University of Nevada, Reno

Navigating Cyberthreat Intelligence with CYBEX-P: Dashboard Design and User Experience

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Adam Cassell

Dr. Sergiu Dascalu, Co-Advisor

Dr. Shamik Sengupta, Co-Advisor

May, 2021

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

ADAM T. CASSELL

Entitled

**Navigating Cyberthreat Intelligence with CYBEX-P:
Dashboard Design and User Experience**

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

MASTER OF SCIENCE

Dr. Sergiu Dascalu, Co-Advisor

Dr. Shamik Sengupta, Co-Advisor

Dr. Janet Usinger, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

May, 2021

Abstract

As the world's data exponentially grows, two major problems increasingly need to be solved. The first is how to interpret large and complex datasets so that actionable insight can be achieved. The second is how to effectively protect these data and the assets they represent. This thesis' topic lies at the intersection of these two crucial issues. The research presented in the thesis learns from past work on applying data visualization to multiple domains, with a focus on cybersecurity visualization. These learnings were then applied to a new research area: cybersecurity information sharing. The frontend considerations for CYBEX-P, a cybersecurity information sharing platform developed at UNR, are discussed in detail. A user-facing web application was developed from these requirements, resulting in an approachable, highly visual cyberthreat investigation tool. The threat-intelligence graph at the center of this dashboard-style tool allows analysts to interact with indicators of compromise and efficiently reach security conclusions. In addition to research and related software development, a user study was conducted with participants from cybersecurity backgrounds to test different visualization configurations. Subsequent analysis revealed that the misuse of simple visual properties can lead to perilous reductions in accuracy and response-time. Recommendations are provided for avoiding these pitfalls and balancing information density. The study results inform the final functionalities of the CYBEX-P front end and serve as a foundation for similar prospective tools. By improving how insights can be extracted from large cybersecurity datasets, the work presented in the thesis paves the way towards a more secure and informed future in a technology-driven world.

Dedication

I dedicate this thesis to my tirelessly supportive family, without whom my education and this research would not be possible. The unparalleled nurturance provided by my mom, dad, and little brother deserves all of the credit for this accomplishment. I thank my Grandma Nancy for the encouragement to pursue graduate studies, and my cousin Dillon for inspiring me to tackle impossible challenges.

Acknowledgments

I would like to thank my co-advisors Dr. Dascalu and Dr. Sengupta for their invaluable guidance throughout my graduate studies. My appreciation is extended to my additional committee member, Dr. Usinger, for her time and generosity.

All past and present members of the CYBEX-P research team have my thanks for their incredible collaboration on this work. Additionally, my co-researchers for the user-study, Zachary Black and Tapadhir Das, were instrumental in this research.

This material is based in part upon work supported by the National Science Foundation under grant number 1739032. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Humankind is generating data at a rate that would have been unfathomable at the turn of the century. Making sense of all these bits, and connecting them in meaningful ways, is one of the predominant challenges of our time. The solutions to many of the word's problems may very well be hiding in plain sight, just waiting to be uncovered. The invention of the internet means that there are now very few limits to how society's collective knowledge can be connected. However, with this enormous opportunity, comes tremendous risk. There are bad actors who seek to take advantage of the information superhighway we all enjoy. Alongside enhancing knowledge, growing businesses, and curing diseases, the internet can also play host to crippling infrastructure attacks and data breaches. As Figure 1.1 demonstrates, the financial impacts of these events each year is severe. Like every civilization before us, we must act diligently to protect our next advancements from potentially devastating setbacks.

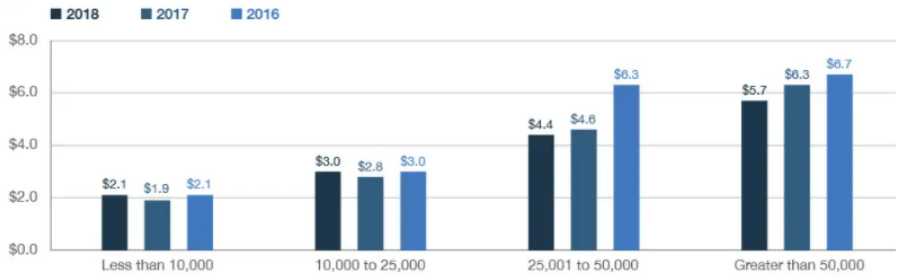


Figure 1.1: Average cost of different data breach sizes in the U.S. from 2016 to 2018, measured in millions of U.S. Dollars. [6].

The first goal of this thesis is to advance the field of understanding complex and

interrelated data. The second and deeply connected goal is to build an interactive software solution that makes this data safer. The former is accomplished through an exploration of data visualization. It is imperative that humans, not just machines, can understand data implications. This research looks especially closely at how to best visualize relationships between objects in datasets – these connections often yield crucial insights in a myriad of domains. The latter goal (protecting the world's data), is aided by the creation of a comprehensive cyberthreat analysis tool. This is a fully-featured application that helps cyberanalysts understand and defend data traveling across networks. Both of these motivations culminate in the creation of the CYBEX-P web application. This is a methodically developed software product that uses data visualization and a human-centered approach to unlock actionable threat insights from network data. The application makes contributions to interactive data visualization, which exists at the intersection of human-computer interaction (HCI) and data science. Armed with a human-centered design, this software can protect the very infrastructure that data discovery, sharing, and analysis rely on.

To reach its goals, this research leveraged existing literature, human-centered design approaches, software engineering processes, and experimental evaluation. Thoughtful consideration was given regarding how to represent relationships among complex and variably-sized datasets. Different approaches to visualizing graph-based data structures, while emphasizing malicous items within them, were examined. Key challenges exist within the scopes of network visualization and the investigation workflow. A graph design was needed that can effectively display indicators of compromise (IOCs), their threat levels, and the connections between them. In addition, a fully interactive interface was required to facilitate direct data manipulation and focused observation. All of these considerations were addressed in a final implementation that serves as a functional display of this thesis' research.

The rest of this thesis is structured as follows: Chapter 2 provides background knowledge that is foundational to understanding the research and development contributions. It introduces visual analytics, collective threat intelligence concepts, the

CYBEX-P platform, and technologies used to build the frontend application. Chapter 3 inspects key research in cybersecurity visualization and current approaches to investigation workflows. Motivation for the CYBEX-P web application and an overview of the proposed solution are given in Chapter 4. Main goals, intended users, and the development approach for the application are outlined in detail. Chapter 5 presents software modeling and implementation considerations. It describes the requirements specification, architecture, and implementation challenges for the software. The resulting application is demonstrated in action in Chapter 6. An overview of the interface and features are given, including sample usage scenarios. Chapter 7 evaluates the implementation against alternative designs as part of a user study. Main objectives, experimental setup, results, discussion, and study conclusions are provided. Finally, overall research conclusions and future work are contained in Chapter 8.

Chapter 2

Background

2.1 Visual Analytics

2.1.1 The Value of Data Visualization

No matter the context, all computer data can be simplified down to the same, deceivingly simple set of electrical signals. Switches that are on or off, numerical values of one or zero, the boolean concept of 'true' or 'false' - all of these metaphors highlight a crucial and inherent truth. Reading these descriptions, one's subconscious immediately goes to work connecting abstract concepts to fundamental physics. This is a massively underappreciated skill of the human mind that allows us to communicate ideas freely and effectively.

Contrast our reality with one where we could only communicate in a literal sense. Instead of reading words on a screen, it would be immensely challenging if all digital print was in binary. Likewise, messages with emojis and other images are far more vivid and captivating than antique communication methods like morse code.

Datasets of any kind, especially those that are large and/or complex, can be as indiscernible as the scenarios presented above. Even if a small number of experts can distinguish the meaning from these signals, the value is inaccessible to a wider audience and to new interpretations. Herein lies the importance of data visualization - it is the tool that bridges the gap from first principles to interpretable insights. Without this field, there may still be mathematics, physics, and engineering. The arts and human storytelling may likewise still exist. However, these domains would

seldom intersect. Logic and the senses must intertwine for ideas to transmit from person to person, organization-to-organization, or industry-to-industry.

2.1.2 Survey of Visualization Methods for Complex Data

It follows that data visualization must excel in the two things it aims to connect. First, visualizations must be rooted in a thorough understanding of the underlying data. Second, they should be expertly tailored to the audience that will interpret it. From these requirements, a massive and constantly-evolving variety of visualization methods have been developed. Across countless disclipines and contexts, a number of visual frameworks and applications have emerged. This is both especially challenging and interesting when it comes to complex datasets that are becoming increasingly available in today's connected world.

Heer et al. establish a comprehensive baseline of current standard practice for visualizing complex data in their work, *A Tour Through the Visualization Zoo* [22]. This survey provides recommendations ranging from how to build effective time-series plots to how to visualize hierarchies and networks. The authors focus on these more complex, often multi- dimensional types of data rather than simpler charts and graphs. One of the key themes is that all visualizations share a similar set of 'DNA'; that is, they represent the mapping between data attributes and visual characteristics. More specifically, the claim is that position, size, shape, and color are the binding frameworks of any good visualization.

A few primary categories of data are examined, beginning with time-series. The authors mention this is one of the most widely used forms of recorded data, and drive an endless number of practical domains from business to engineering. One thing that is emphasized is that this type of data often needs to be compared against other time-series data concurrently. This is especially true when trying to observe potential correlations over time. Some sub-types of time-series plots include index charts, stacked graphs (shown in Figure 2.1), small multiples, and horizon graphs [22]. Some of these, like index charts, are better for showing relative changes. Others,

like horizon graphs, are ways to fit more absolute data into extremely limited space without sacrificing resolution of the data you are displaying.

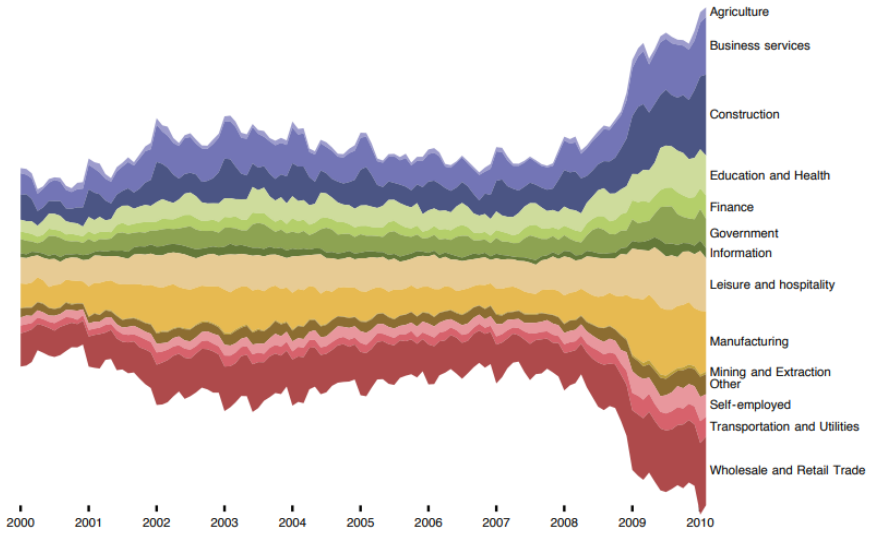


Figure 2.1: Example of a stacked time-series plot for U.S unemployment numbers [22].

The next category is visualizing statistical properties, which is useful for understanding data relative to their observed or expected distributions. Statistical depictions are often used when analysts want to select data that are fit to statistical models and need a way to visualize whether a sample fits such requirements. One of the more powerful plots mentioned by Heer et al. are Q-Q plots, which compare probability distributions by graphing their quantities against each other [22]. This is incredibly useful for understanding whether some sample data is Gaussian, for example, or perhaps follows some other distribution. Pair-wise scatter plots and heatmaps are examples of incredibly practical statistical visualizations used for multivariate data analysis.

Additional data visualization categories include mapping, hierarchies, and graphs. Mapping generally refers to visuals that are based on cartographic projection of geographic data. Some datasets have natural organization and structures to them. These are good candidates for hierarchical visualizations, such as node-link diagrams

or enclosure diagrams. Lastly, network graph visualizations represent relationships between data. Relational tasks are the focus of this thesis' later chapters. They require visualizations that carefully represent connections and communicate contextual significance. Graphs can be visually implemented as adjacency matrices (shown in Figure 2.2), arc diagrams, or force-directed layouts. The latter communicate objects in two-dimensional space and are highly effective for facilitating exploration. As studied later in this work, force-directed graphs are a great fit for displaying network topologies and can help facilitate cyberthreat analysis.

Figure 2.2: Example of an adjacency matrix that visualizes relationships between various artist names [26].

2.1.3 Visual Analytics for Cybersecurity

Next, it is worth examining the core achievements and common principles that have shaped how data visualization is used in cybersecurity contexts. As Cooke et al. explain, cybersecurity involves "intensely cognitive" tasks, and is made up of a "large multi-layered sociotechnical system of analysts, computers and networks" [28]. Many factors contribute to the difficulty of analyzing this data, including its often-large size and complexity. Organizing this data into actionable results, often by exposing network and actor relationships, is no easy task. Solving this problem requires thoughtful consideration of how to map data to its most useful visual form. Doing so can improve how quickly and accurately threats are identified, both proactively and reactively. The end result is a safer world for countless individuals and organizations.

Cybersecurity-related data visualization has quickly developed over the last two decades, with the sub-topic rising to prominence in the early 2000s. A retrospective on the area's trends over time, provided by Crouser et al., gives some insight into how it began and evolved [8]. Analyzing the content of cybersecurity publications over time shows that low-level forensic analysis was the primary research focus early on. Around 2010, the research community began shifting to higher-level topics such as situational awareness and network defense. Over this more recent decade, this has mean that cyberthreat models and graph theory have moved to the forefront of some of the most important research in this space. Such concepts have directly led to increased attention around visualizing network relationships and connectivity.

There have been a number of key achievements under the Human-Computer Interaction (HCI) umbrella that have contributed to more effective cybersecurity software. This background survey focuses on Visual Analytics (VA) in particular, which describes any methods that extract value from visually-driven contexts. Visual analytics was originally defined by the US Department of Homeland Security as "the science of analytical reasoning facilitated by interactive visual interfaces", and this is the definition that is still widely used today [7]. This topic, as a whole, is one of

the most important advancements within the broader cybersecurity field. Researchers from the European Union Coordination Action expand on the significance of VA, noting that it is now the "medium of a semi-automated analytical process" [24]. More effective results, it is argued, can be achieved by leveraging the respective advantages of humans and machines. Machines excel at quantitative conclusions, yet humans remain superior at deriving intuitions when given imperfect contexts.

There are three major sub-achievements that have directly led to the effectiveness of VA within cybersecurity and other fields. Lavigne et al., in their overview of the state of the art in VA, consider the following developments to be the most significant: visualization capabilities, interaction science, and analytical reasoning [25]. This work is in agreement with these conclusions, and briefly describes each of these advancements in this section.

Visualization itself is perhaps the oldest achievement of the three mentioned. Throughout recorded history, it has been used to present information in order to assist in deriving insights. Yet, the development of core visualization principles should be considered paradigm-shifting even by today's standards. Without them, humans would quickly lose a key medium of communication and knowledge ingestion. Today's visualization capabilities were made possible by key discoveries in human psychology. One of these is the idea of pre-attentive features. These are "properties that are detected very rapidly and accurately by the low-level visual system" [25]. This has led to today's use of visualizations that leverage these features to minimize the amount of cognitive workload required to reach conclusions through visual stimuli. For cybersecurity, this means threats and other critical data can be inferred much more effortlessly. Examples of commonly utilized properties are color, motion, size, orientation, and curvature, and are pictured in Figure 2.3.

Implementing interactivity into cybersecurity visualizations is a more recent achievement that has had profound impact. In the 21st century, the variety of devices and input/output methods have continued to grow rapidly, affording more ways humans and computers can interact. The key advancement is that experts have, through

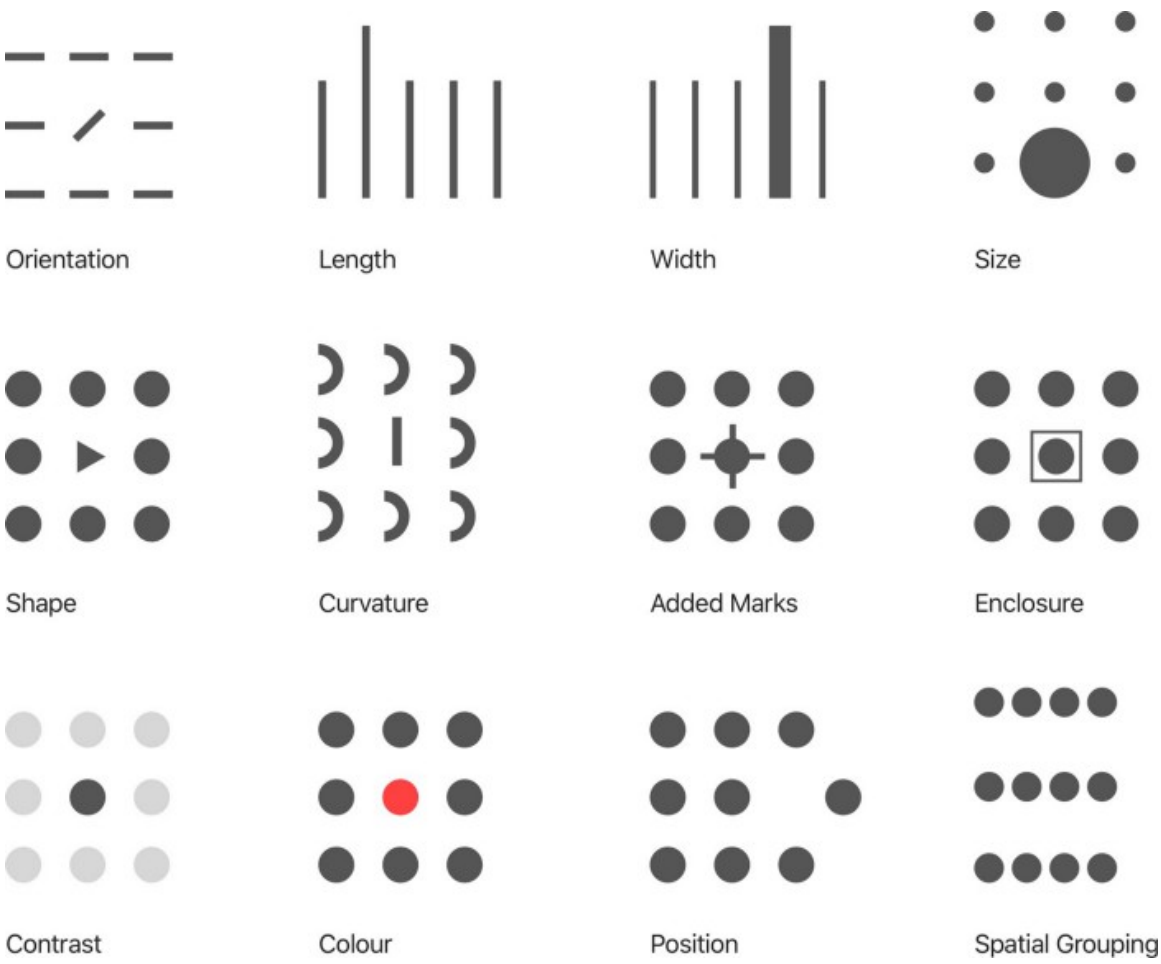


Figure 2.3: Types of pre-attentive visual features derived from research in cognitive science [33].

extensive human research, determined fundamental interaction principles that maximize effectiveness. One guiding principle that is especially relevant to cybersecurity investigations is to follow natural knowledge-gathering approaches. Users should be presented with overview analysis first, where important items are highlighted. Then, data should be filterable with deeper details exposed on demand [23]. Thanks to modern graphics advancements, low latency reactivity to user input is possible even in visualizations that render large numbers of points. The parallel development of these technical capabilities (graphics, I/O, etc.), as well as our understanding of human problem-solving, led to modern achievements in interaction that are central to many cybersecurity applications.

