database management system (DBMS) [1]. Whereas traditional OSs rely on the file system, and can then interact with a database, DBOS bypasses the file system entirely. The novel architecture of DBOS tightly integrates computation and data, which allows it to be particularly performant on data-centric tasks.

A crucial component of DBOS's system design is its robust provenance system [2]. All of DBOS's current state is stored in a database, and DBOS additionally allows for logging of all user requests. This could allow users of DBOS to go back to any previous state, or track exactly how some piece of data arrived at its current state.

Because DBOS is built atop a database, it is critical that its data is impervious to attacks that would illegally access, modify, and/or delete it. Luckily, the provenance system supported by DBOS allows users to log all changes with relative ease.

2.1.1 Nectar Network

Nectar Network is a web application for which DBOS serves as the backend. It is a fairly simple application, serving largely as a test bed for development, which allows users to register, login and send or receive messages between each other. This

served as the web application for which threat data is displayed by the new graphical user interface developed. In order to generate data, the website was made public at `http://nectarnetwork.org/`. In addition to the random attacks generated by internet traffic finding the site, specific attacks against the site were generated using attack tools. All of these were logged through DBOS's provenance system.

2.2 Modern Security Threats

Security threats may be constantly evolving, but we know what types of attacks are the most common. In an effort to make application developer aware of these attacks so that they can combat them, The Open Web Application Security Project (OWASP) lists 10 of the most common vulnerabilities in web applications [3]. We'll consider the first 3 examples to understand the trends they paint in web security. All of these rely on abusing vulnerabilities in an application, and will differ significantly from normal traffic in their inputs to the application.

2.2.1 Broken Access Control

This broad category includes any way in which an attacker might abuse an applications login access control to gain illicit access. As an example, an attacker might try to directly navigate to a website's admin page. An application with broken access control might not verify that the attacker has admin privileges, and thereby grant access.

2.2.2 Cryptographic Failures

This category encompasses any vulnerabilities which may lead to encrypted data stored in a database to being exposed by an attack. Any attack which can access an application's database can constitute a cryptographic failure, but only if a web application has failed to properly protect users' data. Common causes cited include not encrypting all sensitive data or failing to keep the encryption keys properly secured.

2.2.3 Injection

This category includes various types of code injection, including SQL injection and cross site scripting. SQL injection attacks operate by including SQL strings in queries which attempt to pull confidential information from an application's database. This might give an attacker read and write access over the entirety of the database. Cross site injection also involves injecting an attacker code into a website. Instead of targeting the database, this method targets other users who browse the webpage, often to steal confidential information.

2.3 Modern Security Software User Interface

Security threats pose a massive problem to any company which has a software component. The industry that exists to protect information online is similarly massive, with many competitors providing in-depth analysis and protection for their users. With such a proliferation of companies, the exact ways in which they build their user interfaces can vary wildly from service to service. However there are trends that we can analyze to understand the design philosophies that they take. We'll consider some of these security services and see what features they consider key to their interfaces, and how those features are informed by their users' needs.

2.3.1 Splunk

Splunk [4] provides data analytics software to thousands of companies [5], and includes the product "Splunk Enterprise Security". This product offers users threat detection and the ability to investigate possible incidents quickly.

The main dashboard for Splunk [6] is its Security Posture view, seen in Figure 2-1, which provides a high-level overview of the entire security state of a system. In order to achieve that goal, it condenses an entire day's worth of security threats into just 5 numbers. The first 3, shown in red, are the number of intrusion alerts, infected hosts and malware signatures detected. An arrow is included to show whether that number

has increased or decreased in comparison to the previous day. The 2 numbers in blue tell a user how many hosts and accounts are being protected. These numbers provide a quick 5-second view of the system, and so massive differences in these values will point to a failure in the system.

Figure 2-1: Splunk Security Posture Dashboard

We also see that right below the 5-second overview of the system, Splunk gives the user ways of digging into the data. The three graphs, which list intrusions by severity, over time and by source, are all linked to the Intrusion Detection Dashboard (Figure 2-2), and clicking any of them pulls up that dashboard.

In this new view we get another 5-second overview, but this time of just the intrusions detected. This overview splits intrusions based on severity, color coding them so that red is the most critical intrusion type. Clicking on any of these values keeps the user on the Intrusion Security Dashboard, but limits the resulting statistics to only intrusions of the associated type. This allows users to easily filter intrusion by severity or action.

In the graphs and lists below the 5-second overview, intrusions are compared by their characteristics such as signature, source or location. Clicking on any of these characteristics moves users to the search page [7] where all events matching that characteristic are listed. As an example, selecting a specific signature in the Intrusion Detection Dashboard will result in a search for all instances with that signature.