The third major achievement related to visualizing data relationships is the conclusions derived from research on analytical reasoning. This is again rooted in the cognitive sciences. There are a few key capabilities that have contributed to these breakthroughs in analytical reasoning. The first, as mentioned above regarding visualization, is that human workload is significantly reduced. This leads to more informed and quicker decision making. This is also thanks to reduced 'search' times, where large and complex data is more easily compartmentalized and interpreted thanks to visual cues [25]. We also can expose patterns that are non-obvious before data is visualized. These, among many other things, have all contributed to significant augmentation of human reasoning abilities in high-stakes domains such as cybersecurity.

Clearly, leveraging the full scope of human senses is increasingly advantageous when sharing information. Cybersecurity is a critical field in which these methods can make very measurable impacts.

2.2 Collective Threat Intelligence and CYBEX-P

2.2.1 Cyberthreat Intelligence

Computing environments have never faced more threats to their security than at this moment in time. With the increasing usage of connected computer devices and the internet, cyberspace is a hotbed for attacks. Cyberthreats are occurring at a much more frequent rate and are evolving to become more lethal. The number of cyberthreats and data breaches in the United States has increased ten-fold between 2005 and 2019. These data breaches have successfully exposed billions of records and have cost organizations billions of dollars [6]. In 2018, it was reported that hackers conducted an attack every 39 seconds, with an average of 2,244 times a day [9]. It is imperative that strong countermeasures are developed and maintained in order to combat these alarming trends.

Cyberthreat intelligence is defined by a collection of important considerations. The discipline aims to discover and document key facts pertaining to cyberattacks.

This includes identifying attacks that have occurred historically and are likely to happen in the future. It is crucial to understand how these can be detected, avoided, and when possible, neutralized. Alongside understanding attack events, cyberthreat intelligence is also concerned with the details of the actors involved. This includes the actors' motivations, capabilities, past actions, and more [3]. Analyzing all of these components helps yield important investigative insights, such as identifying which vulnerabilities certain threats may target.

To engage in cyberthreat intelligence gathering, cyberanalysts need a robust set of tools. Traditionally, the most primitive type of investigations have been conducted by parsing log files and other structured systems data. While this certainly can be effective when pieced together by skilled analysts, there are many parts of the investigative process that can be augmented, automated, or made more precise. To meet these demands, a myriad of frameworks, applications, and advanced methods have been developed. Some solutions pertain more to the backend infrastructure for information gathering and system monitoring. Another set of tools are intended to interface directly with cyberanalysts. Examples of the latter include visual event reports and applications to organize investigative data (several of these are detailed in Chapter 3). This work focuses on hands-on software that is used and manipulated by investigators, introducing many HCI considerations. These details are evaluated in depth in later chapters.

2.2.2 Information Sharing and the CYBEX-P Platform

In recent years, cybersecurity information sharing has become an important area of research that aims to reduce the threat of cyberattacks for participating organizations [10] [18] [45]. The main motivation behind this research domain is to facilitate the sharing of cyberthreat information between organizations using a common platform. The more information that is shared by contributors, the more aware all members can be of threats to their assets. Using this practice, organizations can defend against similar cyberattacks and data breaches. Exposing additional relationships and threat

metrics allows contributors to be better collectively protected. Figure 2.4 illustrates the concept of cybersecurity information sharing.

The Cybersecurity Information Exchange with Privacy (CYBEX-P) platform, developed at the University of Nevada, Reno, was developed to enable collaborative sharing of cyberthreat information [35]. CYBEX-P empowers more effective discovery of new threats and swift distribution of threat signatures. Figure 2.5 represents the platform architecture of CYBEX-P and the location of the front end within the larger data pipeline.

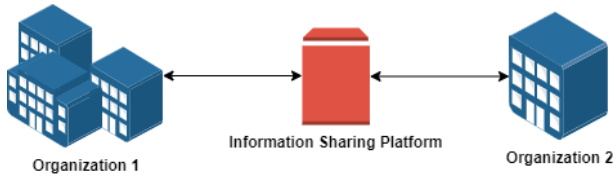


Figure 2.4: Cybersecurity information sharing.

The central CYBEX-P platform is built on a mutual value policy, where users gain access to the system by contributing their own data. Contributors can include businesses, hospitals, banks, universities, government agencies, and other organizations. Users can submit their own network event logs and other report data, which is then anonymously aggregated into a backend database. This database uses Tahoe, a universal, structured database format developed by the CYBEX-P team. Some of the most important elements of these database records are indicators of compromise (IOCs). These are any entities that may be of investigative interest within event data. These include IP addresses, URLs, domain names, emails, and more. This data is stored as objects in NoSQL format. Database records can be subsequently processed to reveal insightful IOC relationships based on their associations with common events.

Whenever data is submitted, it is validated and processed into the Tahoe format. This is done together with other contributors' data, combining it together into one database. The process is privacy-preserving, and the result allows for a variety of powerful operations. These include data look-ups, such as returning event data that corresponds to some criteria. Other capabilities include computing aggregate statis-

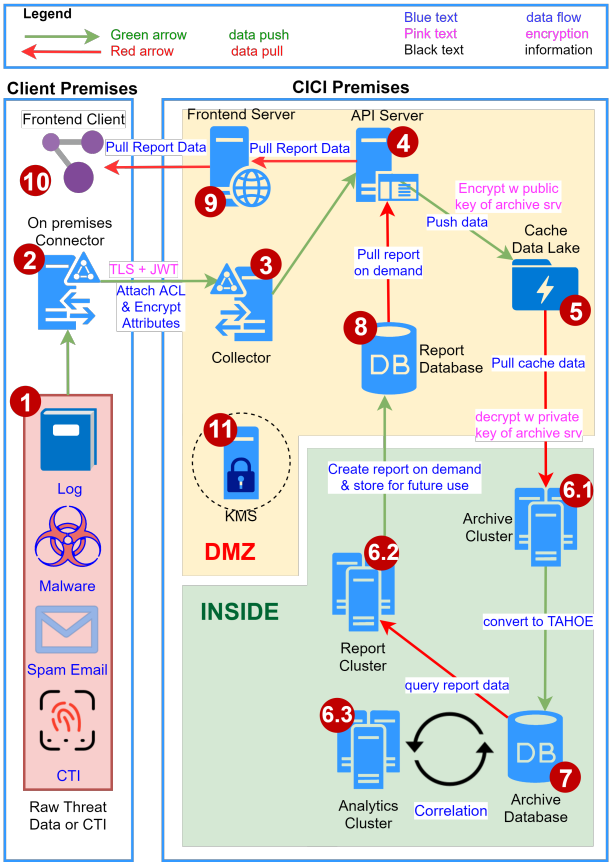


Figure 2.5: System architecture diagram for the CYBEX-P platform.

tics (such as the number of times an IOC is seen in the entire database). The Tahoe format is also well-suited for machine learning tasks and other big-data analysis.

One of the backend components of CYBEX-P that is highly relevant to this work is its threat ranking feature. The system would have limited effectiveness if it were to rely on narrowly-sourced threat statistics, especially if limited to the organizations' internal data. Likewise, the stored data would have minimal value if threat classification was left completely up to user interpretation. CYBEX-P solves this challenge using the confidence of large-scale, real-world event data. Such an approach does not assign threat rankings based on some 'black box' policy. Instead, it relies on empirical observations of what is seen in real-world networks. All IOCs in the database are tagged with records of their context within different categories of events. Specifically, the system tracks the number of times an IOC is seen in benign or malicious con-

texts. As a result, computed threat scores can be communicated to users and guide investigations. These computations serve as the foundation for threat analysis within the frontend application.

2.3 Technologies

The CYBEX-P web application is built using multiple existing technologies. This section details the backend and frontend tools necessary for development.

CYBEX-P's web application has a Python-based back end that uses the Django web framework [12]. Django supports creating applications with a comprehensive suite of features and extensions that are helpful for creating web applications quickly and efficiently. The Django Rest Framework [13] is used to build and serve the API endpoints that the client-side code sends its requests to.

The frontend for the CYBEX-P web application uses React [34] as well as independent static HTML5/CSS pages. Vis.js is used for rendering of graph visualization elements [46]. React uses JavaScript to drive component logic and integrates with npm for package management [32]. Babel is used to compile React code into traditional JavaScript code that the browser can understand [1]. All of these technologies enable streamlined development of single-page web applications.

Beyond this background, more details and additional technologies are described in Chapter 5. Working knowledge of the tools and frameworks in this section was a prerequisite for developing the CYBEX-P web application.

Chapter 3

Related Work

3.1 Key Research in Cybersecurity Visualization

There are several major research and development directions when it comes to visualizing relationships in data. There is a subset of these research directions that is especially relevant to the cybersecurity context. Several major topics are summarized quite effectively by Gates et al. in their position paper on cybersecurity visualization. These are categorized as: visualization for a specific goal, visualization for exploration, visualization as a stepping stone, visualization for evaluation, and visualization as evidence [16]. These classifications have been adapted and expanded upon for this work, which are presented below.

Designing for Specific Goals. This first research direction may seem trivial, but it is a critical component to get right when it comes to designing visualizations. In many cases, a particular type of outcome or workflow is expected, such as a security analyst that is simply wanting to be alerted to malicious actors in a network. In other cases, the goal may be more pro-active than reactive. Further still, tasks could be theoretical or centered around exploration of real data. This all means that user-studies, interviews, and other user-centered research is necessary to extract the core requirements of various tasks. A large amount of cybersecurity research fits into this category of simply trying to understand the many unmet software needs of today's security analysts. One such example is EEVi, which is a model for determining

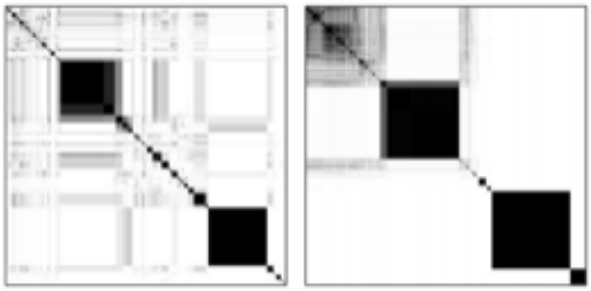
effective visualization methods for cybersecurity tasks [37]. This method translates qualitative interview responses about the relevance of different visualization characteristics into quantitative metrics. Research like this can help inform the state-of-the art for common security needs.

Exploration. It is often the case that analysts do not have a particular objective in mind when looking at network data. Instead, the task may be more exploratory, where the goal is to understand basic relationships and patterns in the data. These workflows can often lead to more specific questions and workflows afterwards. Without an objective in mind, visualizations must 'zoom out' and allow the user to design their own path – with unbiased connections and insights easily accessible. OverFlow is a tool developed to help users decipher the known properties and interactions of their network traffic [17]. The creators of the tool emphasize that the goal is to both help users understand the state of their systems and spark subsequent ideas for further investigation.

Bridging the Gap. There are cases where users already know the question they want to ask, but don't yet have any intuitions on what to do to find answers. This somewhat niche part of the workflow is just as important of a research area as the others. That is because investigations often begin with limited information sets, where the full picture is not yet realized. Effective solutions in this category often help guide a path from one analysis tool to another, depending on the user's goal and the characteristics of their data. A tool that embodies this idea is BlackWidow, which monitors the dark web for relevant threat data [36]. This tool could also be classified under the category of exploration, but it serves as a great example of bridging the gap between tools. BlackWidow assumes a particular goal – in particular, it is designed to help guide analysis of malicious trends on the dark web. Yet, it isn't designed to automate a direct conclusion in a pre-configured fashion. Instead, it generates a graph of relationships between dark web data, and then tries to highlight a subset

of event data that may be especially useful. A security analyst can decide if these insights necessitate further investigation, and more quickly move to another tool that is specialized in that context. This subsequent tool would fall under the category of 'specific goals' mentioned above.

Security Evaluation. There is an area of research that is focused on evaluating the security strength of both real and hypothetical environments. Goals in this space are often more proactive and preventative. A great example of development in this area is RoleVAT, a tool that visually assesses user and permission tendencies [48]. This tool visualizes user permissions and user groupings in a network, which ultimately helps evaluate whether certain security policies can be effectively applied. Figure 3.1 demonstrates how RoleVAT generates a dissimilarity matrix to visualize the relationships among permission tendencies and user tendencies, respectively. Figure 3.1(b) implies that this particular network has tightly related subgroups of user tendencies, compared to the less clear groupings of permission tendencies in Figure 3.1(a). This visual informs a network administrator that they should enact user partitioning policies, but not necessarily permission partitioning. These types of visualizations are highly practical in a variety of industry settings.



(a) RoleVAT on permission tendency (b) RoleVAT on user tendency

Figure 3.1: Relational visualization for security evaluation implemented in RoleVAT [48].

Historical Evidence. One last area of development to consider is using visualization to retrace events and justify conclusions from the past. Sometimes, solutions in

this space are even needed to act as forensic evidence in criminal cases [16]. These types of scenarios often require stories to be told to individuals that may lack cybersecurity expertise. An example of such a tool is APT-Hunter, which was developed to help detect and trace account compromises using malicious login behavior [39]. This tool, and others like it, help analysts and executives at organizations prove and understand breaches that may occur.

3.2 Existing Works in Investigative Workflows

For this work, is important to consider the end-to-end workflows that cyberanalysts use on a daily basis. Some improvements to investigative workflows have come in the form of more compute-efficient tools. These include solutions like CLX, which is a Python library that improves data processing for infosec data scientists [4]. Another ongoing effort is the design of cloud infrastructure that can support cybersecurity-oriented machine learning workflows [5]. The focus of this thesis, however, is on the user-facing aspects of cyberanalysts' manual tasks.

Franklin et al. provide an excellent introduction to the challenges in this pace space using a human-centered approach . Their paper includes interview-led research on real-world workflows [15], which often rely on command line tools and custom scripts. Through multiple semi-structured converstations with a group of analysts, the authors identify four main categories of analyst tasks. These are alerting, thresholding, threat hunting, and reporting. When certain network rules are violated, these trigger alerts that analysts then must investigate. Thresholding refers to setting and optimizing the rules that drive those alerts. Threat hunting is described as "long term analysis of network activity for trends and targeted exploration" to find malicious behavior. Threat-hunting outcomes are eventually reported, resulting in better alerting thresholds and creating a continuous workflow cycle. Of these tasks, proactive threat-hunting generally gets allocated the least amount of time because the other steps require more frequent attention. Given this constraint, it is critical to make threat analysis as efficient and accurate as possible. The cyberthreat intelligence tool

contributed as part of this thesis prioritizes this part of the workflow. However, its design (featured in Chapters 5 and 6) anticipates seamless integration between the other workflow tasks. The authors also note that analysts crave better relationship identification; this includes things like understanding which network resources compromised credentials can access. This need has led to relational visualization being a key focus of this thesis.

There are a few advanced threat analysis tools that are gaining traction in industry. One of them is Splunk, which is a machine-learning driven dashboard system that supports a number of security-related use cases and chart types [42][20]. It has a comprehensive backend and frontend feature set that goes beyond just cyberthreat intelligence. However, its default supported visualizations do not cater to graph-based visualization of relational data.

A tool that does implement this type of visualization is SpiderFoot [41] [29]. It represents network objects as nodes, and basic relationships between them as edges. This is the most similar product to the implementation presented in this thesis, however it has some key limitations. First, its relationships are limited by data sources. CYBEX-P's collaborative event-based database, on the other hand, exposes a unique set of connections that may be missed by SpiderFoot. Second, supplemental interface elements in tools like SpiderFoot are often obstructive to the main graph view, as shown in Figure 3.2. Better graph-centered UI design is needed to provide a less disruptive experience. Third, SpiderFoot has a rather minimalistic approach to threat visualization. It marks potential threats with a simple, subtle flag. The visualization approach presented in this thesis, however, considers much more threat detail and context. As shown later in Chapters 5 and 6, this method communicates relative threat strength, secondary metrics, and event contexts directly using visual properties of the graph. The shortcomings of existing work in this space inspired the visualization goals for CYBEX-P.

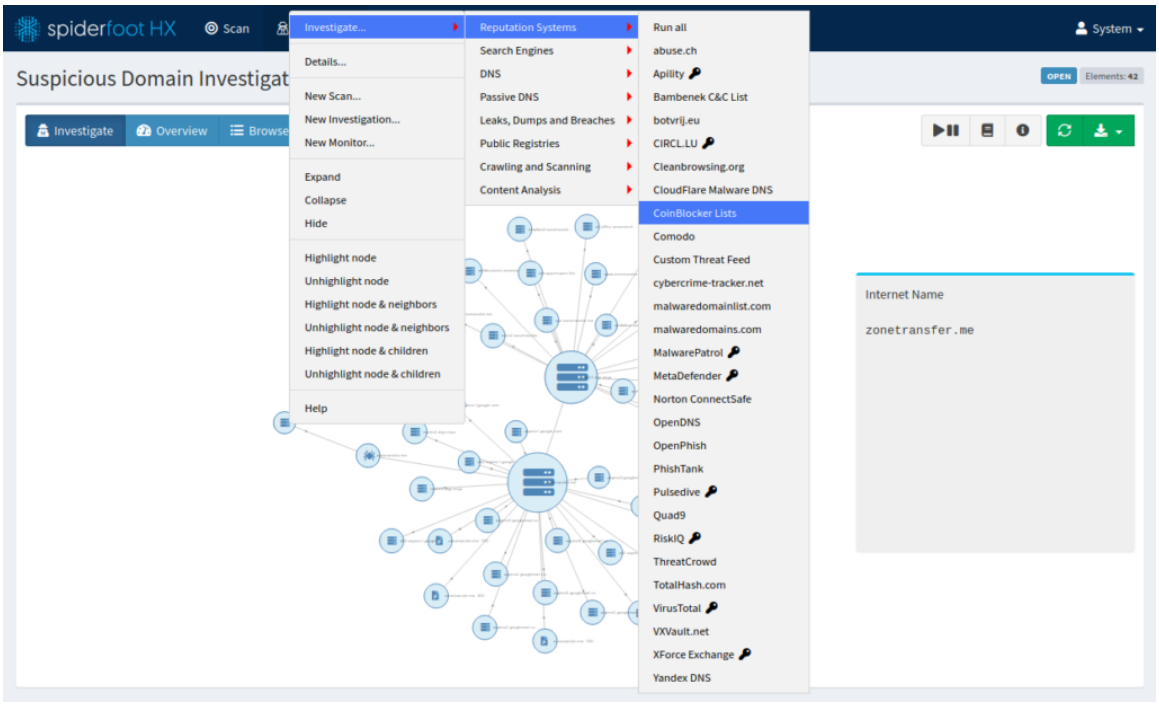


Figure 3.2: Existing network graph tools like SpiderFoot often have obstructive interface elements that are disconnected from the graph. Their visualizations also often lack detailed and glanceable threat context. [41].

Chapter 4

Motivation and Overview of Proposed Solution

4.1 Main Goals

This chapter introduces the frontend web application for the CYBEX-P platform. The application is intended to be used by cybersecurity analysts in order to better understand threats to their networks and assets. It serves as an exploratory tool that visualizes relationships between indicators of compromise (IOCs). IOCs can be items such as IP addresses, Uniform Resource Locators (URLs), domain names, emails, and file hashes. These connections facilitate analyses relevant for investigations into threats on observed network graphs. The application leverages anonymously aggregated data from CYBEX-P's database to build insightful visual graph models. Through this approach, the front end reveals critical threat information available within the backend system.