Splunk has additional features other than intrusion detection that it can visualise,

Figure 2-2: Splunk Intrusion Detection Dashboard

such as firewall alerts or malware detection, but they all follow a similar user flow. A user is presented with an overview dashboard with a few key numbers and features; clicking on them will filter the results based on those features, and eventually that filtering leads to the search page. At this point, overview and statistics aren't useful as there are too few instances, so Splunk lists all of the instances for the user to peruse and investigate.

2.3.2 Crowdstrike

Let's consider Crowdstrike to be exemplary of a different approach to dashboards in the industry, that being the idea of "modular" dashboards. Crowdstrike encourages users to build their own dashboards by providing widgets [8] to display data (Figure 2-4). This might be in the form of number counts, line graphs over time, or simply filtered lists. The philosophy behind these designs is that no one understands what's needed from a dashboard more than the experts who use it. As such, it's better to give security experts all the tools to build their own dashboards.

Modular, customizable widgets are a fundamental part of their core design, and

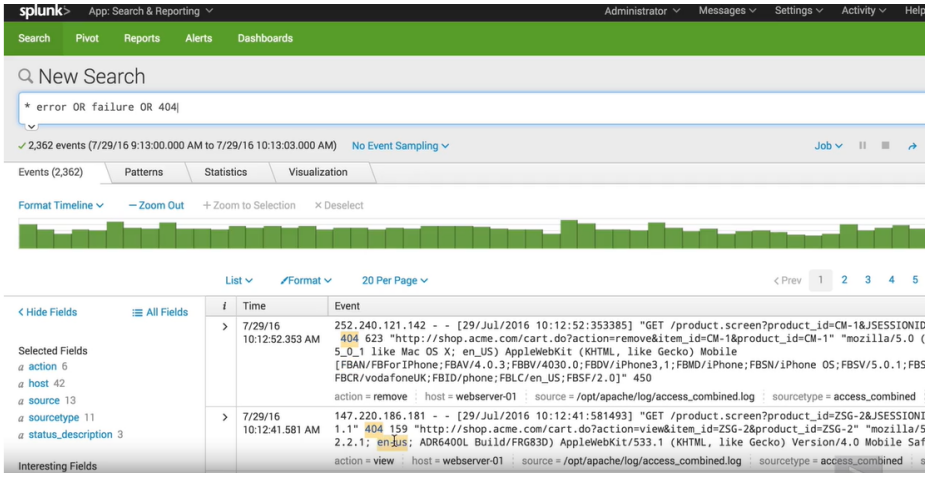


Figure 2-3: Splunk Search Page

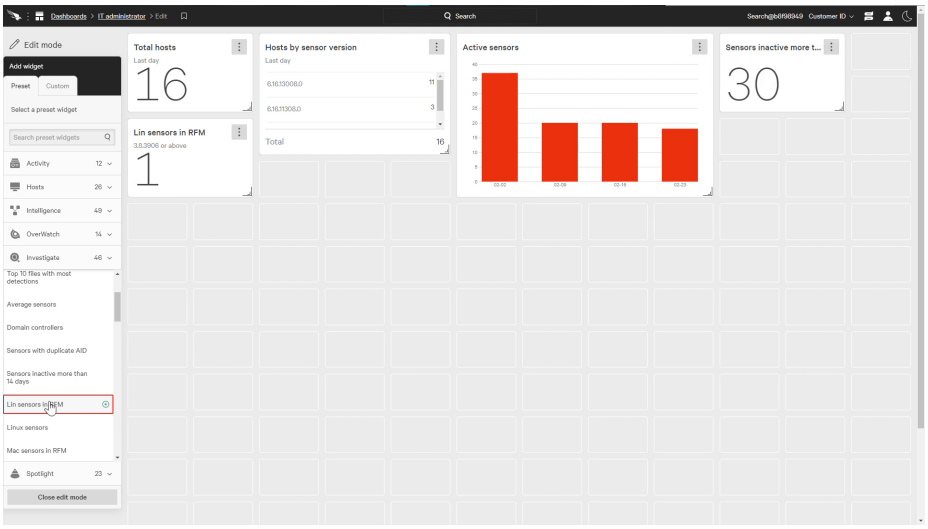


Figure 2-4: Crowdstrike Dashboard Creator

the interface reflects that. There are clear distinctions between widgets and they are all designed to fit in the predetermined grid. This isn't to say that Splunk and other similar security dashboards don't include customizable dashboards, but that Crowdstrike makes creating a dashboard mandatory for its users.