The primary purpose of this software is to efficiently transform CYBEX-P's stored data relationships into actionable investigation paths for human analysts. The application is to be used with network objects and event data from a wide variety of organizations. Therefore the front end, like the underlying infrastructure, is designed to be highly scalable. Analysts for large and small organizations alike can utilize the same unified interface for a variety of investigation patterns. Subjects of analysis can include anything from a single endpoint to network graphs containing hundreds of nodes. The frontend applies CYBEX-P's aggregated threat statistics to these subjects

in a highly interactive and visual manner. The more instances in which something is reported in benign or malicious contexts, the more prominently that description is portrayed. The tool visualizes real encounters with real data at scale. It follows that the interface must instill confidence in how this data is displayed or risk forfeiting its tremendous value. Additional application features, such as graph commenting and investigation sharing, aim to address the daily routine tasks of cyberanalysts. These adjacent workflow elements are often neglected in other cybersecurity visualization research [15].

To satisfy the aforementioned needs, the CYBEX-P front end enables two primary usability goals:

1. Employ graph visualization to communicate threats to specified assets.

2. Provide an effective investigation workflow centered on the graph.

Sections 5 and 6 detail how the CYBEX-P web application satisfies both of these goals. This is accomplished through a comprehensive featureset and a human-centered approach to design.

4.2 Intended Users

The intended users of this application include cybersecurity analysts, researchers, and tech-savvy executives at security-concerned organizations. Large government organizations, international companies, and smaller research firms can all benefit from contributing to and using the CYBEX-P platform. Users may use the tool exclusively, or as a part of a more complicated workflow. It is expected that users of the system have basic networking knowledge so that they can understand the threat implications revealed by the system. However, being an expert on advanced network security is not necessarily required to extract some high-level value from the system. The following exercise illustrates the range of intended users. A series of questions are presented that could be asked to hypothetical users of CYBEX-P:

1. What is your background knowledge regarding cybersecurity?
2. What is your current process for gathering and analyzing cybersecurity data?
3. Do you prefer looking at raw data, or looking at graphical visualizations of the data?
4. When presenting this data, what type of information is most useful to summarize?
5. What data that is unique to the CYBEX-P database is most useful to you?
6. Do you prefer having dense information available at a glance, or a more basic display?
7. Do you prefer a colorful and color-coded interface, or a more serious basic palette?
8. What kinds of data would you want to explore more deeply through supplemental interfaces (such as a companion tabular data component)?

While the example responses below are fictional, they are modeled off of realistic personas identified while interacting with researchers in this space. They reflect the varying types of value that different roles would seek to get from the system. This storytelling exercise assisted in defining use cases and eventual requirements for the application. Real discussions with actual professionals are presented later in Section 7 as part of a user study.

Role 1 - University Cybersecurity Researcher The first proposed role is that of a graduate researcher at UNR. Specifically, it is that of one working on the CYBEX-P project as a researcher focused on cybersecurity trends. In regard to his background knowledge, this user is well-versed in cybersecurity terminology and concepts. He has spent substantial time reading scholarly journals and performing research on emerging trends globally. Currently, this researcher uses these papers and observes

study results to inform their own research and analysis. He also performs small, local studies himself by running analysis on sample data collected through the CYBEX-P database. Most of this investigative work is done by looking at raw data in a database or using basic database plotting tools. There is no centrally accessible interface he can use to peruse this same information easily. As a result, this researcher prefers looking at well-organized tabular data because he understands the data well and has yet to find an effective visual interface to replace that. However, he is highly interested in the improved efficiency that may come with visual navigation.

When this researcher needs to convey data and findings to others, he usually focuses on global, large trend information rather than individual, specific events. When it comes to what unique CYBEX-P data is most useful to this researcher, he is therefore most interested in 'count' statistics. This refers to how often a certain type of malicious event occurs, or how many times a specific IOC is associated with a malicious event. As a researcher, this person prefers to have as much useful information as possible displayed on the screen at a time. This researcher will then often want to take a closer look at certain data points and observe them in a more detailed or tabular form. He is knowledgeable enough that this data is not intimidating, and he does not like to leave any details out. A colorful, dynamic palette is preferred by this person for functional and organizational reasons. Color-coding helps keep track of the many types of data and objects that are potentially observable in the system. Even if it looks less 'professional' with many bright colors, it has far more practical utility.

Overall, this university researcher is excited for the prospect of this interface because it will provide greater usability and visualization of the raw data he is studying.

Role 2 - CTO of a Small Company The second role is that of a CTO at a small company, who needs to occasionally lead a cybersecurity analysis of her company's operations. As for background knowledge, this user has a strong technical and leadership skillset, but she is less familiar with the specific trends and concepts in

cybersecurity. Being the technology leader at a small company, she does not have someone specifically in charge of cybersecurity; nor does she have a cybersecurity team. Cybersecurity responsibilities thus fall on the CTO and some of her subordinates. Currently, this person gets most of her knowledge about security trends and patterns from reading articles, some networking courses back in college, and occasionally from outside consultants.

When it comes to studying particular security events related to her organization, this person works with software engineers at her company to identify strange patterns in their company's own systems. They may identify a malicious event, but rarely are able to dive deeper to understand its low-level details. They may simply move on after a vulnerability is patched and fail to prevent similar issues in the future. This is because they do not have the resources or talent themselves to know more about these trends. The CYBEX-P frontend interface would help solve this and be an invaluable tool for people like this CTO.

When it comes to looking at raw data, or graphical visualizations, this person definitely prefers visualizations. Even if she does not understand the details of the raw data, she can more easily train herself to notice important trends and malicious events with a visual aid. When presenting findings, these visuals are also easier to communicate to other, potentially less-technical people at the company (Like the CEO). As far as what the CYBEX-P database can provide that other data sources cannot, this CTO is mostly interested in specific malicious events rather than general trend data. This is because she needs to take action on specific IOCs that impact and threaten her company, as well as learn from breaches in the past. Without this platform, she may have been able to identify an event as described above; however, she would not have the increased contextual information that the CYBEX-P database provides. She wants to know what other entities may have been associated with a breach at her company, or perhaps what other types of organizations get hit with similar kinds of attacks. An interface that makes these details immediately apparent is enormously useful.

This CTO prefers a more streamlined and simplified display. This is because, as a non-cybersecurity expert, the intricate, low-level details become distracting and cluttered. She wants to see only the high-level information she needs. It is important to be able to navigate events quickly and highlight the most significant information. Then, she feels that the extra, detailed information should be optionally available if the user so chooses. For example, if one of the other engineers knows more about a suspicious part of the network, they should be able to optionally toggle on more advanced information. Also, the CEO at this company prefers a basic and less colorful default color palette for security reports. He shares a common misconception that cybersecurity tools should appear somewhat militaristic. Even if this is less functional, it reveals a desire for a professional-looking design that may be important when establishing credibility with non-technical executives.

As mentioned, this person is most interested in a select set of relevant threat information. This on-demand detail can be useful in tracing the sequence of events that led to a breach, ultimately allowing the CTO to enact changes that enhance future safety. Visuals can give her the confidence to identify more threats without needing expert consultation.

Another hypothetical persona is presented in Table 4.1 with an example of how the software may be used. Following a human-centered design approach, user diversity necessitated a flexible implementation. Careful consideration of user needs influenced every aspect of the implementation and design described in Section 5.

4.3 Approach Outline

To satisfy the motivations for the CYBEX-P web application, a thoughtful research and development approach was required. Initial prototypes were designed from a HCI-focused perspective. After creating a minimum viable product (MVP), development shifted to a more standard software engineering approach. This was appropriate because initial prototypes required rapid design iteration. Once the design matured, it was then important to begin developing the product using the same software engi-

Role	**FBI Cybersecurity Investigator**
Name	Agent Holmes
Details	Holmes has worked for government for all of his career. He is 37 years old, and has both a B.S. in Computer Science and M.S. in Cybersecurity. He sifts through hundreds of investigations a week.
Goals	Predict the greatest cyberthreat to U.S business interests for the upcoming month of November 2019.
Resources	As an FBI agent, he has access to state-of-the-art technology and computing resources. However, he does not have readily-shared data directly from businesses themselves without a warrant. Luckily, because the FBI is a contributing organization to the CYBEX-P platform, he can access the aggregated, anonymized security data of many organizations.
Scenario	Agent Holmes needs to quickly run an investigation and accompanying report on the biggest likely threats to large American businesses in the month of November. Holmes logs into the CYBEX-P Threat-Intelligence Graph front end as part of the FBI organization. He starts a graph by adding nodes for individual IPs that are attributed to the 10 most valuable fortune 500 U.S companies. Holmes then runs a query on all IP nodes to add all related IOCs and events to the graph. Holmes sees that denial-of-service events are linked to 8 of the 10 largest companies' IP's, making it far more prevalent than other attack types. Holmes can then issue a FBI report that it is in the United States' best economic interest to focus research on preventing DNS attacks. Using the application's supplemental trends about DNS events over time, he can also show that DNS attacks are being logged more frequently in CYBEX-P with every passing month.

Table 4.1: User Persona: FBI Cybersecurity Investigator

neering methods often used for industry products. The final deliverable needed to be built reliably, configured for production deployment, and maintained through increasing collaboration with other team members. The end result is a complete and fully documented application that will be handed off to other researchers and developers. Individual components of this approach are detailed in Figure 4.1.

The prototype/research phase consisted of an iterative HCI design process that included requirements analysis, design, implementation, and evaluation [38]. Chapter 7 presents a user study that evaluated the application design and informed subsequent

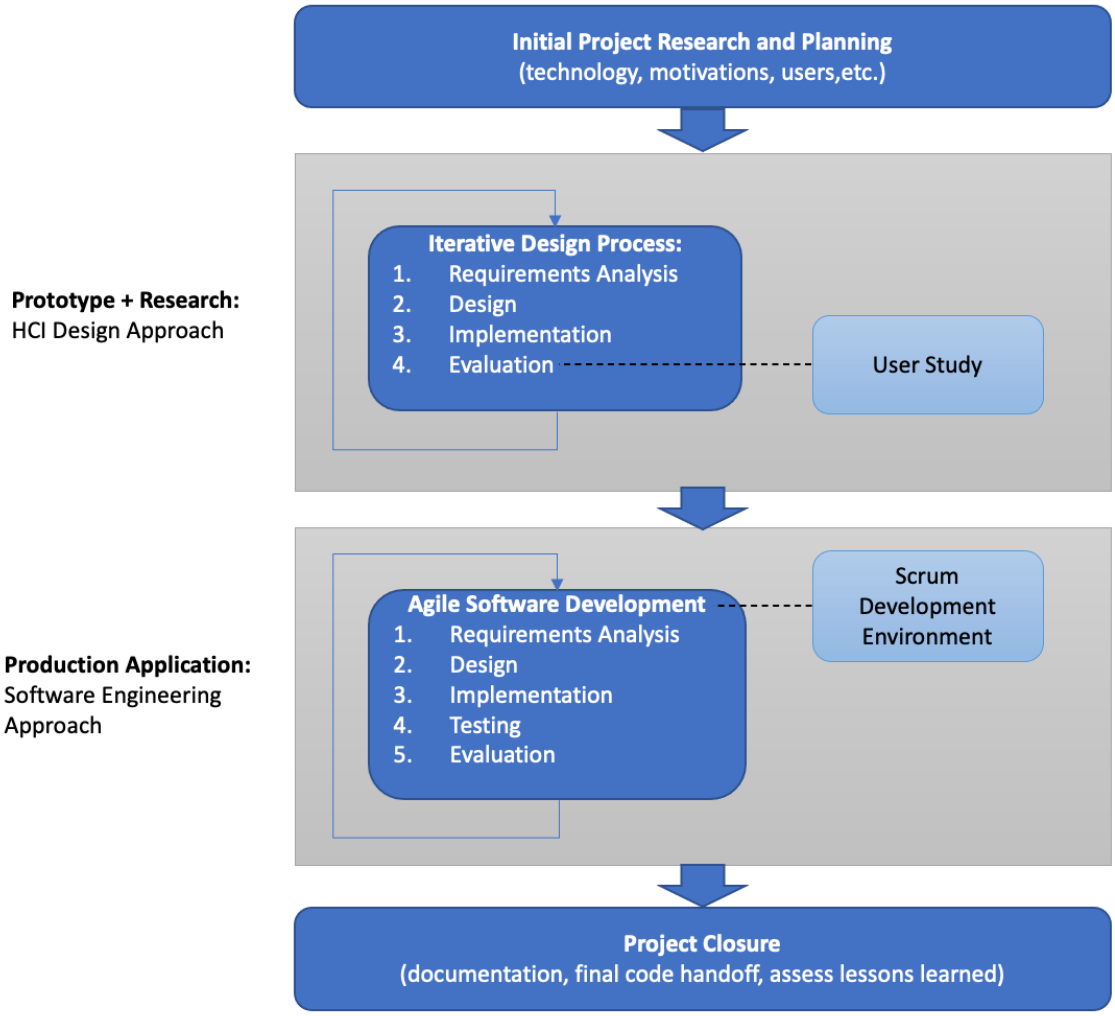


Figure 4.1: The research and development approach for the CYBEX-P Web App involved both HCI design and software engineering processes.

design cycles. The later phase of application development introduced testing to the design process. Scrum development practices were also established during this time, which improved both independent and team productivity. Issue tracking and project management software was utilized to monitor progress. After these changes, the process more closely resembled a typical software engineering approach [40]. Because the iterative HCI design process has many parallels with agile software development, this transition was frictionless.

Chapter 5

Software Modeling and Implementation

5.1 Requirements Specification

5.1.1 Use Case Modeling

As described in Chapter 4, requirements specification was a crucial part of the development cycle. The process always began by defining user stories for all the desired features and interactions. User stories are short statements that define usability goals by describing a specific scenario. The following is an example of a user story: "As a reader of this thesis, I want to read Chapter 5 so that I can understand how the requirements for CYBEX-P were formulated". An example within CYBEX-P may be: "As a cybersecurity analyst, I want to run threat analysis on a network graph so that I can identify vulnerabilities to my organization's assets".

High-level user stories can be broken down into use cases, which are more compartmentalized and specific. They describe how part of the system is interfaced with from the perspective of some actor. Examples of use cases for the CYBEX-P web app are detailed in Table 5.1. Being more atomic than user stories, use cases can serve as the building blocks of several types of diagrams. See Figure 5.1 for an example of how some of the above use cases can be visualized in a use case diagram. Note that it is not only human users that can be considered actors in the system. The CYBEX-P backend API, for example, also can be considered an actor. Some use cases in the

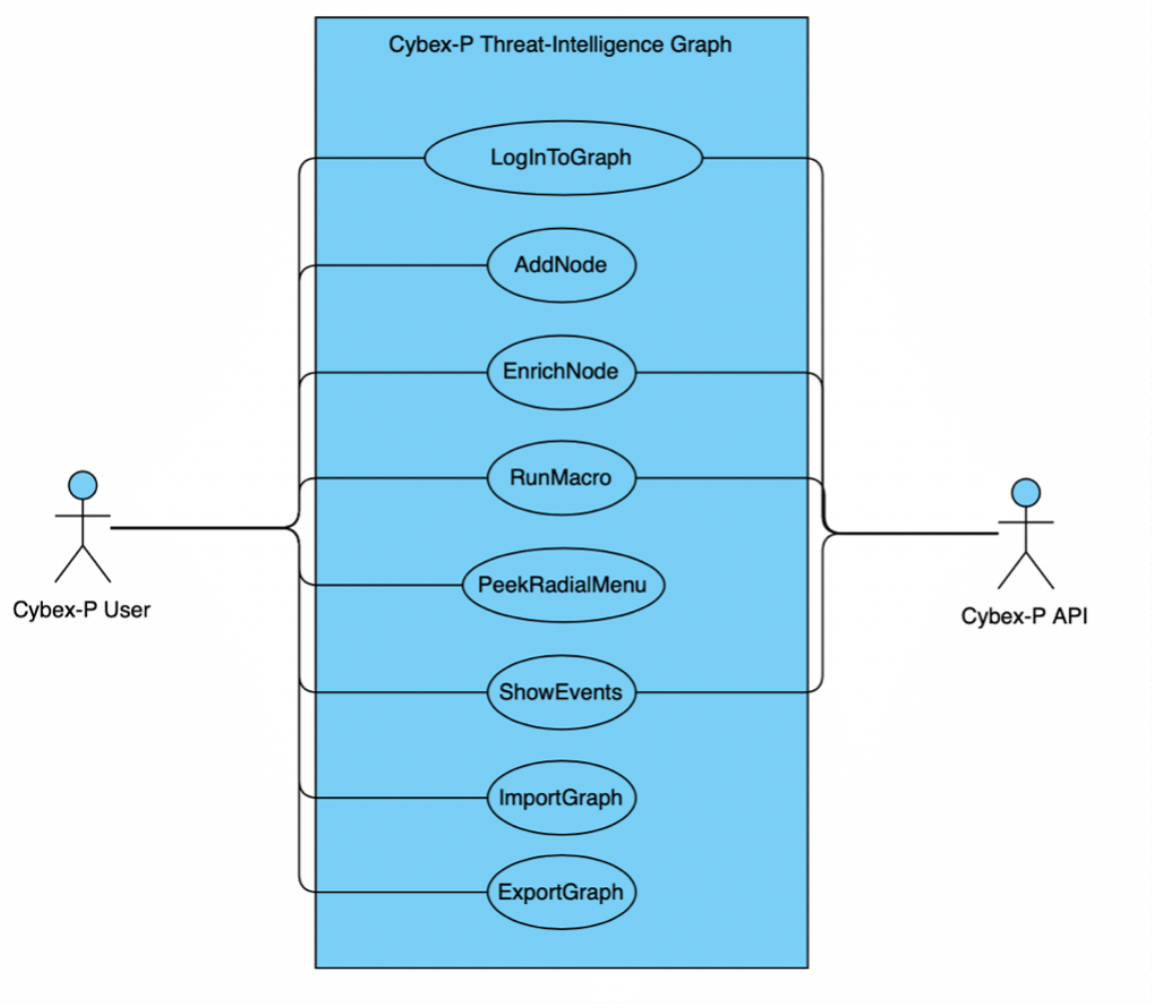


Figure 5.1: Use case diagram detailing eight usage examples for the application. Some use cases are client-side only, whereas others must interface with the backend API.

diagram pertain to multiple actors, while others may only involve one.

The items listed and depicted in this section are not a comprehensive list. However, they highlight key use cases that help CYBEX-P achieve its primary usability goals. The agile development process described in Chapter 4 constantly amended and re-prioritized these use-cases over the project's lifecycle.

5.1.2 Requirements

After all desired use cases were modeled, the next step was to define requirements. To accomplish this, use cases were broken down into their low-level, logical compo-

Use Case	Name	Description
UC01	LogInToGraph	The user navigates to the CYBEX-P Login Page. The user may enter their username and password, and is then redirected to the graph interface.
UC02	AddNode	The user selects the 'add node' button from the right-hand expandable menu. The requested node of the graph is rendered on the canvas.
UC03	EnrichNode	The user selects the 'enrich node' button from the right- hand expandable menu. Any attributes associated with that node get linked to the graph as nodes themselves.
UC04	RunMacro	The user selects the 'execute macro' button from the left- hand expandable menu. The desired macro is run, with its enrichments applied to all relevant nodes.
UC05	PeekNode	The user clicks a node on the graph with their mouse cursor. A radial menu appears around the node showing the enrichment options available for that node. Node data is shown in separate panel.
UC06	ShowEvents	The user investigates related events for a given node. All related events are rendered within tooltips along graph edges.
UC07	ImportGraph	The user selects the 'import graph' button from the bottom expandable menu. The user is shown a file explorer where they can select a JSON file to import.
UC08	ExportGraph	The user selects the 'export graph' button from the bottom expandable menu. A download modal is presented and the graph's JSON file is saved locally.
UC09	ThreatAnalysis	The user exposes threat information about graph nodes and the events that connect them. This can be analyzed node-by-node, or across the entire graph using macros.
UC10	PlatformTrends	The user uses secondary statistics and plots to understand aggregate trends within the system. This analysis is performed in a separate interface from the main graph canvas.
UC11	UploadData	The user uploads their own event data to the CYBEX-P database. The user supplies a supported file, timezone, and other metadata before submitting.