Chapter 3

System Architecture

3.1 Labeling Anomalies

Both Splunk and Crowdstrike offer labeling services in addition to their user interfaces. Common techniques to label anomalies include building comprehensive sets of rules and using machine learning classifiers. A novel machine learning based technique for labeling malicious request is currently being developed in the DBOS system.

This new technique can quickly and accurately label incoming requests based on their malicious nature. Because it is based on machine learning instead of more traditional rules, this system can output the probability of any incoming traffic being malicious. This has the distinct advantage of giving a user a clear idea of how likely an incident is of being malicious before they start investigating, which can help direct a user's efforts in investigation.

This does come at the price of being understandable. A rule is designed to capture a single type of attack, which means any request that is flagged has a corresponding reason to be flagged. Machine learning based approaches are often opaque, meaning that in order to fully understand why a request is malicious, a user will need to manually investigate that request. Therefore, making that investigation process as quick and painless as possible is vital.

3.2 Information Retrieval

Because DBOS is built on the fundamental assumption that databases are the building blocks for a system, retrieval of anomalies within the system is trivial. The current DBOS stack, which supports the Nectar Network web application, logs all incoming requests to Vertica. A labeling daemon can automatically label these logs as malicious or benign by applying the machine learning model to them. As such, retrieving the data for the web app is nearly trivial, executed as a series of SQL queries on the database.

Additionally, because Vertica is a column oriented database, it stores data tables by column rather than by row [9]. This is particularly efficient when users are filtering traffic based on the values in specific columns, which is the expected behavior for investigation. This makes loading information into the interface quick and easy.

3.3 Codebase

The codebase was developed in R using the Shiny [10] package, which is a web application framework. Importantly, Shiny establishes "reactive" bindings between inputs and outputs, allowing for an interactive user experience with minimal coding. Additionally, Shiny comes with many prebuilt widgets that are directly usable out-of-the-box or with suitable customization options. The structure of code utilizing Shiny can be split between the user interface (UI) and server logic. The UI specifies the layout of inputs and visualizations that are provided to the user in the web application itself. The server logic defines "reactions" to changes in user inputs that may result in new UI elements/visualizations along with internal changes to the server environment. The following example describes a typical interaction between the UI and server within a deployed web application. A user selects a filtering option from a dropdown menu. This change in the dropdown menu value prompts a new query with the newly selected filtering option. The returned data is then redisplayed in real-time to the user. This structure, in which user inputs immediately change the display,

makes Shiny applications reactive and provides immediate feedback to users. There are some cases where users will want to spend time editing their inputs before they are processed, in which case a more traditional submit button makes sense. However, the ability to respond to user input in real-time makes the user experience fluid and seamless, and is vital to the designed interface.

Chapter 4

Interface Design

The goal of the interface is to provide a security administrator with the necessary resources and visualizations to further investigate predicted/flagged anomalies. To that end, I have developed a local web application using R to facilitate anomaly investigation. In particular, the web application consists of three tabs that each display relevant data or visualizations that can be updated in real-time based on user input. The first tab (default tab on launch) is the overview page (Figure 4-1), which allows the user to apply various filters to the provenance data for interactive display. The second tab is the search page (Figure 4-3), which allows the user to directly query the provenance data using SQL commands as input. The third tab is the stats page (Figure 4-5), which allows the user to visualize anomaly trends through a historical line graph of detected anomalies.

4.1 Prioritizing anomalies

The main function of the overview page is to provide an interface for a user to filter anomalies based on the available logged fields and an anomaly threshold. This allows for anomaly prioritization based on the filtering options selected. An example of how this process is enacted can be seen in Figure 4-1. The left sidebar specifies the available filtering options and the right main content displays the output data table based on the selected filtering options.

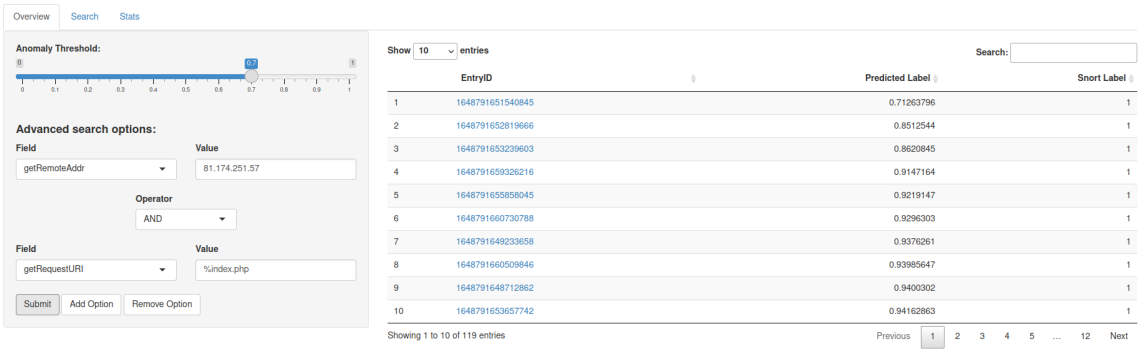


Figure 4-1: Web Application Overview Page

The sidebar consists of two components: the anomaly threshold and advanced search options. The anomaly threshold is simply a slider between 0 to 1 that specifies the minimum value of the predicted label that should be displayed in the output data table. The advanced search options allow a user to filter for any number of HTTP fields with associated values. The full list of available HTTP fields in the dropdown menu is as follows: 'cookie: JSESSIONID', 'getAuthType', 'getContentType', 'getContextPath', 'getLocalName', 'getMethod', 'getPathInfo', 'getPathTranslated', 'getProtocol', 'getQueryString', 'getRemoteAddr', 'getRemoteUser', 'getRequestURI', 'getRequestURL', 'getRequestedSessionId', 'getServerName', 'getServletPath', 'header: accept', 'header: accept-encoding', 'header: accept-language', 'header: connection', 'header: cookie', 'header: dnt', 'header: host', 'header: referer', 'header: upgrade-insecure-requests', 'header: user-agent'. An operator dropdown menu appears between consecutive field-value pairs to denote whether the relation between them should be 'AND' or 'OR'. It has been noted that this formatting makes it impossible for the user to specify complex or nested conditions based on field-value pairs, but this functionality is available in the search page through SQL queries. The syntax would be fairly long and convoluted, but it is fully possible within the search page, which takes in raw SQL commands. The add and remove options are fairly self-explanatory, and allow the user to define the number of field-value pairs that will be considered. The submit button processes the chosen filtration options into a SQL command that is then outputted in the main content panel. For reference, the SQL command associated with the filtration options selected in Figure

4-1 is “SELECT LOG_TIMESTAMP, RAW_REQUEST, MODEL_LABEL, SNORT_LABEL FROM HTTPLOG_REQUEST_LABELED WHERE MODEL_LABEL > 0.70 AND RAW_REQUEST LIKE ‘%"getRemoteAddr" : "81.174.251.27"%’ AND RAW_REQUEST LIKE ‘%"getRequestURI" : "%index.php"%’ ORDER BY MODEL_LABEL".

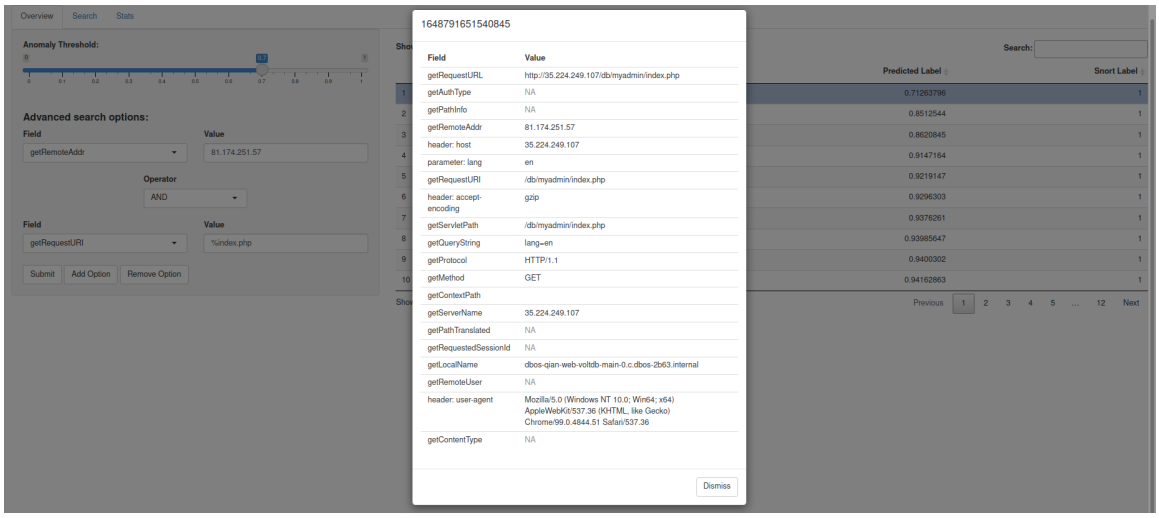


Figure 4-2: Web Application Overview Popup

The main content is a filtered list of labeled entries in the database. There are three columns that correspond to the “EntryID", “Predicted Label", and “Snort Label". The EntryIDs are the UNIX timestamps that correspond to unique HTTP requests. Importantly, each timestamp is a clickable link that generates a popup containing the raw HTTP field-value pairs in a human readable format. Figure 4-2 displays a popup that results from clicking on the first EntryID from the data table in Figure 4-1. Note that the value of “getRemoteAddr" matches the input value of “81.174.251.27", and the value of “getRequestURI" matches the input value of “%index.php", in which % denotes a wildcard character that represents zero or more characters. The predicted label is the outputted probability from the previously described ML model that predicts anomalous Nectar Network requests. The output is consistent with the anomaly threshold option selected in the sidebar that corresponds to a minimum value of 0.7. The Snort label corresponds to ground truth, which was used to train and evaluate the machine learning model. Notably, new entries will only have a predicted label with no available Snort label. By default, entries are sorted in