Table 5.1: Use cases for the CYBEX-P Web Application

nents. Requirements describe the functionalities that must be implemented for the system to facilitate the targeted use cases. Many of the requirements for CYBEX-P's web application are outlined in Table 5.2. A priority level is given for each requirement, with level 1 typically representing items needed for a minimum-viable-product (MVP). However, priorities shifted often in practice as part of the agile development process. An example of a requirement is that the system must display a node's available enrichment options when it is selected. All of the items in this list are considered functional requirements. These state what the system must do, rather than how the system should do it. Requirements that describe the latter are instead referred to as non-functional requirements.

There are several non-functional requirements that influenced development of the application. An example of this was limiting CYBEX query processing times to less than 30 seconds to improve overall performance. Another non-functional requirement was that two-factor authentication use FIDO2/WebAuthn (so that multiple authentication factors could be supported) [47]. The need for the application to render responsively in all major browsers was also an important example.

It is often useful to track which requirements are serving which use cases. This can help prioritize development tasks according to the use cases they directly enable. A simple tool for visualizing these relationships is a requirements traceability matrix, depicted in Table 5.3. Using this table, it is easy to see that Use Case 9 depends on more requirements than the others. This is sensible, because this is the threat analysis use case, which is one of the largest and most complex features in the application. This exercise helped approximate the implementation time of different use cases. Many complex use cases were developed early on to ensure they would be completed in the required timeframe.

As with the use case list, the requirements presented in Tables 5.2 and 5.2 are not comprehensive. Figure 5.2 better summarizes the most important use cases and requirements satisfied by the production version of the application. Created towards the end of development, this informal graphic provides a feature overview for prospective

users.

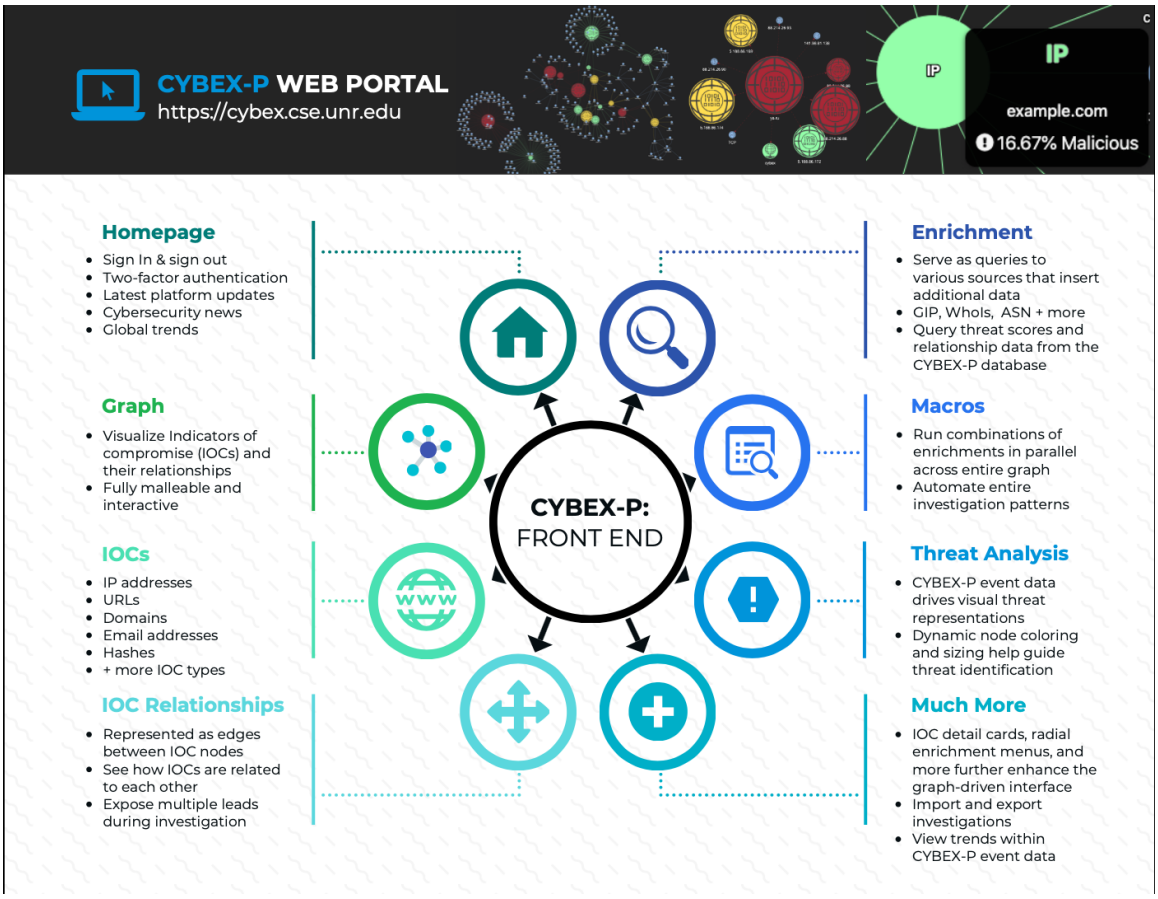


Figure 5.2: Infographic detailing the many features of the CYBEX-P web application.

5.2 Software Architecture

The CYBEX-P web application consists of several components that work in concert to provide a comprehensive software experience. These include a Python/Django back end and a React.js front end that communicate via a RESTful API. This web application back end is not to be confused with the CYBEX-P back end, which hosts the core API for querying the CYBEX-P database. The CYBEX-P back end and related services are the subjects of other works [35]; this Chapter focuses on the architecture and design of the web application only. Figure 5.3 is a system context diagram that visualizes some example interactions between backend data and frontend

Req.	Level	Description
R01	1	The system will allow the user to log in to the graph Interface using an authenticated username and password.
R02	1	The system will enforce two-factor authentication.
R03	1	The system will allow nodes of multiple IOC types to be added to the graph canvas.
R04	1	The system will allow nodes to be removed from the graph.
R05	1	The system will initially place nodes in visually optimized locations according to the graph's physics model.
R06	1	The system will allow node locations to be modified within graph physics constraints.
R07	1	The system will persist graph data, arrangement, and other user modifications between application re-loads.
R08	1	The system will visualize connections between nodes as graph edges with descriptive tooltips.
R09	1	The system will allow enrichments (GIP, WhoIS, CYBEX, etc.) to be queried for any graph node, adding the results to the graph.
R10	1	The system will allow stored macros to be run, applying enrichments to all nodes in the graph.
R11	1	The system will allow users to view more details about each macro in the macro menu.
R12	1	The system will allow users to see available enrichment options for a node using a radial menu.
R13	1	The system will allow users to view events in the CYBEX-P database that are related to a selected node.
R14	1	The system will allow the user to import a graph JSON file.
R15	1	The system will allow the user to export a JSON file to store the user's graph in an external file that can be later re-imported.
R16	1	The system will convey attribute types in the graph using a standard set of icons corresponding to each attribute category.
R17	1	The system will communicate threat rankings of each node through visual properties like color and size.
R18	1	The system will communicate the number of CYBEX sightings in each node's supplemental data.
R19	2	The system will allow the user to select a graph node and view related details directly in a connected, adjacent panel.
R20	2	The system will allow users to see CYBEX event contexts that connect related nodes while exploring the graph.
R21	2	The system will allow users to specify filters when querying related items, such as filtering results between certain time ranges.
R22	3	The system will allow users to visualize CYBEX event timeseries plots using a secondary view.
R23	1	The system will allow users to upload event data to CYBEX.

Table 5.2: Functional requirements for the CYBEX-P web application.

	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11
R01	X										
R02	X										
R03		X									
R04											
R05		X									
R06		X									
R07		X					X	X			
R08						X			X		
R09			X	X	X						
R10				X					X		
R11				X							
R12					X						
R13						X			X		
R14							X				
R15								X			
R16		X			X						
R17									X		
R18								X	X		
R19					X						
R20						X			X		
R21			X	X					X		
R22										X	
R23											X

Table 5.3: Requirement traceability matrix between use cases and functional requirements. Note that use case numbers are referenced as 'UXX' rather than 'UCXX' for formatting purposes.

elements. The intended functionalities of these components will be explored in more detail in Chapter 6.

5.2.1 Back End Architecture

The web application back end is built in Python using the Django web framework [12]. Django utilizes a MVT design pattern, consisting of model, view, and template components. The model drives all logic regarding the data in the application. The CYBEX-P web application's model incorporates Neo4j [30] graph-based databases to persistently store user investigation data. The web application's model also extends

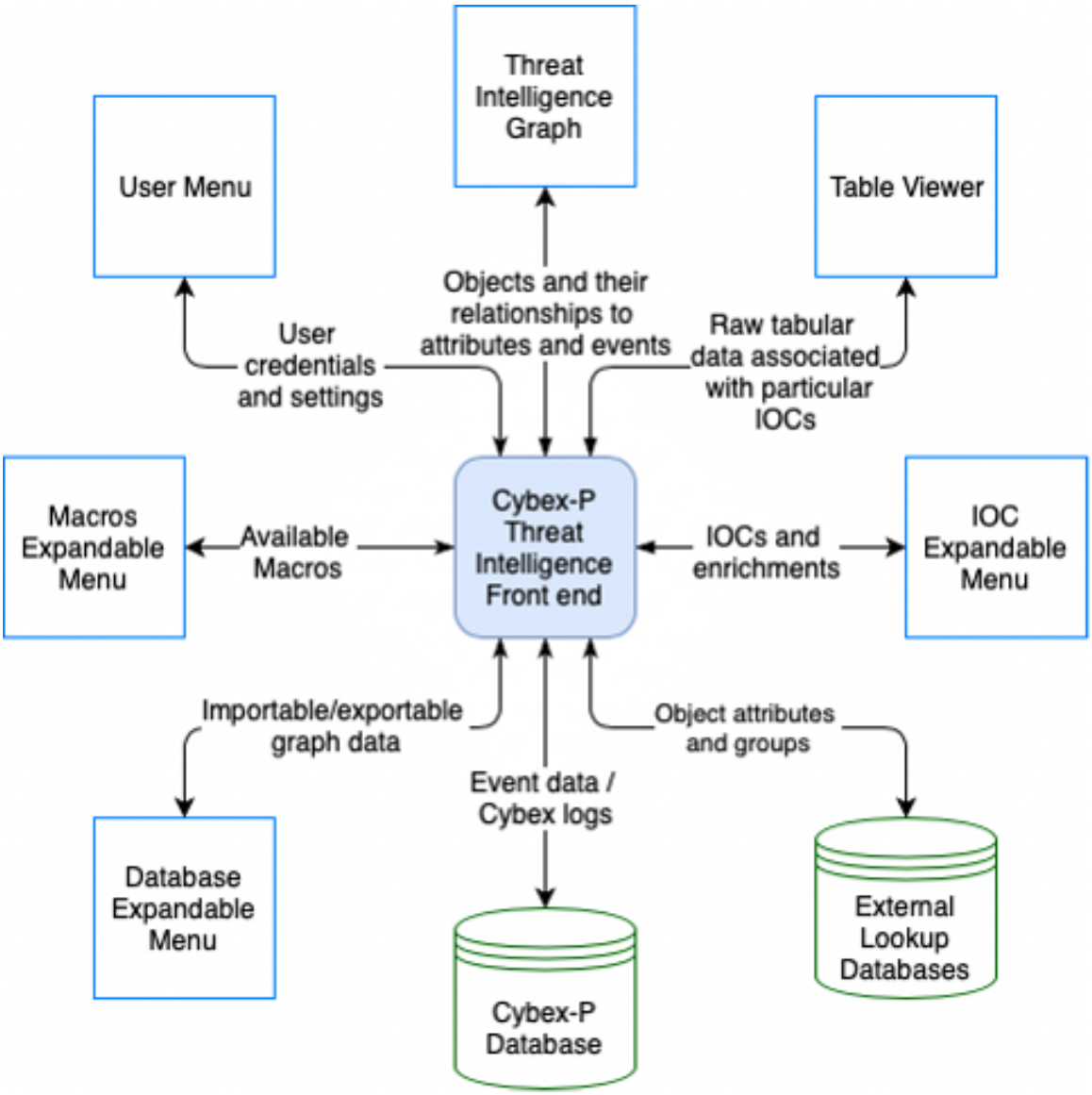


Figure 5.3: System context diagram illustrating the high-level structure of some of the application's interactions.

Django's built in user model (implemented using SQLite [43]) for user authentication and other user data. The application's templates describe instructions for generating and displaying both static and dynamic HTML output. Views represents the user interface that is displayed in the web page. They are responsible for taking data from the model and rendering it into templates. The views for this application are constructed using a number of other technologies that are detailed later in the next section.

This application extends Django with the Django REST framework, which is used to serve the RESTful API that the front end will make requests to [13]. Anytime frontend users perform an action that requires data to be queried, the back end application is responsible for handling these requests. The back end may then need to make subsequent requests to other services to gather the required data. This includes third-party APIs like Whois, GeoIP, and more. These lookups power the standard enrichments that are detailed in Chapter 6. Similar requests are made to the CYBEX-P API for specialized enrichments that reveal threat contexts. Once the Django application processes these data, it passes them back as responses to the frontend so that the results can be interacted with.

As mentioned previously, the Neo4j graph database must also be communicated with. Unlike the prior examples, this is not an API request to an external service. The Neo4j database exists on the same server as the web application back end. This database directly represents the data of the user's current graph. Separate instances of this database are isolated within Docker containers for each unique user [14]. This allows the application to save and persist every user's graph data in between usage sessions. Direct queries are made to the current user's Neo4j database within their corresponding container. These queries either modify the database or simply return its current graph representation so that it can be processed and rendered in the web application client. Figure 5.4 visualizes the architecture described in this section. See Figure 2.5 for a summary of the web application's position within the larger CYBEX-P platform architecture.

5.2.2 Back End Deployment

The Django application is deployed on a web server that allows it to be accessed publicly. The application is executed using a Gunicorn application server. This is a HTTP web server gateway interface that allows the Django application to be run as multiple concurrent python processes [19]. Such a configuration is important because several users may make requests to the system at the same time. A NGINX web server is also used to put the Gunicorn process behind a reverse proxy. This allows the application to be executed on the server's localhost, disconnected from direct outside requests. Instead, NGINX takes the responsibility of fielding requests from clients, serving as a middleman that then directs requests to the local application. This approach has multiple advantages, allowing load balancing of client requests and increasing security for the backend services [31]. Lastly, Supervisor is used to control and monitor the Gunicorn process [44]. This is used to start, stop, and reload the application. Supervisor constantly monitors process status and can restart the application if it crashes or if the server gets rebooted. A summary of the aforementioned deployment details is depicted in Figure 5.4.

5.2.3 Front End

The frontend of the web application consists of two parts. The first is a homepage that is rendered statically using HTML5/CSS. The homepage consists of mulitple static HTML files that are served as a series of templates from the Django backend. This site serves as an informational home for the CYBEX-P project, allows users to sign in with their accounts, and acts as a portal to the main application

The majority of the front end is contained within the threat intelligence graph application. This is a single-page React application built using 'create-react-app' [34]. This means that rather than having several page links to different application views, only one view is used. Components are the building blocks of the user interface that form this single page. Components are defined by a collection of logic and visual properties that together serve some functional purpose. For example, the navbar at

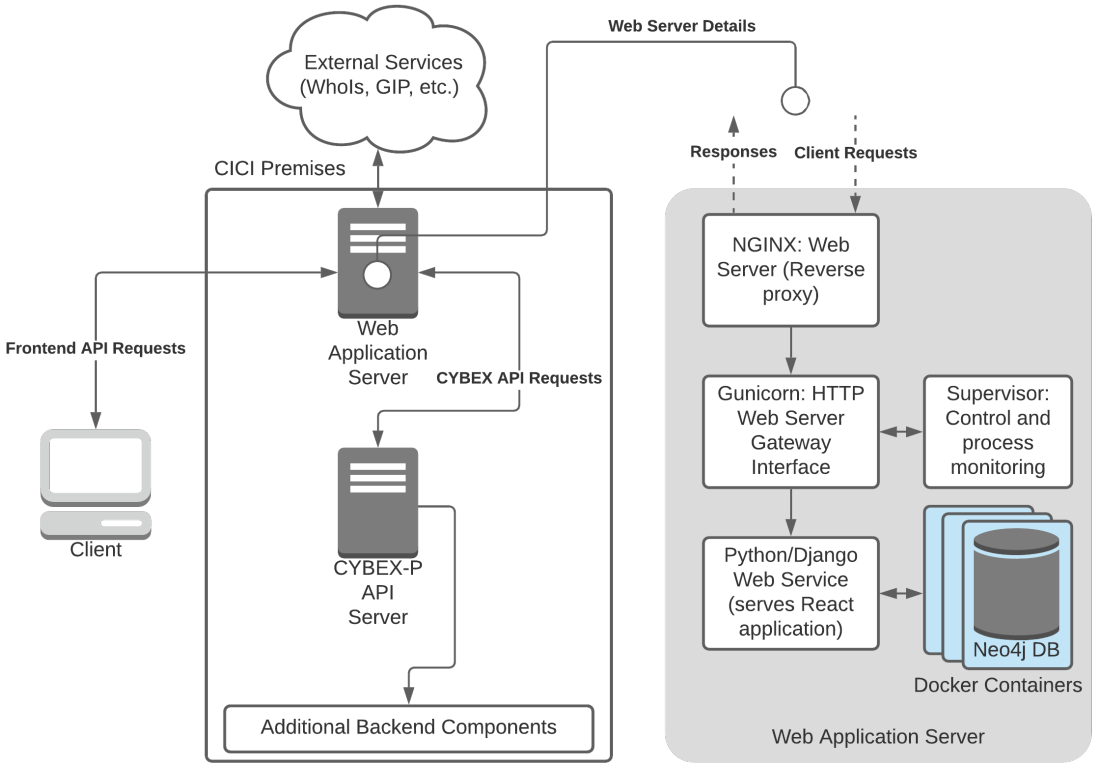


Figure 5.4: High-level architectural diagram for the CYBEX-P web application.

the top of the application facilitates a common set of actions, so it is designed as a component. Within it, there can be multiple child components, such as the main dropdown menu that is available to users. The navbar itself is a child of a master component which contains the entire application. The combined functionalities of these components are demonstrated in Chapter 6.

React can implement either class-based or functional components. The CYBEX-P web application opts for functional components that conditionally render their contents. This means that the UI will automatically and selectively render whenever application states change. State is an important React concept that describes the data components use. This data changes over time, and thus is referred to as 'changing state'. An example of a state variable could be the value of a dropdown, which changes when the user selects a new choice. The UI or backend logic may be expected to react or change according to this state change. Automatic rendering refers to the fact that

the programmer doesn't define 'how' the components should visually update; rather, they just define 'what' the UI should look like in different scenarios. Conditional rendering means that UI will only re-render if state data it is directly attached to changes. This is far more efficient that re-rendering the entire page if just one value in one sub-component gets updated.