ascending order by the predicted label, but there are ascending and descending sort options available for each column.

4.2 Investigating anomalies

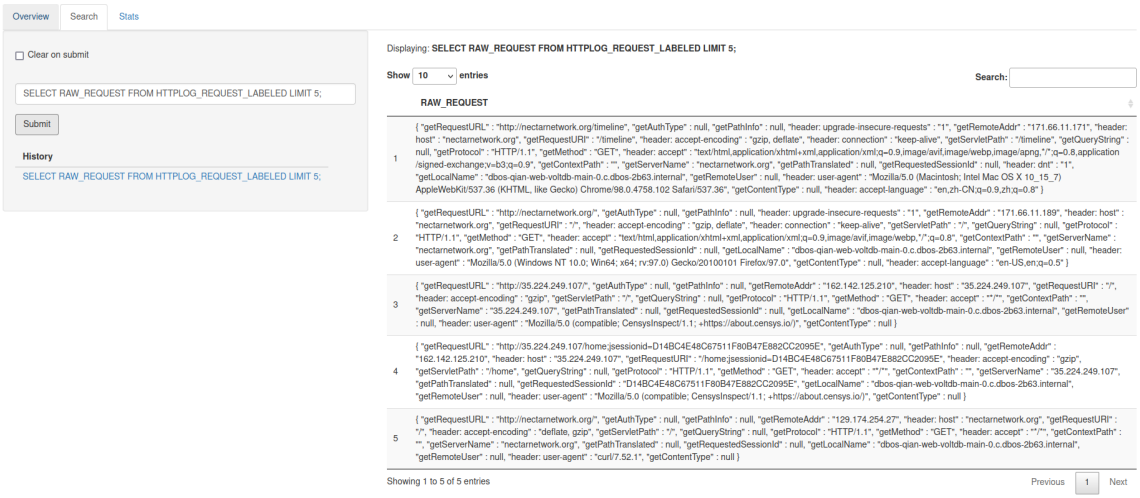


Figure 4-3: Web Application Search Page

The main function of the search page is to enable the user to directly query the underlying Vertica provenance database. This is done by inputting and executing arbitrary SQL queries, which gives the flexibility needed to handle complex investigations. An example SQL command has been executed and displayed in Figure 4-3. The left sidebar specifies user inputs and options, and the main content panel displays the result of the SQL query.

The sidebar provides several quality of life features to facilitate the search process. The top checkbox dictates whether the user text input below should be cleared every time a new query is submitted. Importantly, queries are not sanitized or modified before submission. The web application operates under the assumption that it is being used by a security administrator or other party with intimate knowledge of the underlying Vertica provenance database schema. An additional quality of life feature is that queries can be submitted by either using the submit button or pressing the keyboard enter button. Once a query has been submitted and displayed, a link is

generated under the history section with the exact text submitted. This allows the user to easily resubmit a previous query by simply clicking on the associated link. Note that the history only maintains the previous 10 submissions, and each submission in the history is unique. Repeatedly submitting the same query will simply result in that query staying at the top of history.

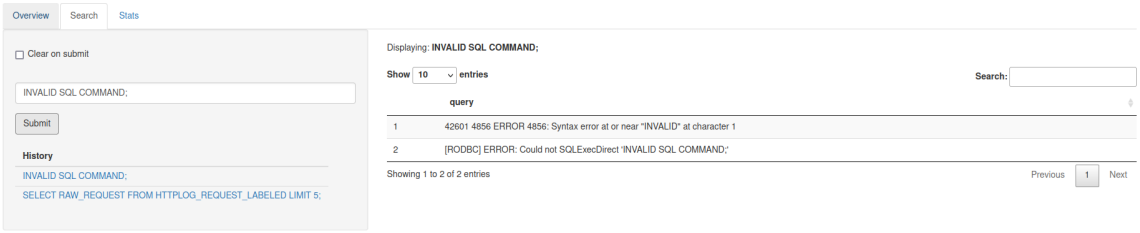


Figure 4-4: Web Application Invalid Search

The main content, on the right of Figure 4-3, consists of the table that results from the most recent query submitted by the user. Importantly, the text of the most recent query is displayed in bold above the table for ease of use. In the event of an invalid SQL query, the web application handles it gracefully by displaying a two-entry data table that consists of the problematic syntax as well as the original query string. An example of an invalid SQL query can be seen in Figure 4-4.

4.3 Historical Analytics

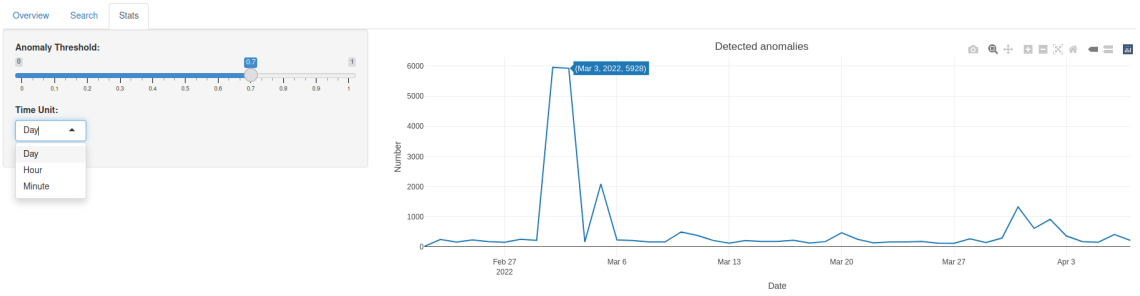
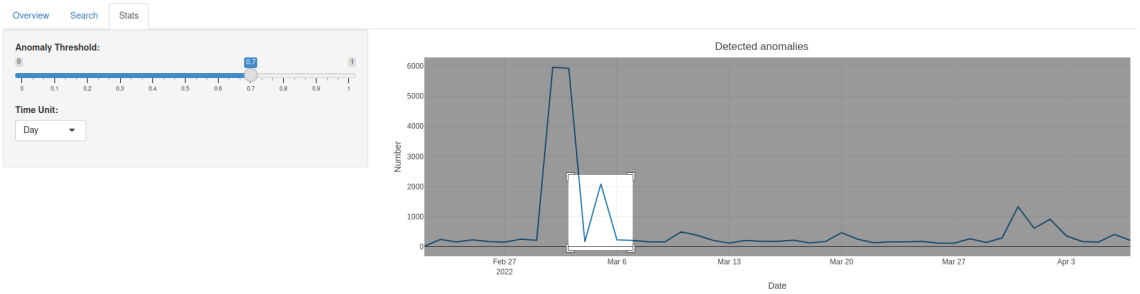


Figure 4-5: Web Application Stats Page

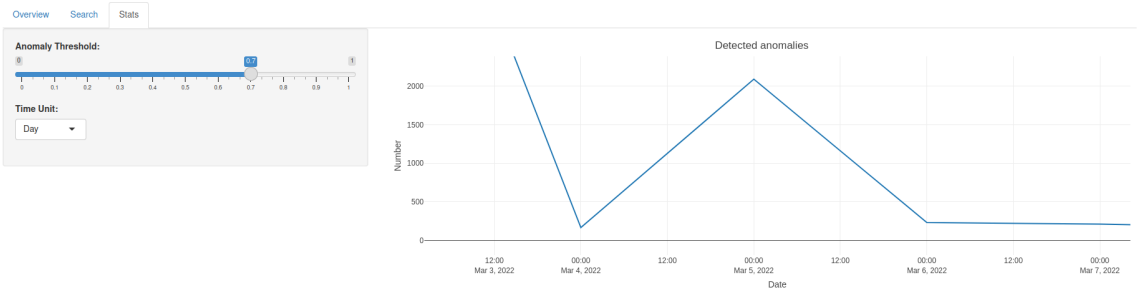
The main function of the stats page is to provide a historical view of anomalous behaviors by visualizing aggregate anomalous activity over various timeframes and

dates. The default settings and associated line plot are shown in Figure 4-5. The left sidebar specifies all available user options, and the main content panel displays the resulting line plot. It should be noted that there is no submit button as in the other pages since the line plot reacts automatically to changes or selections in the user options. This is simply because there are no text inputs in the stats user options that would cause unnecessary and excessive queries upon all textual changes whatsoever. Users can use the historical view to recognize peaks in the number of attacks which might point to a concerted attack effort, or to understand long term trends in their security.

The sidebar consists of only two components: the anomaly threshold and time unit. As in the overview page, the anomaly threshold dictates the minimum value at which predicted labels are considered to be truly anomalous. The time unit has three options: day, hour, and minute. The time unit indicates the granularity at which anomalies should be aggregated and displayed in the line plot.