With few exceptions, most components are custom-built rather than using off-the-shelf component libraries. This includes writing custom JavaScript functions for component logic, as well as defining custom CSS to achieve the desired visual design. All project components are written using JSX, which extends standard JavaScript. JSX is designed to be used declaratively to describe the desired appearance of the UI. It can be thought of as incorporating HTML directly into JavaScript logic, and can be incredibly efficient for rapid UI iteration. All components in the CYBEX-P web application are written in separate JSX files that isolate their respective logic and styling. This pattern of development seeks to keep all aspects of components localized. This is different than alternative methods which abstract styling and logic away from the UI elements they describe. It is worth noting that React doesn't enforce any one particular design pattern (such as Model-View-Controller (MVC) or Model-View-ViewModel (MVVM)); however, the general approaches described in this section are standard best practice.

The central graph component is one of the more complex pieces of the application. To help implement visual graph representations, the vis.js graphics library [46] was used as a foundation. In its default form, this library was not sufficient for achieving the project's usability goals. As a result, the threat-intelligence graph customized the base functionality and appearance of this tool. The tailored integration of vis.js into this application had several benefits. It allowed the backend data and frontend visuals to have a one-to-one relationship, where data is represented in a graph data structure in both the Neo4j database and React code. The result is efficient visualization and modification of graph data, with changes concurrently reflecting in users' browsers and the backend database.

5.3 Implementation Challenges

With the main goals, intended users, and requirements all defined, implementation of the web application could begin. The transition from this initial specification and design into a functional featureset presented many challenges. To solve them, it was beneficial to frame all requirements in the context of the primary usability goals. From Chapter 4, the first high-level goal was to employ graph visualization to communicate threats to specified assets. The second was to provide an effective investigation workflow centered on the graph. In this section, these goals are distilled into some guiding concepts and design principles. The resulting ideas served as important references whenever individual use cases were being translated into working features. With a large number of interactive features in the application, it was important that they feel intuitive and familiar to each other. Rather than implementing requirements in isolation, related features needed to have common design languages according to their purpose or context.

5.3.1 Graph-Based Data Visualization

Before a user can manipulate data stored within CYBEX-P, objects and threat information must be presented effectively. This includes all indicators of compromise (IOCs), such as IP addresses, domains, and email addresses, that a user wants to investigate. IOCs, their attributes, and relationships can all be represented by a network graph representation. The first Human-Computer Interaction challenge lies in visualizing the graph in a way that maximizes its static informational effectiveness. At a glance, the user should be able to determine the following about each IOC that is displayed: identifying metadata, prominence, threat level, and entity relationships.

Identifying Metadata Each IOC is unique and must be communicated accordingly. Security analysts benefit when minimal manipulation is required to view identifying information. There are three layers of identity visualization for IOCs displayed throughout the graph: type classification, source classification, and value.

Type classification simply refers to showing the user what type of IOC (IP, URL, email, and more) a particular node refers to. The solution was to superimpose different icons on different types of IOC nodes, as shown in Figure 5.5. Doing so helps analysts better identify the origin of entity relationships during an investigation. Other candidate designs were considered that used color-coding, rather than iconography, to differentiate IOC types. However, the node color property ended up being chosen to communicate threat levels, so this approach was discarded.

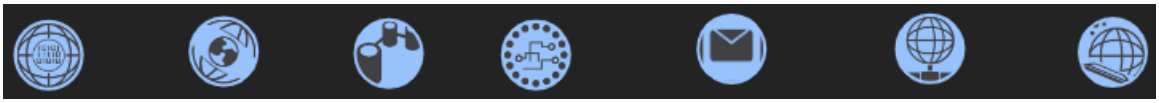


Figure 5.5: Node icons differentiate IOC/attribute types. From left to right, these icons represent addresses (IPs and URLs), countries of origin, ASN groups, files, email addresses, hosts, and registrars. These are just a subset of the full icon set.

For source classification, the goal was to communicate which mechanism added a node to the graph. IOCs can be added to the graph directly through user input, or connected through 'enrichments' (specialized queries that connect new data to the graph). There are two distinct classes of enrichments, explained thoroughly in Chapter 6. The first category is used for building and understanding network topologies. The second is used to add objects that are related according to CYBEX-P's crowdsourced event data. It is desirable to visually differentiate which class of enrichment led to a node being added. Originally, different node shapes were used to identify their source. However, this proved inadequate; sometimes the same node could end up being related by both a standard network relationship and a related CYBEX event. It was therefore decided to keep node shape neutral for this property. The focus switched to graph edges, which naturally signify relationship information. The final solution was to modify edges' line strokes according to source. Nodes added from standard lookups have solid lines between them and the originating node. Those added from CYBEX lookups use dashed lines. This a simple property that is extremely effective at making this information glanceable.

Lastly, all IOCs have text-based values that identify them. These take the form

of IP addresses, subnet IDs, country names, or other values. Identifiers should be presented only once per graph, avoiding any duplication. The graph presentation implies an object-oriented organization; thus, it is crucial that each node is truly unique. This promotes confidence in users who can trust that all important information about a node is always going to be centered about one location. Multiple designs were considered for identifying metadata. As shown in Chapter 6, identifiers were ultimately implemented as simple text labels that float below each node.

Prominence and Threat Level One of the most important conclusions a security analyst can make is whether or not a particular entity is a threat. CYBEX-P provides data points that can help users estimate the threat level of an IOC. The database keeps track of the total number of times an entity is observed within a benign event context or a malicious event context. Two resulting insights can be communicated to the user from these data points: IOC prominence and threat level.

IOC prominence refers to its total number of sightings within CYBEX-P's event database (the sum of both benign and malicious counts). Some straightforward mechanism was required that allows users to quickly see the influence of various actors according to this value. The more data on an IOC that exists in the system, the more confidently its threat level can be inferred. This was visualized by scaling node sizes directly according to their number of sightings. IOCs that have a large number of sightings with a high threat level should be considered verifiably extreme threats.

The CYBEX-P threat level of an IOC is a calculated ratio of malicious occurrences to total occurrences. This value needed to be visualized through a highly noticeable and readily-available mechanism. It was desirable for clusters of malicious entities to be made immediately apparent (i.e. surrounding relationships to the malicious node are as important to highlight as the node itself). The key challenge was to use node-level visualization tactics to build a categorical cyberthreat heatmap. The implementation needed to serve as a trustworthy visual guide at all levels and scopes of the graph. A color-driven, 'stoplight' design was ultimately converged upon and is

presented in Chapter 6.

Entity Relationships According to the gathered requirements, investigators needed to see which IOCs/attributes relate to each other in the graph. In its simplest form, this is accomplished by drawing graph edges between nodes.

When observing large-scale graphs, it was also desirable for relationship-based node clustering to form. This would allow users to begin investigations at an overview level. Then, they could focus in on particular clusters, omitting others and improving efficiency. Thus, it was a key priority to explore how clustered layouts could be implemented. An example of the desired behavior is shown in 5.6.

The motivation for visual clustering can be explained by using binary search trees as a metaphor. In binary search trees, the algorithm looks for one value or 'leaf' node in the tree. To arrive at the correct node, the tree is sorted such that all items to the 'left' of a node are less than the current node's value. Likewise, all the values to the 'right' of a node are greater. With this assumption, the search for a given value can start at the root node of the tree. Half of the tree can then be eliminated from the search space at each step until the desired node is reached. This is a popular data structure in computer science due to its favorable O(log(N)) time complexity (compared to the O(N) time complexity required to traverse each node in an unsorted tree). This efficient approach is the inspiration for using node relationships to visually 'cluster' nodes in the graph. The idea is that a malicious node should quickly draw attention to itself and the immediately surrounding IOCs. In a large graph, this may bring the search space down to a small subset of its original clusters. Entire benign clusters can be deprioritized while the investigator thoroughly analyzes the more suspicious ones. Clustering behavior is dictated by the graph's physics system, which is detailed later in this chapter.

The other value of this approach is that, like with binary search trees, all original value is preserved in the data structure. No data is removed from the graph; clustering simply improves visual search efficiency. So, if someone is prioritizing different data

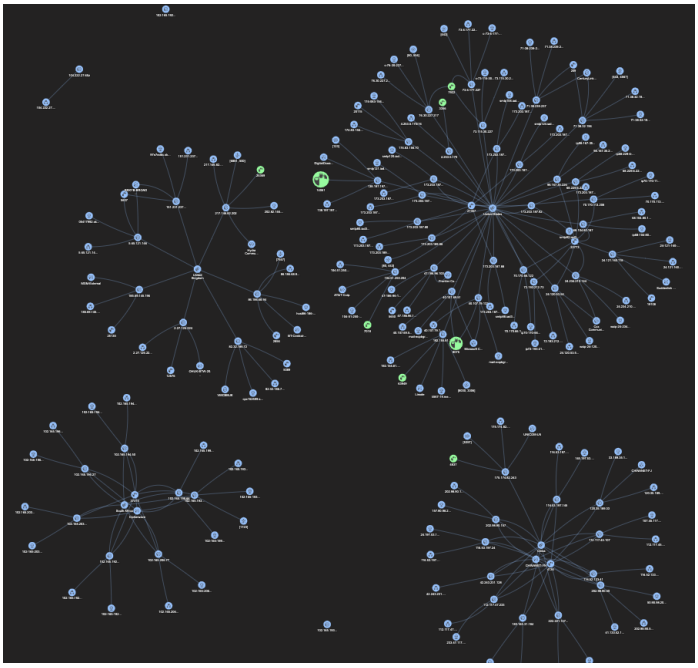


Figure 5.6: An example of a graph that has stabilized into four primary clusters.

(i.e. something other than malicious nodes), the exact same graph can be used to find that information too.

A key goal for this tool is to effectively integrate Schneiderman's visualization mantra [38]. The first step of the mantra is to provide an "overview first". The visual should next allow you to "zoom and filter". Lastly, details should be made available "on demand". The aforementioned clustered graph visualization helps satisfy the first two of those three points. The third point, "details on demand", is best achieved through effective graph interaction. That implementation challenge is covered in the following discussion on investigation workflows.

5.3.2 Interactive Investigation Workflow

In addition to providing effective static graph visualization, it is necessary for CTI Graph to afford direct interaction to facilitate investigations. User interactions have two primary goals: data manipulation and focused data observation.

Figure 5.7: One effective solution for node-localized functionality is the use of radial menu UI components.

Data Manipulation Users must be able to add and remove data nodes from the graph before analysis can occur. Certain operations are performed directly through interacting with the graph, which will be explored later in this work. Others are handled using secondary, dynamic menus adjacent to the main graph interface. The primary needs of each menu will now be detailed.

First, a user needs to be able to add various IOC types. When a user selects an IOC type and assigns a value, that node needs to be added to the central canvas. Another function to be facilitated with direct interaction is node enrichment. Enrichments are the specialized queries that retrieve other objects or attributes related to a node. These items should be immediately linked to the calling node and become part of the graph. A key design challenge was building a contextual menu system for these enrichments that is also localized. This way, when a user selects a node, its available actions are immediately presented precisely at that node's graph location. The purpose of this approach is to keep primary user actions centered around their graph, rather than directing their attention to disconnected interface elements. This design, pictured in Fig. 5.7 reduces the number of visual traversals a user must make, improving investigative efficiency.

Similarly, an additional menu was needed that could be devoted to macros. Macros are stored procedures that contain a list of IOC enrichments. When executed, each macro performs these operations on the entire graph and all of its nodes.

This allows users to perform multiple enrichments at scale much more quickly than manual execution. While more straightforward than the other interface elements, the macro menu still had some notable design challenges. It required a design that balanced utility with informational context. Providing a convenient list of macro options in a small amount of space was important to reserve screen real-estate for the graph. To solve this, helpful information about each macro needed to be optionally toggled in an unobtrusive manner. The second challenge was due to the nature of how macros process. Because they are a collection of enrichments that may take several minutes to compute, it is important for users to get visual feedback of processing status. Returning and rendering query results in batches was a desired solution. This meant that the graph shouldn't wait for the entire macro to finish processing before displaying new graph connections. Instead, users would prefer to work with data as soon as it is available, increasing the usability of the graph while in loading states. The implemented solutions to these challenges are presented in Chapter 6.

Focused Data Observation The second major purpose of graph interaction is focused data observation, which exemplifies Schneiderman's concept of "details on demand". [38]. The word "demand" in this phrase refers to some decisive user request. The CYBEX-P web application must intuitively translate these user requests to the precisely desired visual output. There are a number of forms that these interactions can take. Some intend to manipulate the graph canvas view. Others are direct actions on graph nodes and edges. All of these actions are crucial to accomplishing the application's graph-centered design goals.

Canvas manipulation is desired for quick adjustments to investigation scope. It should take as few steps as possible to go from an overview state to a more focused, zoomed-in view. Malleability is also a desired graph characteristic. This means that users should be able to easily re-position all graph elements to their liking. No matter which of the above actions occur, the graph should always render in the most useful and visually optimal way for its currently displayed scope. Like with node clustering,

this manipulation challenge was solved with the physics model defined later in this section. See Chapter 6 for a walk-through of the canvas modification features that were designed to meet these needs.

Meaningful threat conclusions require a full set of information to be available for every individual node and edge. This includes the full identifying text for each item, the exact breakdown of its threat statistics, any investigator comments, and other specific details. It would be far too visually cluttered to render all of these details near each and every graph item. Instead, the application's design aimed to only reveal full details for one or two nodes/edges at a time.

The layout for presenting these details was designed using a "baseball game" metaphor. When one traditionally watches a baseball game, only basic information about each player is immediately obvious as they compete on the field. Each player wears a uniform that associates them with basic identifying information (such as team and player number). This uniform is directly worn by the player, so those details are directly tied to their position in space. Viewers can therefore easily comprehend the basic information of a game as it unfolds in real-time. There is no need, for example, for a commentator to consult a player photo-to-number database as they announce plays. However, there are situations where more details are desired. If a fan wants to learn about a player's statistics and game history, they may buy baseball cards containing that extra information. These details are meant to be read at one's leisure, absorbed separately from the main action of the game. Both the uniform and baseball card concepts are core to CYBEX-P's detail visualization model. They represent distinct contexts and use-cases for on-demand information.

Following the baseball game model, two view levels were desired for revealing network and security details. The first was a minimally-intrusive state that shows basic and universally useful information about a node or edge (such as identifier or threat summary). The second was a view that could represent 100% of the available information about an item. An important design determination was how to handle the permanence of each of these views. It was concluded that the more lightweight

view would be mostly used in temporary "quick-glance" situations. Conversely, the full-detail view needed to anticipate more complex and lengthy observations. The former is analogous to baseball player uniforms; the latter is like studying baseball cards. There are two user actions that map well to these types of pointed information requests: mouse hover and clicks events. Mouse hovers are inherently more suitable for temporary selections. It is generally difficult for users to keep their cursor steady in the same precise location for long periods of time. Thus, this action was mapped to the simpler summary view. Tool-tips were used to display this basic data "in-place" as the user's cursor moves across each graph element. With mouse clicks, users can make a selection and then feel free to move their cursor anywhere else on-screen. The selection will still persist until the next item is clicked, providing freedom to perform other actions in between. The longer view intervals provided by these interactions were natural fits for the "full detail" states. Comprehensive information of selected items were structured into "IOC cards". These are placed in the corner of the screen away from the main graph, making them both functionally and visually similar to the baseball cards they are inspired by. A full demonstration of these concepts in action are presented in Chapter 6.

Graph Physics To achieve a graph experience that was truly dynamic, arbitrary and static node positioning would not suffice. If nodes simply persisted wherever they were placed on the canvas, the connections between them would eventually become difficult to follow. Adding more nodes may cause their connected edges to cross over graph elements in arbitrary paths, and they may have wildly variable lengths. While a graph can be considered 'correct' as long as the accurately properly connects nodes and edges, bad positioning can be difficult to interpret. Without interpretability, the graph has no value, and the application will not be trusted for critical use. This is why a graph layout policy was desired that converges on consistent visual patterns. This policy needed to also be reactive to graph modification and users' direct manipulation. These requirements resulted in the adoption of a physics-based layout model that gives

each graph element a sense of interactive realism.

The Barnes Hut approximation method was chosen to implement the required behavior. This is a hierarchical O(N log N) force-calculation algorithm proposed by Josh Barnes and Piet Hut in 1986 [2]. It was designed to address the N-body problem, which is the need to compute forces that result from pairwise interactions among a set of points [21]. Conceptually, graph nodes can be thought of as charged particles that repel each other. Conversely, edges are analogous to springs that pull nodes together. A brute-force approach may try to calculate the force contributions of the interactions of each and every node pair. This would would require an unacceptable quadratic running time ($O(N^2)$ time complexity).

The solution contributed by Barnes and Hut was to estimate forces by substituting groups of points with their center of mass. While some small error is introduced, this approach improves time complexity to O(N log N). The next problem is to determine which nodes should be grouped together for these calculations. A 'spatial index' is needed to facilitate these groupings. The quadtree data structure is ideal for this utility, and recursively subdivides square areas of space into four smaller squares, until each node resides in its own cell. The center of mass is then computed for each cell of the quadtree. This is equivalent to the weighted average of its four child cells' centers of mass. After these initial calculations, force estimation can be implemented. For closer nodes, the center of mass for smaller cells are used; for more distant nodes, larger cells are used.

To classify 'close' versus 'distant' points, a parameter θ is used. This is the ratio between the distance of a cell and the cell's width. If θ for a sample point and a particular cell is less than a selected threshold, then that cell's center of mass is used in force calculations. Otherwise, the algorithm recursively traverses that cell's child cells, until the chosen θ value is satisfied. Figure 5.8 illustrates how quadtree cells are recursively formed, with the center of cell masses being selectively used for force calculations on an example point.

For CYBEX-P's graph visualization, vis.js was used to implement this Barnes

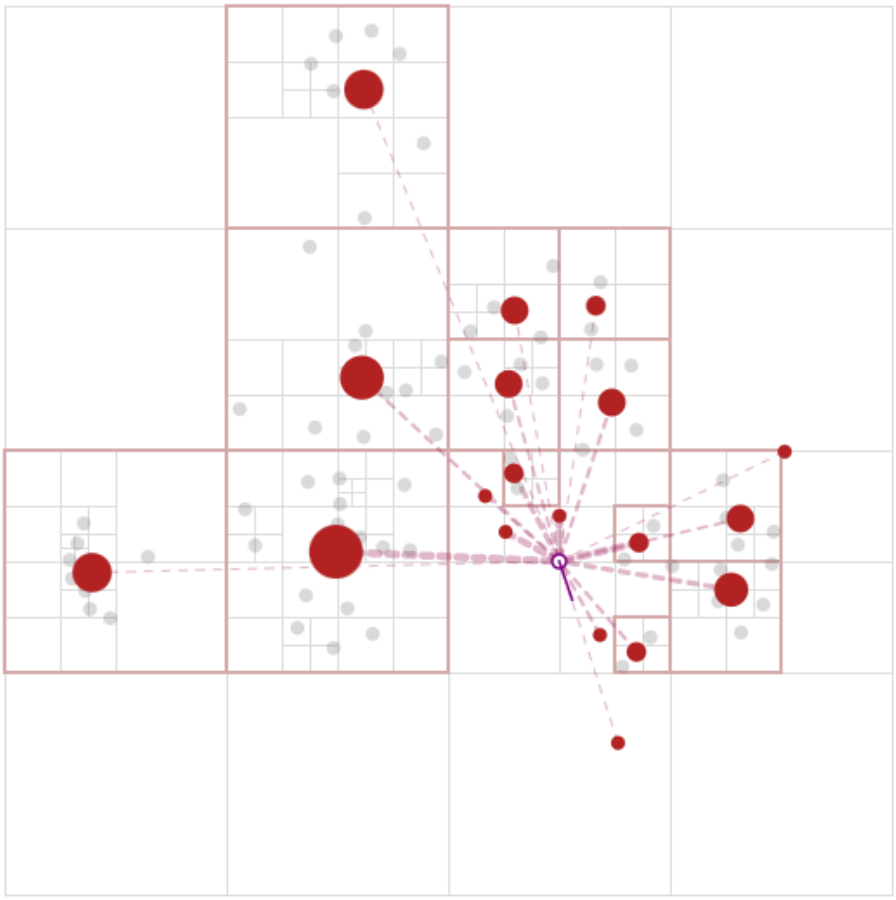


Figure 5.8: Image taken from an interactive explanation of the Barnes Hut approximation method for network graphs [21]. Grey circles are the original nodes shown in two-dimensional space. The red circles are the centers of quadtree cell mass used for force calculations. A sample point is outlined in purple. Dashed lines represent the repulsive forces between the sample point and the other centers of mass.

Hut model. A force strength of -2000 was chosen, as strong negative forces keeps nodes from visually crowding each other (negative = repulsive). θ was originally chosen to be 0.5. After assessing trade-offs between positional accuracy and run times with different θ values, a value of 1 was chosen instead. This improved graph rendering performance, as increasing θ decreases the total point-to-point computations that must be made. Placement accuracy was still acceptable, as ultra-precision was not a requirement. In addition to these core parameters, a central gravity attractor was used with a value of 0.3 to keep the graph pulled towards the center of the canvas. Spring length, representing the resting length of graph edges, was set to 95. The 'sturdiness' of edges, called their spring constant, was 0.04. The damping factor is the amount velocity from previous simulation iterations that contribute to subsequent ones. This was set to 0.09.

Using this model, users can efficiently visualize graphs with hundreds of nodes. When constructing or manipulating the graph, the model provides satisfying and realistic interactivity. The repulsive forces, governed by the constraints described in this section, facilitate an easily navigable graph layout. This layout is immediately reactive, and naturally forms easily perceptible clusters. Rendering performance is optimized, allowing for a smooth user experience. The result is a live, robust structure for hosting data relationships.

Chapter 6

Prototype in Action

6.1 Interface Overview

This Chapter provides a tour of the CYBEX-P web application. There are many components to the final product, and together they satisfy all of the requirements and usability goals described in prior chapters. First, an overview of the user interface is provided and the specific utility of each part is examined in detail. This Chapter concludes with comprehensive usage scenarios from the perspective of intended users.

Figures 6.1 and 6.2 depict the CYBEX-P homepage. This is a public-facing web page that serves two main purposes. First, it acts as an informational hub for the entire CYBEX-P project. Site content includes feature descriptions, screenshots, and videos for the threat-intelligence graph application. General information about CYBEX-P, such as the project abstract, funding information, and personnel, are also provided. Users are kept up to date with industry trends through cybersecurity-related twitter feeds. CYBEX-P's latest platform and software updates are also presented in this location. The second purpose of this site is to act as a 'portal' to the frontend web application described throughout this work. This homepage is used to sign in or out of the application with two-factor authentication. Then, by selecting the "CTIGraph" button, users are redirected to the main threat-intelligence graph canvas.

The starting point for the main application is depicted in Figure 6.3. The user is greeted with an empty canvas, unless they have any previous graph data auto-saved.

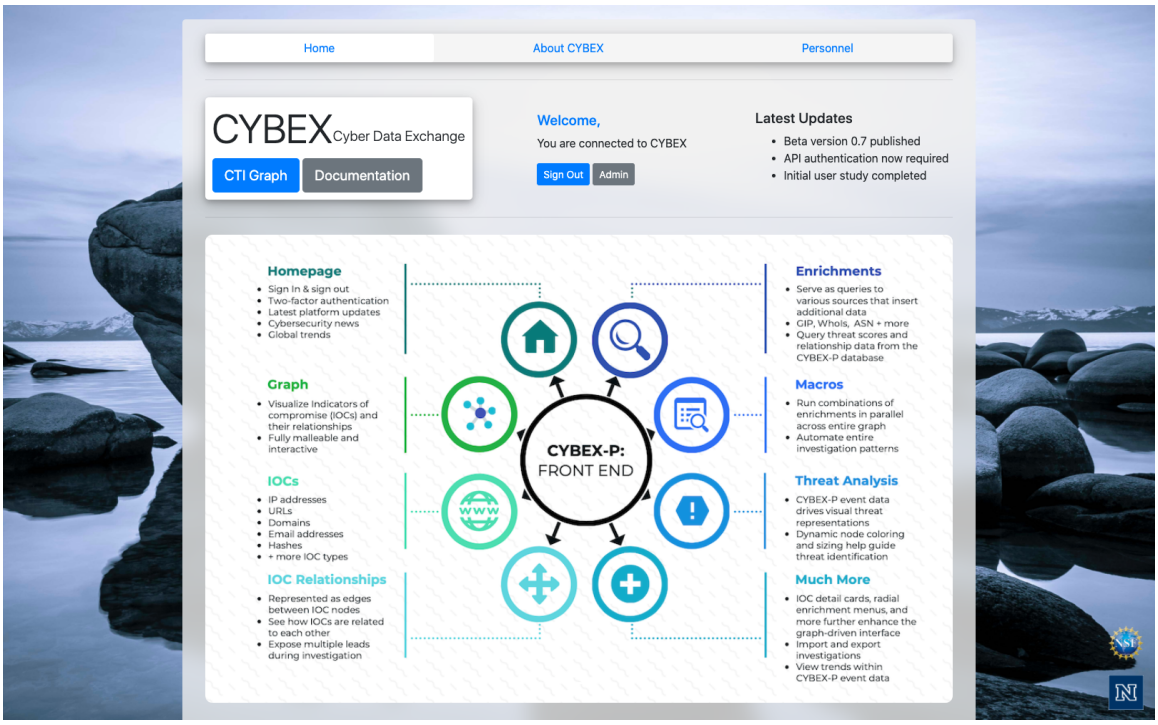


Figure 6.1: Public-facing homepage for the CYBEX-P project.

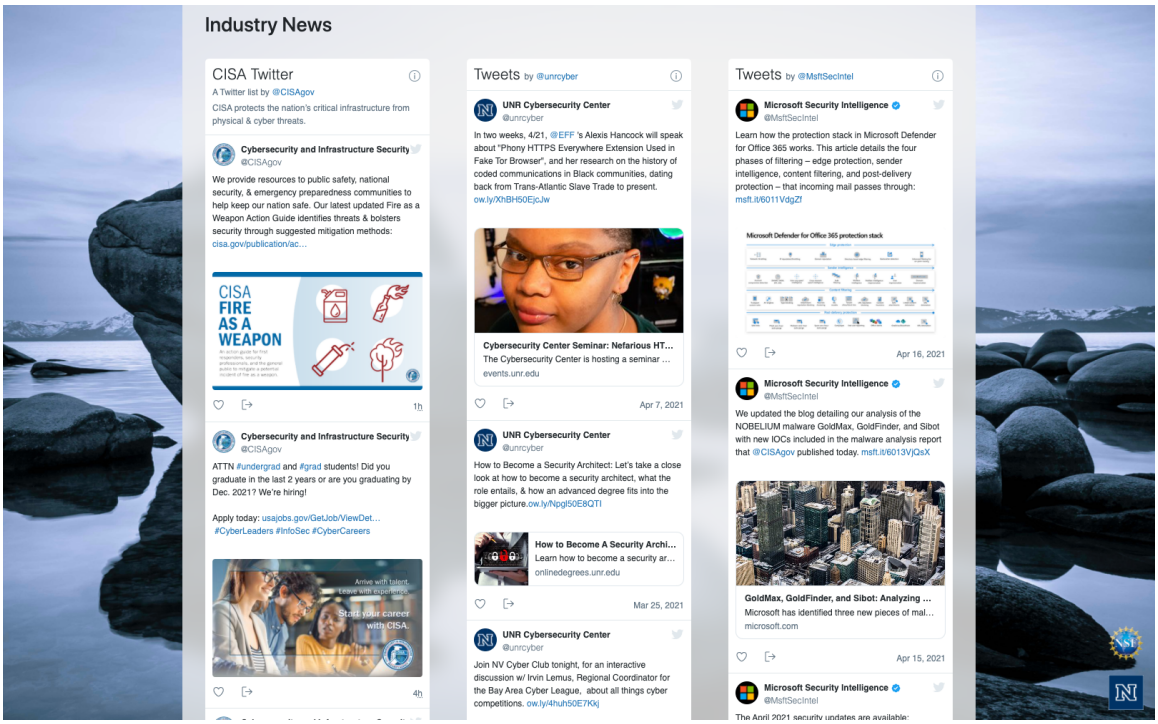


Figure 6.2: Cybersecurity-related news feeds are displayed on the home page.

In such cases, the user's data is persisted from prior sessions and loaded in exactly how they left it. The overall design uses a 'dark' theme, reducing eye strain and emphasizing colors. All peripheral menus and controls are accessible from the perimeter of the canvas. The first class of these controls exist within the top navigation bar. This includes access to functions that are not directly-graph related. Included in these features are links to other CYBEX website pages, user profile information, administrative panel, and a supplemental trends page. These items are depicted later in this section. The second class of controls are those that are deeply related to graph construction and analysis. These are placed on the left, right, and bottom edges of the screen. The three main expandable menus are signified by blue tabs, each with a unique icon. Each tab affords the on-demand display of their associated menus. All three are depicted in their expanded states in Figure 6.4.

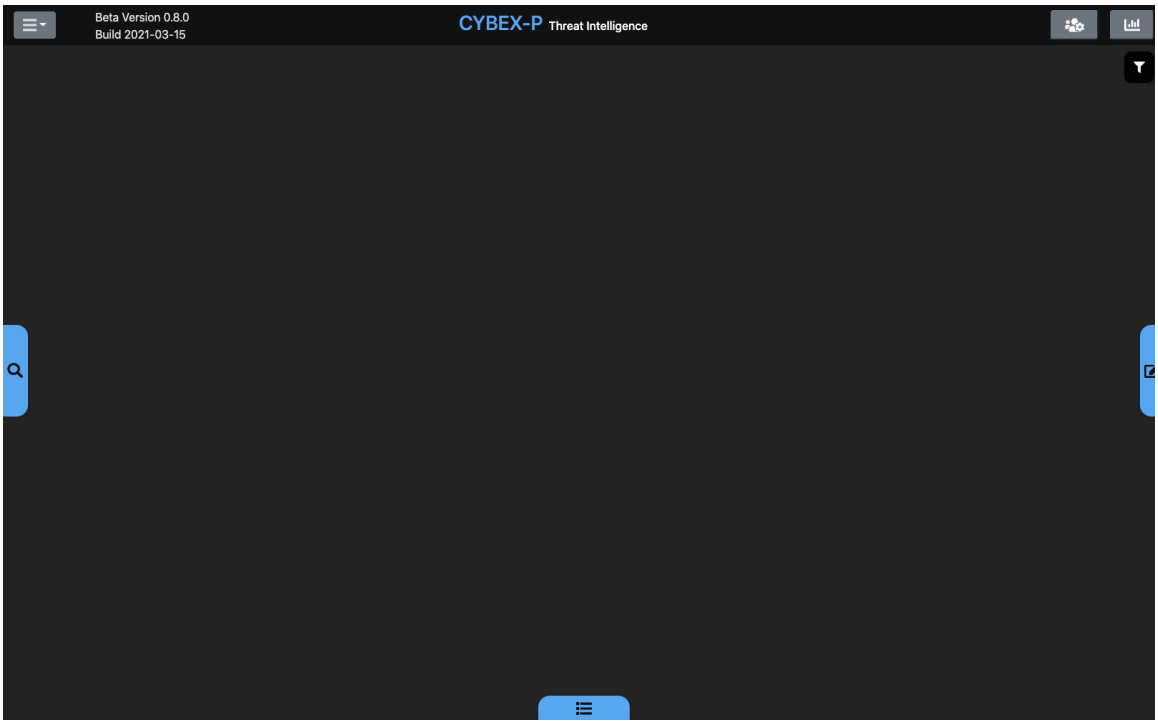
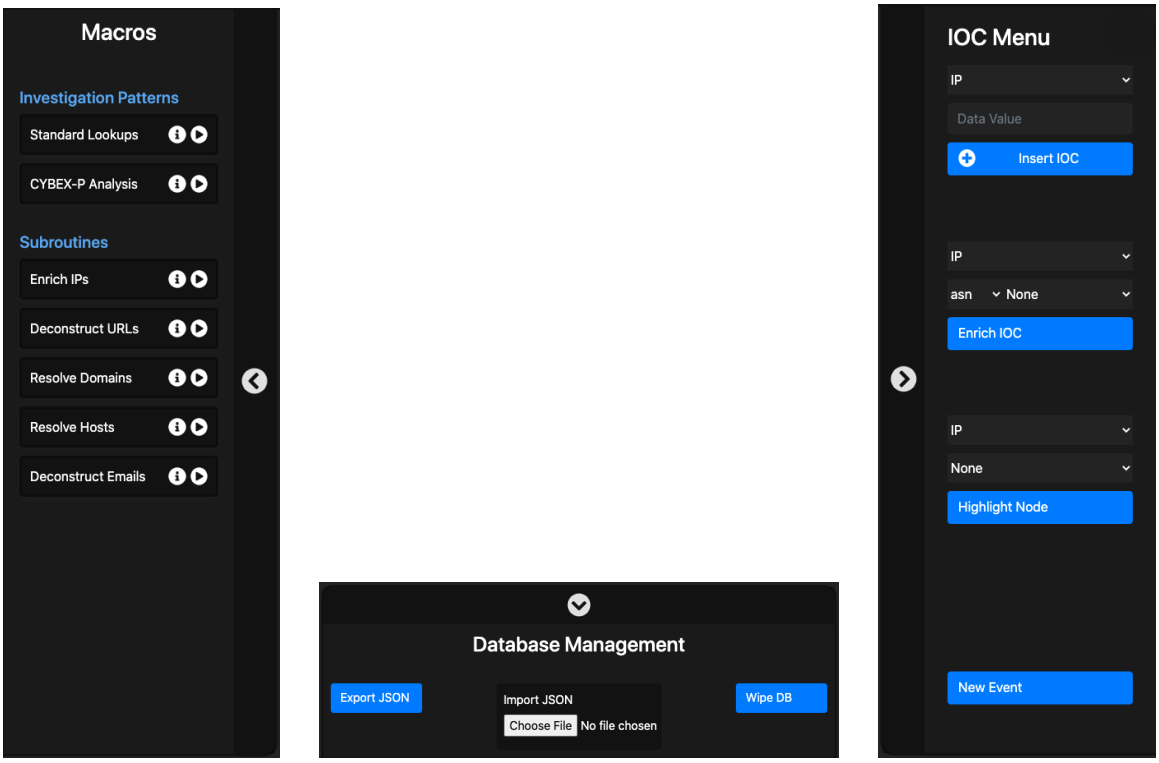


Figure 6.3: CYBEX-P threat intelligence graph interface (empty canvas).

Figure 6.4(c) shows the IOC Menu, which groups together IOC-specific functions. The first thing it affords is adding new IOCs to the graph canvas. Users select from a full list of IOC types (IPs, URLs, email addresses, and more), type the value of



(a) Macro menu (b) Database management menu (c) IOC Menu

Figure 6.4: Three expandable menus containing graph-related functionality.

that item, then click the insert button. Once nodes are added to the graph, the next group of controls in this menu can be used to execute enrichments on them. The list of available enrichments varies depending on the type of IOC selected. Examples for different IOC types are given in Table 6.1. Descriptions of what each enrichment does are listed in Table 6.2.

Figure 6.4(a) depicts the Macro menu. Macros apply enrichments across every node of the graph at once. There are two categories: investigation patterns and subroutines. The former combines multiple enrichment types together at once, while the latter executes a single enrichment type across the graph. For example, the 'Standard Lookups' investigation pattern includes deconstruction of email addresses (extracting the domain names), but it also performs a number of other things. If the user only wants to deconstruct email addresses, they can run the associated subroutine instead of the full investigation pattern. To get the full list of enrichments

IOC Type	Supported Enrichments
IP Address	asn, gip, hostname, whois, netblock, ports, cybexRelated, cybexCount
URL	deconstructURL, cybexRelated, cybexCount
Domain	whois, resolveHost, nameservers, registrar, mailservers, cybexRelated, cybexCount
Host	whois, resolveHost, nameservers, registrar, mailservers, cybexRelated, cybexCount
Email	deconstructEmail, cybexRelated, cybexCount
Hash	cybexRelated, cybexCount
ASN	cybexRelated, cybexCount

Table 6.1: Supported enrichments for some example IOC types.

Enrichment	Description
asn	Returns the IP's Autonomous System Number (ASN).
gip	Performs a GeoIP lookup to return the country of origin.
hostname	Uses address to return the associated hostname.
whois	Returns owner information from Whois record.
netblock	Returns subnet value.
ports	Performs a Shodan lookup to return a list of open ports.
cybexRelated	Queries the CYBEX-P database for IOCs that are related by common events.
cybexCount	Queries the CYBEX-P database for counts of the IOC's benign and malicious sightings.
deconstructURL	Retrieves the domain name from a particular URL.
deconstructEmail	Retrieves the user and domain name from an email address.
resolveHost	Uses hostname to return associated host address.
nameservers	Returns nameservers.
registrar	Returns registrar from Whois record.
mailservers	Returns mailservers.

Table 6.2: Descriptions of each supported enrichment.

each macro includes, users can click its information button. An example of this on-demand information panel is shown in Figure 6.5. This panel is a semi-transparent and easily-closed popup, reducing the amount of graph area it obstructs.

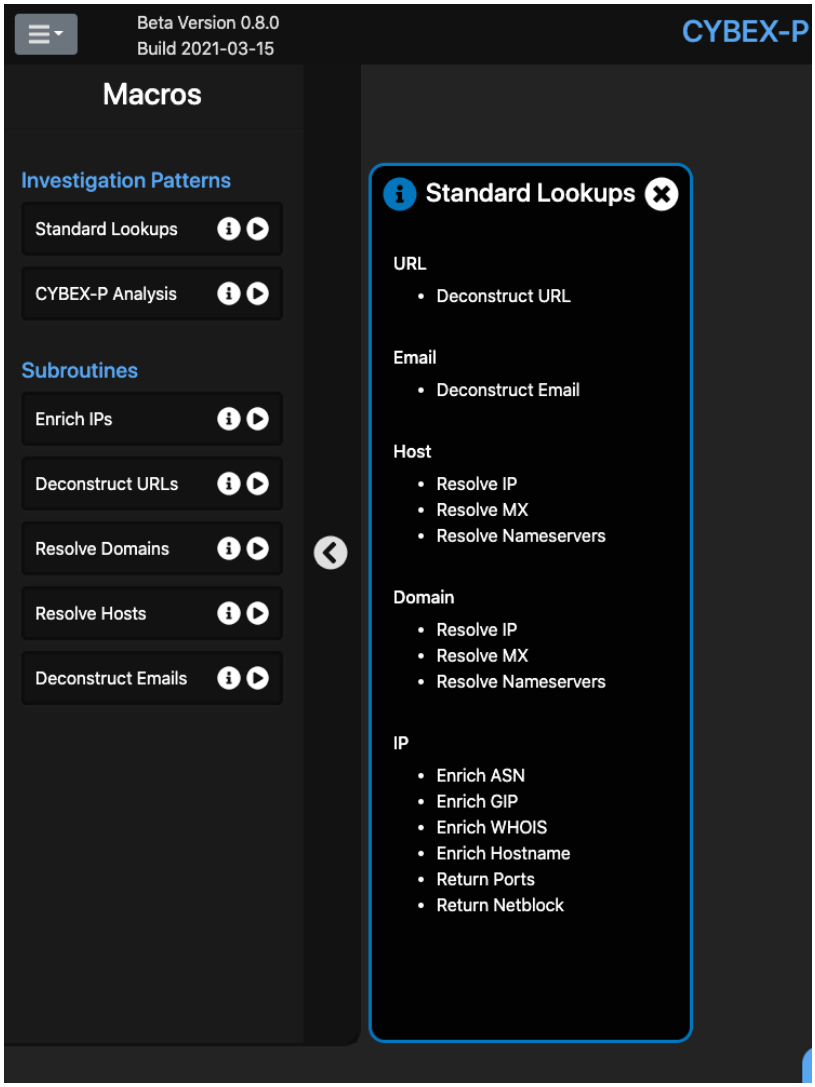


Figure 6.5: Information pop-up for macros is minimally-invasive due to its on-demand nature and semi-transparent background.