(a) Selecting a subsection of the line graph.



(b) Displaying the newly selected subsection.

Figure 4-6: Web Application Interactive Stats Plot

The main content displays the line plot with the selected user options. Importantly, the line plot is generated using the plotly package, which is specialized for

interactive visualizations. Figure 4-6 demonstrates one such capability in that sub-sections of the line graph can be selected and magnified by the user. Also, as shown in Figure 4-7, there are tooltips accessible by user mouse hover that displays a tuple of the exact date and number of detected anomalies. The top right portion of the plot contains icons that showcase the full utility of a plotly plot. From left to right, the corresponding icons allow the user to download the plot, zoom freely, pan, zoom in, zoom out, autoscale, reset axes, show closest data on hover, and compare data on hover.

Chapter 5

User Studies

Throughout the development of the web application, I actively sought out user feedback to better design the application itself and understand the needs of potential users. Concretely, I have conducted a longitudinal study involving five industry professionals to gather and incorporate feedback into the web application. In an initial and final study given to the five participants, they were given a brief explanation on the purpose for the web application as well as a live demonstration of its main features at the time. After each page was described and displayed in its entirety, each user was allowed to freely explore the page to their satisfaction. Afterwards, they were given a page-specific survey to determine whether key aspects of the page related to functionality, aesthetics, and ease of usage were well implemented. Users were asked to give numeric scores between 1 (strongly disagree) to 5 (strongly agree) for each question. They were also given an open-ended prompt at the end of each page survey to provide general feedback and suggestions.

5.1 Initial Study

The initial study was performed in early March when the general functionality, design, and implementation of the web application was completed. Table 5.1 displays the questions given to and aggregate scores given by the participants after the guided, live demonstration and free exploration process of each page. Scores were generally

Table 5.1: Initial Study Results

Question	*Aggregate Score*
Overview Page	
How well do you feel you can prioritize anomalies based on the available sidebar options?	2.8
Does the data table display relevant anomaly information clearly and effectively?	3.2
Would you consider the layout of the page to be well-organized and aesthetically pleasing?	3.8
Is it simple and easy to use the provided interface to produce your desired output?	3.8
Search Page	
How well do you feel you can investigate anomalies based on the available sidebar options?	3.6
Does the data table display relevant anomaly information clearly and effectively?	3.2
Would you consider the layout of the page to be well-organized and aesthetically pleasing?	3.8
Is it simple and easy to use the provided interface to produce your desired output?	4.4
Stats Page	
How well do you feel you understand historical anomaly trends based on the available sidebar options?	4.8
Does the line plot display relevant anomaly information clearly and effectively?	4.0
Would you consider the layout of the page to be well-organized and aesthetically pleasing?	4.6
Is it simple and easy to use the provided interface to produce your desired output?	4.4

above average, but there were some poorer scores in the overview and search pages that warranted further development. The general feedback regarding both pages was that the data table returned by queries was relatively cluttered and lacking in functionality. At the same time, the limited number of available side options in both pages made it difficult to truly prioritize or investigate anomalies in detail.

5.2 Web Application Revisions

Several quality of life changes were implemented following the initial study to improve upon the user interface and general functionality of the web application. In particular, all data tables were replaced and rendered using the DT package, an R interface for working with JavaScript DataTables, rather than the default Shiny package. This allowed for search options within the table itself as well as column-specific sorting options (ascending or descending order). At the same time, the HTTP field-value pairs were not displayed in plain text as another column in the overview page. Instead, the timestamps were used as unique entry IDs with clickable links. When clicked on, these links produced popups that displayed HTTP field-value pairs in a standard table format, making it much more accessible. With respect to the overview sidebar options, the advanced search options were added to let users filter based on any of the fields in the HTTP requests. This allows users to quickly and efficiently dive into only the requests they're interested in. With respect to the search sidebar options, the unique history provided a much needed quality of life improvement that made investigations much simpler and easier. Users can now recall a previously executed query to see the result again, or to make small edits to the query as they explore the data. Additionally, the line plot in the stats page was replaced and rendered using the plotly package rather than the ggplot package. This allowed for interactive features incorporated directly into the plot itself as described in the "Historical Analytics" section of the "Interface Design" chapter.

5.3 Final Study

The final study was performed in early April after careful consideration and implementation of feedback received by those in the initial user study. Table 5.2 displays the questions given to and aggregate scores given by the participants after the guided, live demonstration and free exploration process of each page. Scores were extremely positive, and the vast improvement compared to their corresponding initial values suggest that the web application revisions were generally favorable and beneficial.