The bottom menu, shown in 6.4(b), is for database management. This refers to the graph database that is created for each user (see Chapter 5). Users can export their current graph data as a JSON file, load a new graph from an imported JSON file, or wipe the database for a fresh canvas.

Figure 6.6 shows how nodes appear once they've been added with the IOC menu.

When hovering over each node, a tooltip displays its IOC type and value (see Figure 6.7). When selected, a radial menu renders around the node containing all available enrichments for that node. As demonstrated in Figure 6.8, an informational card is also displayed in the bottom-right portion of the screen. This contains a more complete list of information about the node, as well as some auxiliary options like node commenting. Figure 6.9 shows the result of a GIP enrichment being run from the radial menu. In this case, a single country node is added and connected to the graph. The edge between the two nodes is rendered with an arrow, pointing from the originating node to the new node. This directionality sometimes provides helpful historical context. The arrows allow an investigator to retrace steps, revealing the order enrichments were run. In addition, relationships can sometimes be very directional. For example, if a GIP lookup is run to get country of origin on an IP, the arrow will point from IP to country. The IP is located in the country, not the other way around. In these cases, arrows and descriptive tooltips on edges establish the flow of information.

Figure 6.10 shows a macro in action. Specifically, this is an example of running the 'CYBEX-P Analysis' macro on a single IP address (0.0.0.3). This macro does two things. First, it runs cybexRelated enrichments on all graph nodes. Second, it performs threat analysis of all graph data. When the macro runs, it returns the results of the first step before the second step is done. That way, users can use the new data without waiting for all analysis to complete. Once the second step ends, threat scores are added to each node and the macro is finished. Because macros can have large execution times on large graphs, this sequential approach is key for constant usability.

The resulting graph from a completed CYBEX-P Analysis run is shown in Figure 6.11. Each node's threat score is the percentage of 'malicious' events in which that item was seen (displayed when hovering over analyzed nodes). This is a simple calculation using the 'benign' and 'malicious' sighting counts from the CYBEX-P database. Using this percentage, a categorical threat ranking is assigned to each IOC.

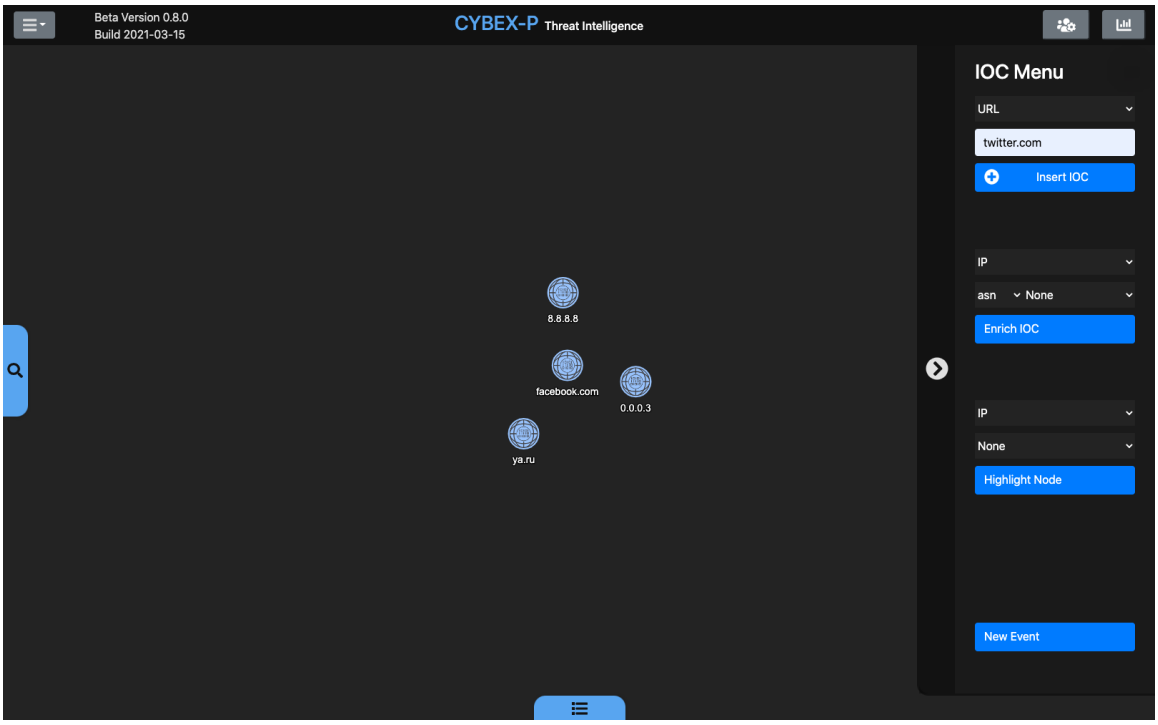


Figure 6.6: IOC menu is used to add some example IP addresses and URLs to the graph.

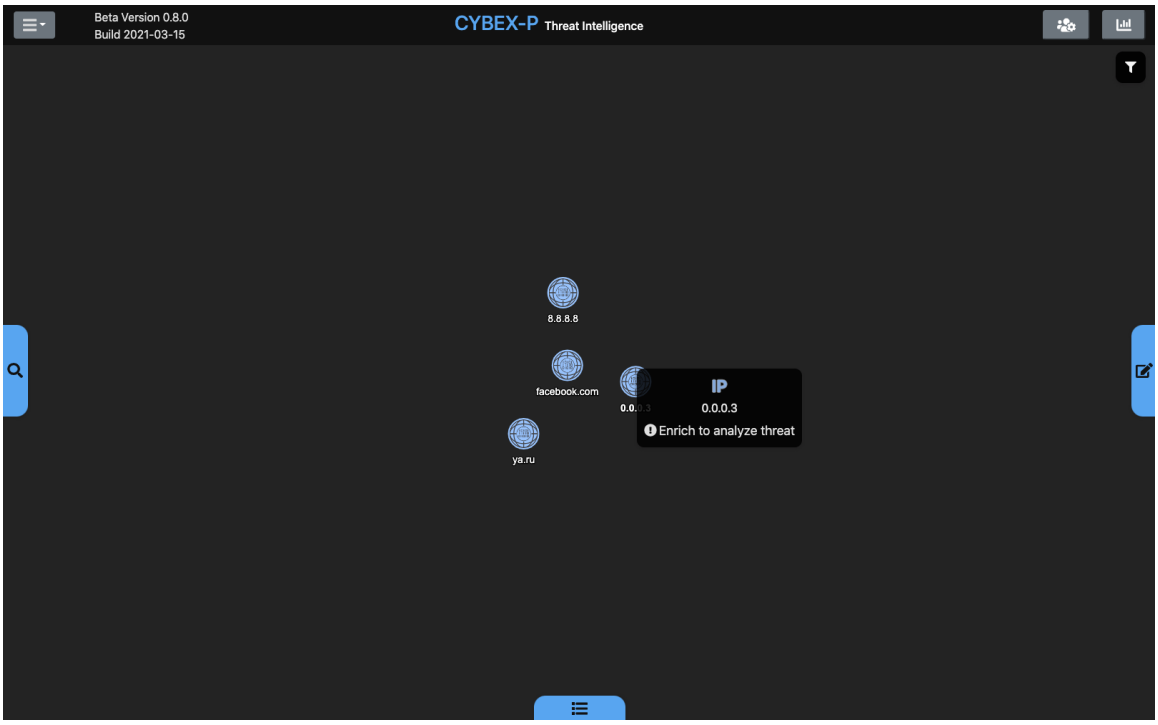


Figure 6.7: Tool-tips containing basic details about nodes are displayed upon mouse hover.

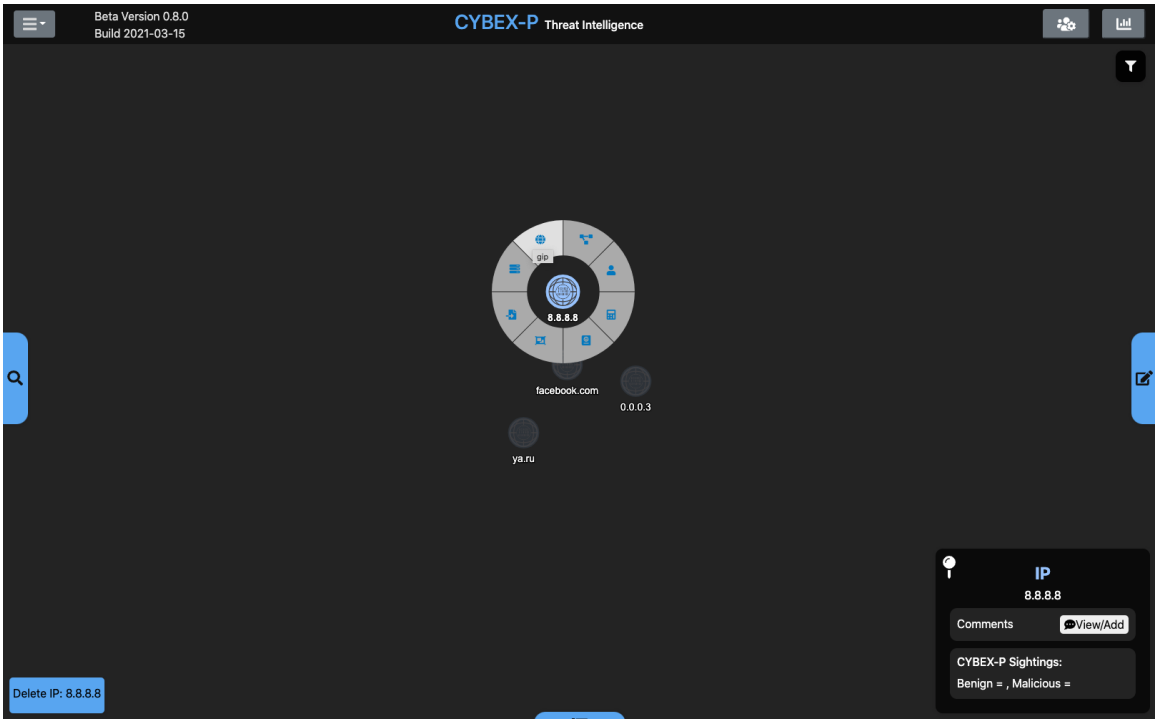


Figure 6.8: The full details and available actions for a node are displayed when clicked on.

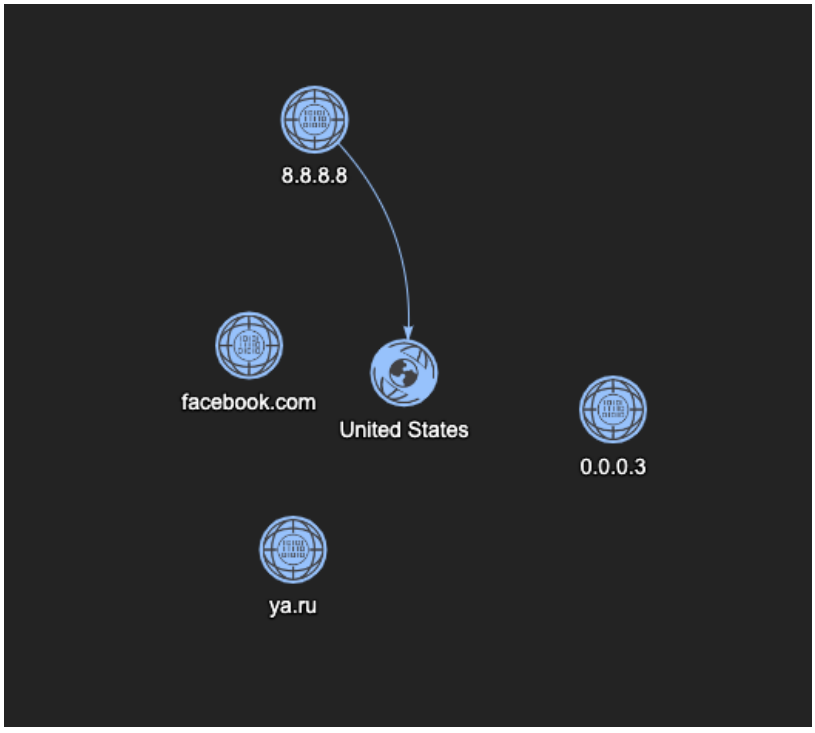


Figure 6.9: The resulting graph after running a single enrichment on a single node.

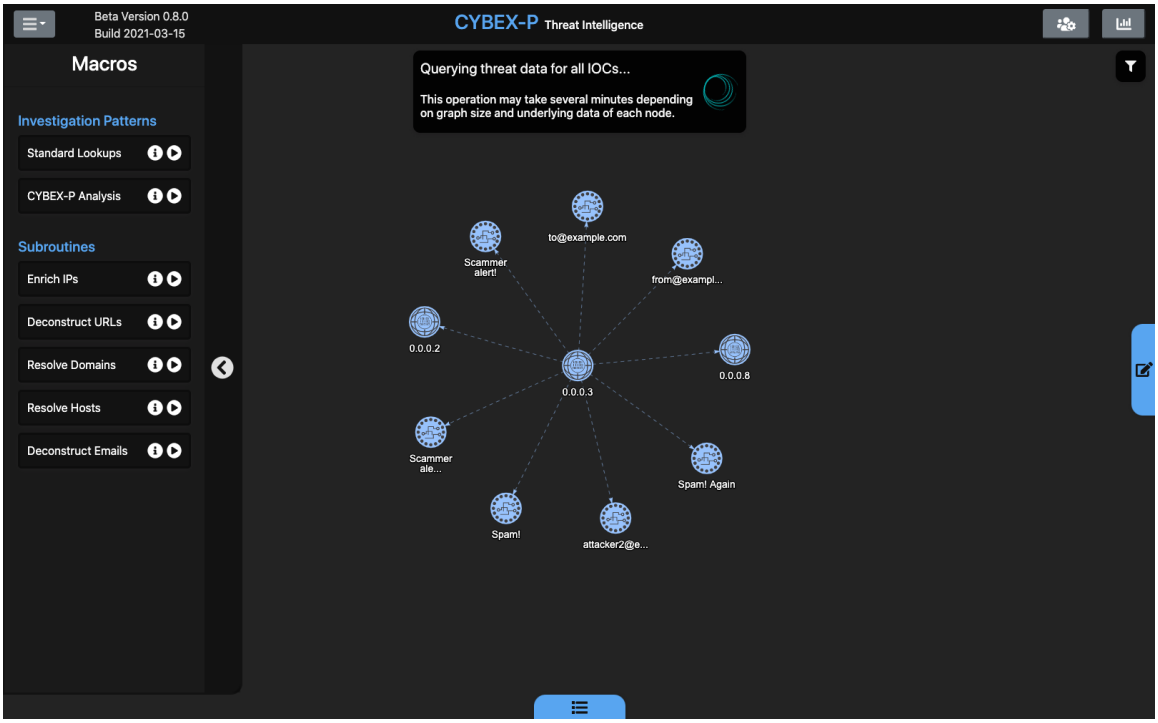


Figure 6.10: The resulting graph after the first operation (adding related IOCs) of the 'CYBEX Analysis' macro has completed.

These are 'benign', 'moderately malicious', and 'very malicious'. These are assigned node colors of green, yellow, and red, respectively. Blue nodes are the default color, and they remain blue if no threat information can be gathered. Threat isn't the only thing that is visualized in this case, however. When the count values are returned, they also are used to create a 'total sightings' value (malicious + benign sightings). Total sightings are visualized using the size of each node. A larger node equates to a larger number of overall sightings in the CYBEX-P database. This feature attempts to add useful context, which is one of the design goals discussed in Chapter 5. However, as discovered in Chapter 7, this feature may need to be adjusted to prevent analyst confusion.

CYBEX-P analysis also provides event context. When obeserving two nodes with a CYBEX relationship, their common edge can be hovered over. The resulting tool-tip goes beyond a simple textual description. Instead, it displays a chain of color-coded boxes. From one end of the chain to the other, this represents exactly how the

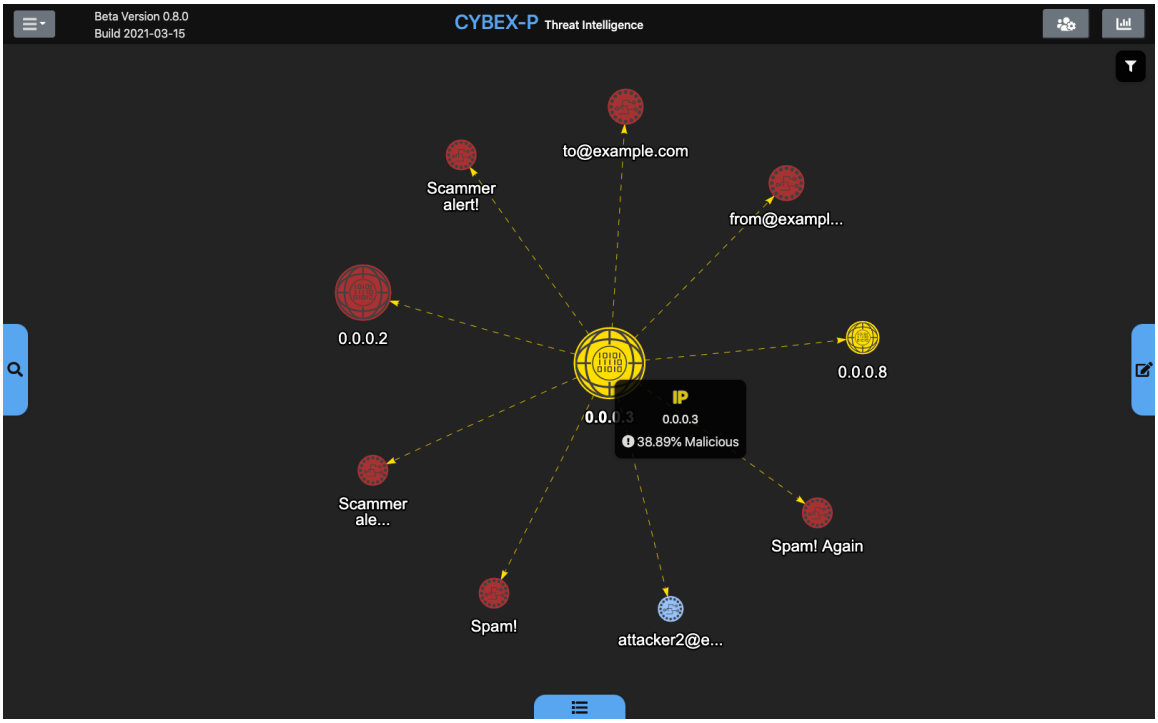


Figure 6.11: The resulting graph after the 'CYBEX Analysis' macro has performed threat analysis. A 'threat score' is given when hovering over the node.

two nodes are connected. The example shown in Figure 6.12 shows the chain between two IP addresses. Attributes are purple, objects are blue-green, and events are gold. Using these classifications requires some knowledge of how CYBEX-P tracks items in its database. The given chain shows that a destination IP (0.0.0.3) was connected to a network traffic event. Note that 'dst' (destination) is the object, with 'ipv4' being its associated attribute. Likewise, it is easy to see that a source IP was part of this same event. Serving as the common ground for these two IPs, the network event exposes this potentially unknown relationship. There are many types of events, objects, and attributes that can form these types of relationships. Dashed lines are used to distinguish CYBEX event relationships from all other enrichment types.