Table 5.2: Final Study Results

Question	*Aggregate Score*
Overview Page	
How well do you feel you can prioritize anomalies based on the available sidebar options?	4.4
Does the data table display relevant anomaly information clearly and effectively?	4.8
Would you consider the layout of the page to be well-organized and aesthetically pleasing?	4.4
Is it simple and easy to use the provided interface to produce your desired output?	4.6
Search Page	
How well do you feel you can investigate anomalies based on the available sidebar options?	4.4
Does the data table display relevant anomaly information clearly and effectively?	3.8
Would you consider the layout of the page to be well-organized and aesthetically pleasing?	4.2
Is it simple and easy to use the provided interface to produce your desired output?	4.4
Stats Page	
How well do you feel you understand historical anomaly trends based on the available sidebar options?	4.8
Does the line plot display relevant anomaly information clearly and effectively?	4.8
Would you consider the layout of the page to be well-organized and aesthetically pleasing?	4.6
Is it simple and easy to use the provided interface to produce your desired output?	4.6

Chapter 6

Closing Remarks

6.1 Future Work

While I am pleased with the current version of the interface, there are additional features which would be welcome. Many of these features relate to the daily user experience with the interface, as opposed to the experience of investigating a particular anomaly; most would be necessary for a professional grade consumer product.

6.1.1 User Profiles

As it currently stands, the interface operates without saving any user preferences or changes. While this works for investigating anomalies, for someone who is using the software daily the ability to save the current session and reopen it another day would be useful. Associating saved sessions, as well as session histories, with a user profile would do just that.

6.1.2 Multiple Databases

Currently, the interface operates on a single Vertica database which represents the labeled logs for a single web app. However, in modern companies it is not uncommon to have tens of applications, each of which is hosted on a different system. For a system administrator trying to protect all of these applications at once, being able

to quickly identify anomalies specific to each application is vital. This would likely involve pooling from multiple databases and including added filters so that users can search for anomalies in specific databases.

6.2 Conclusion

Overall, the current interface contains most of the core features necessary for investigating anomalies as they come into a DBOS web application. Using the novel machine learning approach to labeling anomalies offers security administrators to quickly narrow down which possible anomalies are most likely real, letting them quickly prioritize their investigative time and effort. Providing historical analytic allows experts to quickly identify rapid spikes in malicious traffic, which in turn allows for rapid mobilization against any attacks. And when investigating possible anomalies in order to determine their severity and impact on a system, allowing security experts to quickly search for the exact incidents they are interested in and see all of the related information is vital. There are still improvements that can be made, but the initial ability of the interface to accomplish these key goals is positive.

Bibliography

[1] A. Skiadopoulos, Q. Li, P. Kraft, K. Kaffes, D. Hong, S. Mathew, D. Bestor, M. J. Cafarella, V. Gadepally, G. Graefe, J. Kepner, C. Kozyrakis, T. Kraska, M. Stonebraker, L. Suresh, and M. Zaharia, "DBOS: A dbms-oriented operating system," *Proc. VLDB Endow.*, vol. 15, no. 1, pp. 21–30, 2021.

[2] D. Kumar, Q. Li, J. Li, P. Kraft, A. Skiadopoulos, L. Suresh, M. J. Cafarella, and M. Stonebraker, "Data governance in a database operating system (DBOS)," in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare - VLDB Workshops, Poly 2021 and DMAH 2021, Virtual Event, August 20, 2021, Revised Selected Papers* (E. K. Rezig, V. Gadepally, T. G. Mattson, M. Stonebraker, T. Kraska, F. Wang, G. Luo, J. Kong, and A. Dubovitskaya, eds.), vol. 12921 of *Lecture Notes in Computer Science*, pp. 43–59, Springer, 2021.

[3] "Owasp top 10 security report." https://owasp.org/Top10/. Accessed: 2022-04-16.

[4] "Splunk website." https://www.splunk.com/. Accessed: 2022-04-17.

[5] "Companies using splunk." https://enlyft.com/tech/products/splunk. Accessed: 2022-04-16.

[6] "Using the infosec app for splunk." https://splunk-infosec-documentation.readthedocs.io/en/latest/5%20-%20UsingInfoSec/. Accessed: 2022-04-16.

[7] "Splunk: Search, filter and correlate." https://www.splunk.com/en_us/resources/videos/search-filter-and-correlate.html. Accessed: 2022-04-16.

[8] "How to use crowdstrike dashboards." https://www.crowdstrike.com/blog/tech-center/customizable-dashboards/. Accessed: 2022-04-16.

[9] "Why all column stores are not the same," tech. rep., Vertica.

[10] "Shiny documentation." https://shiny.rstudio.com/reference/shiny/1.0.5/. Accessed: 2022-04-17.