Users can customize CYBEX-P enrichments using the expandable filter in the top-right portion of the screen. Displayed in Figure 6.13, this component affords adding restrictions to the data that is returned. Specifically, start and end dates can be selected, according to a desired timezone. When returning related IOCs or IOC

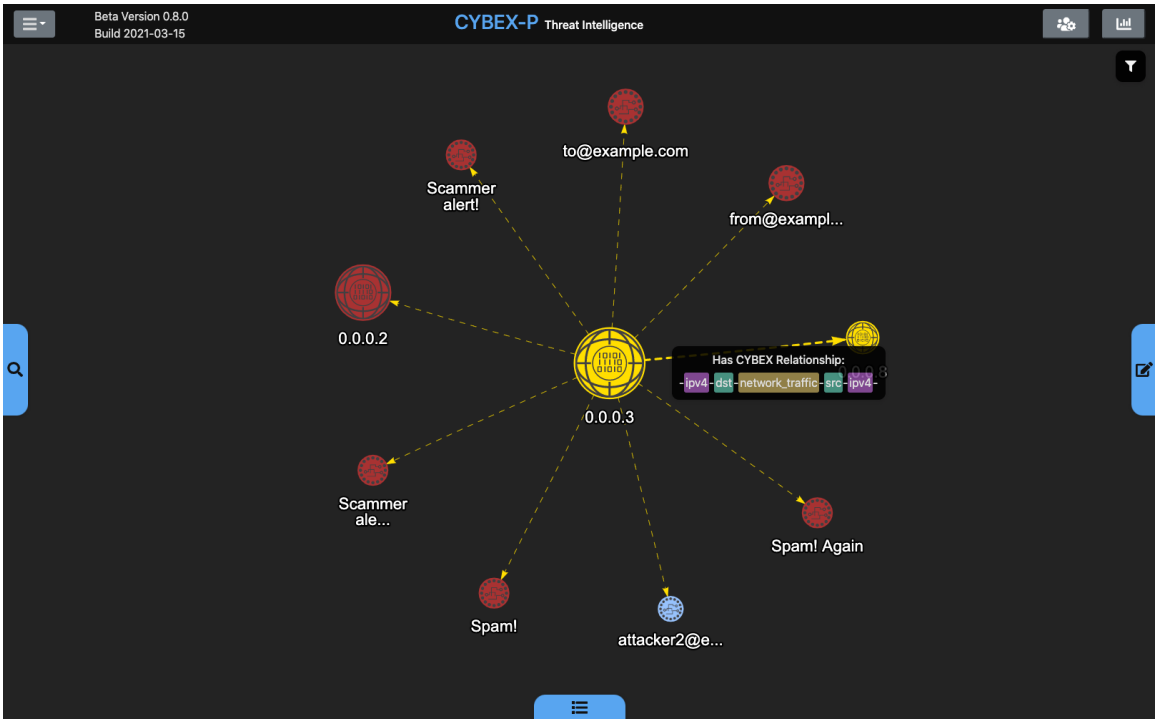


Figure 6.12: When hovering over an edge that represents a CYBEX relationship, the event data that connects the two nodes is displayed.

sightings data, only results gathered within these ranges be used. This can be helpful if an analyst wants to limit an investigation to a known timeframe, such as when some suspicious activity occured. For example, consider an IOC that is only seen in benign contexts for many years. However, it then briefly appears in several malicious events. If too large of a time range is used (such as a year), the many benign counts may outweigh the few malicious ones. This may make an analyst wrongly think the IOC is safe. Instead, if they limit the range to a particular week of interest, they will get threat classifications that are more relevant to their investigation.

Aside from the core graph features, adjacent functionality is provided in the top navbar. First, an expandable menu is depicted in Figure 6.14. This provides links that redirect to the CYBEX homepage and various documentation pages. Another option allows users to download cowrie honeypot log files that are hosted on the same server as the application. These files are purely for experimental purposes and not intended as a main feature of the application. Lastly, users can open a window for

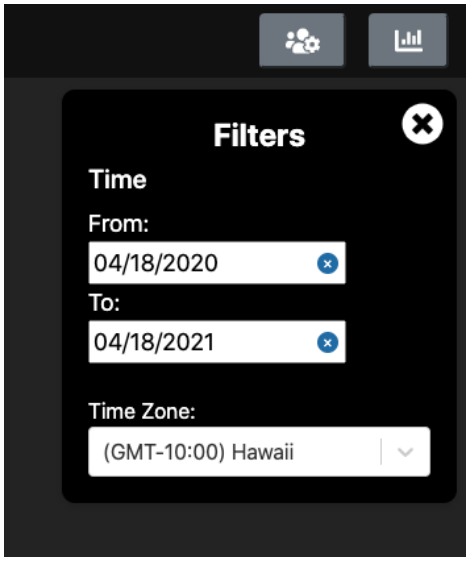


Figure 6.13: Expandable window allows users to filter CYBEX enrichments and macros to particular time ranges.

their user profile. This is depicted in 6.15 and shows the username, email address, and a unique identifier of the person who is signed in. Any organizations the user belongs to is also shown here.

There are two buttons on the rightmost part of the navbar. The first signifies administrative functions, and opens the user management window. Shown in Figure 6.16, this panel allows admins to add and remove users to their organizations. The final buton on the navbar opens the trends panel, displayed in Figure 6.17. This is a space for plots that show high-level trends on the overall CYBEX-P database. The plots shown here are just examples, but can represent things like events over time or the most prevalent event types.

The last major feature is user event data submission. Shown in Figure 6.18, users can upload supported file types. The example shown here is a JSON file from captured Cowrie honeypot logs. The user must also supply a timezone to associate the submitted event(s) with. Strict validation is built into this form. First, the application will not allow users to submit forms that have incomplete fields. If so, required fields are highlighted to users so they can fix their mistake. Next, the actual contents of the file are validated against a list of supported schemas. For example,

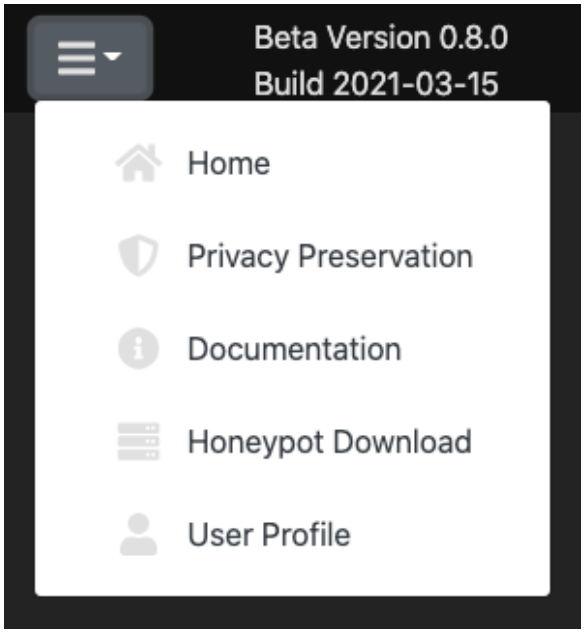


Figure 6.14: Expandable menu in top-left corner reveals some navigation options and secondary features.

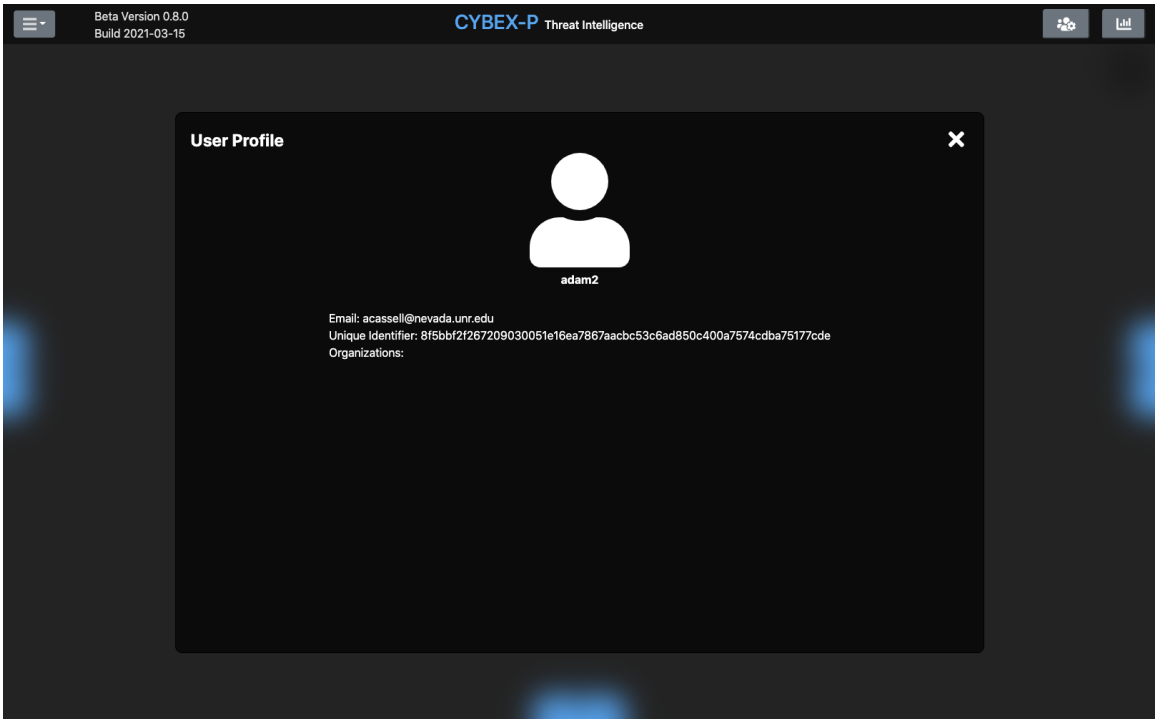


Figure 6.15: User profile panel displays basic account information about the currently signed-in user.

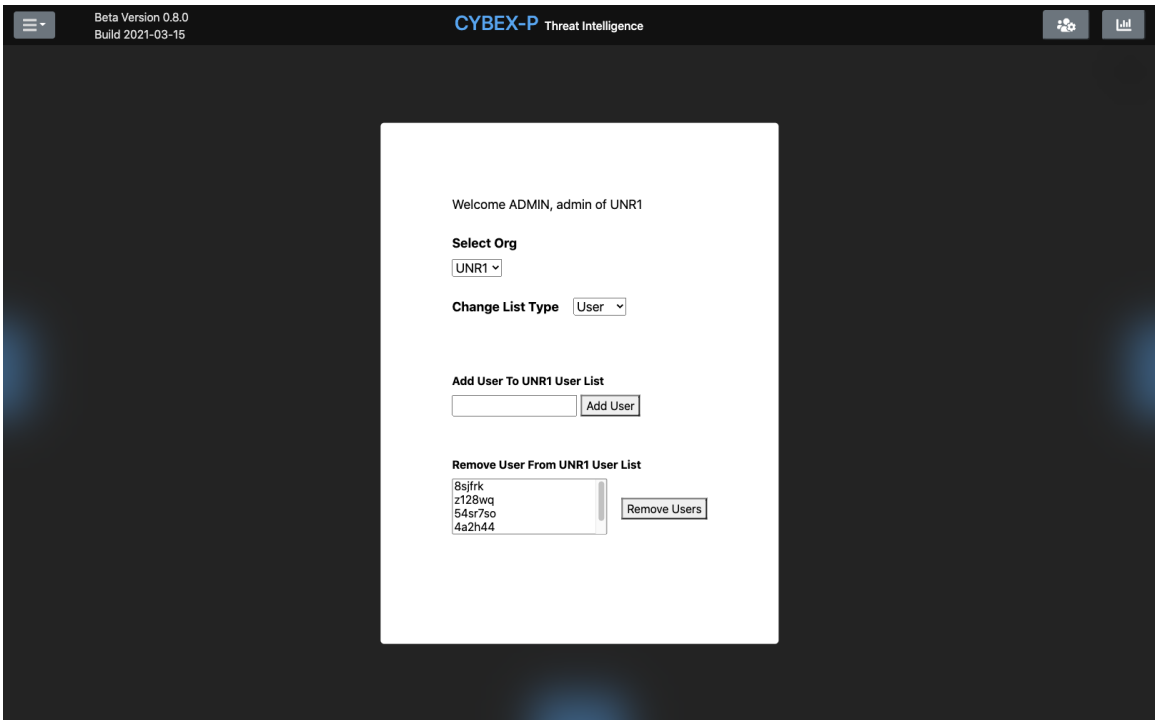


Figure 6.16: User management panel allows admins to add/remove users to their organizations' lists.

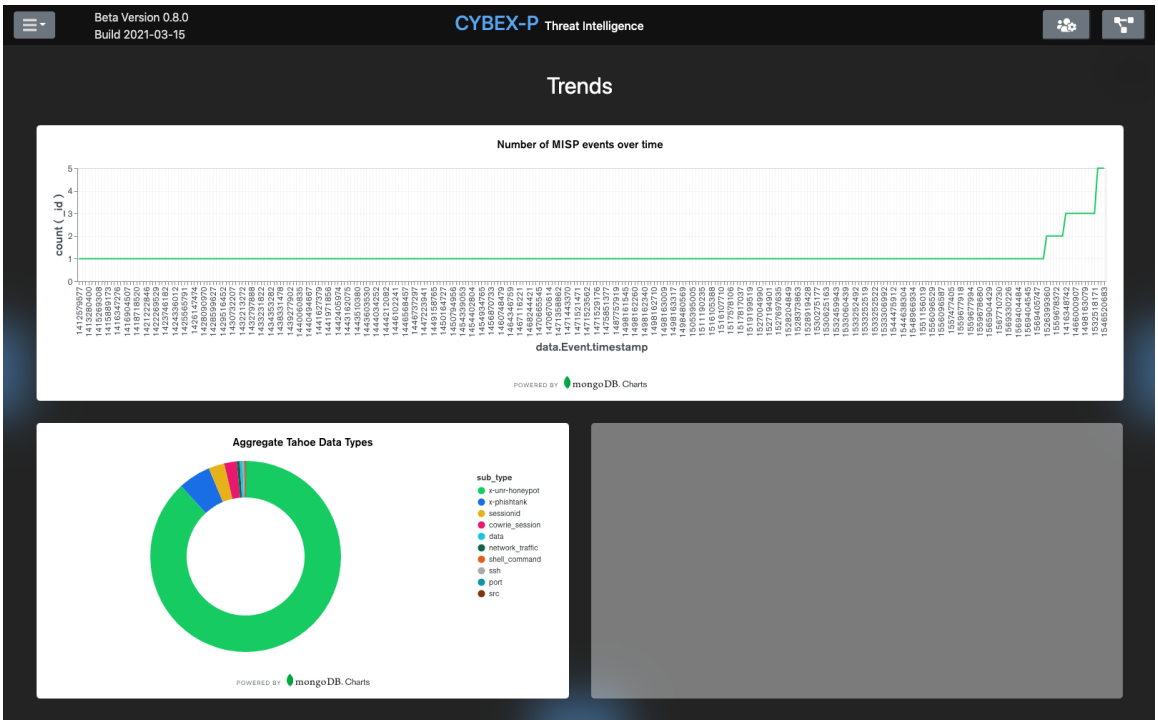


Figure 6.17: Trends panel displays high-level statistics and time-series information about CYBEX-P's data sources.

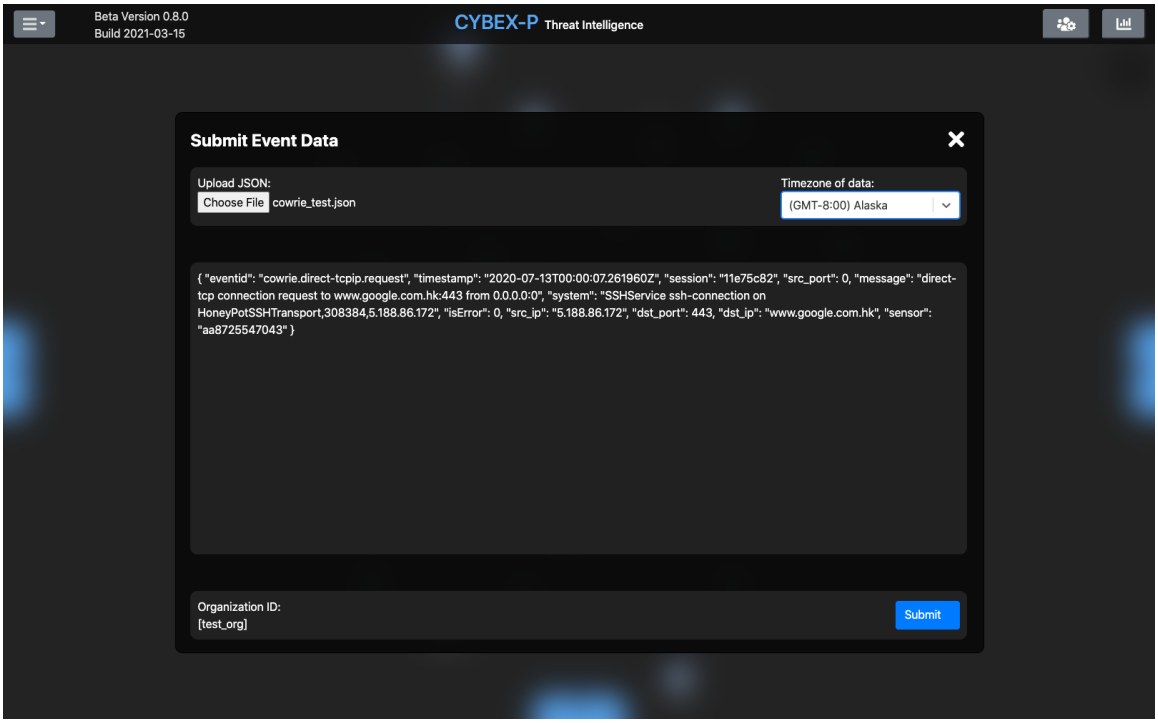


Figure 6.18: Users can upload their own event files as contributions to the shared CYBEX-P database.

Cowrie log files follow a known pattern of key/value pairs. If the data types and key types don't match exactly, the data is rejected. Users are alerted if something is wrong with their data, and can then address it. In addition, there are safeguards in the backend CYBEX-P system to prevent bad data from being introduced. These are outside the scope of this work, but they complete a robust series of validation steps that all data is subjected to.

With all the major features now introduced, it is clear that there is a wide variety of functions that the application can fan facilitate. These satisfy the requirements and intended use cases defined in Chapter 5. Figure 6.19 is an activity diagram representing behavior of the application. It shows how some key user actions drive the main features of the application. Note that this diagram was drawn during early development stages, so a few details have since changed. The following sections walk through some specific usage scenarios step-by-step, demonstrating the intended utility of the application.

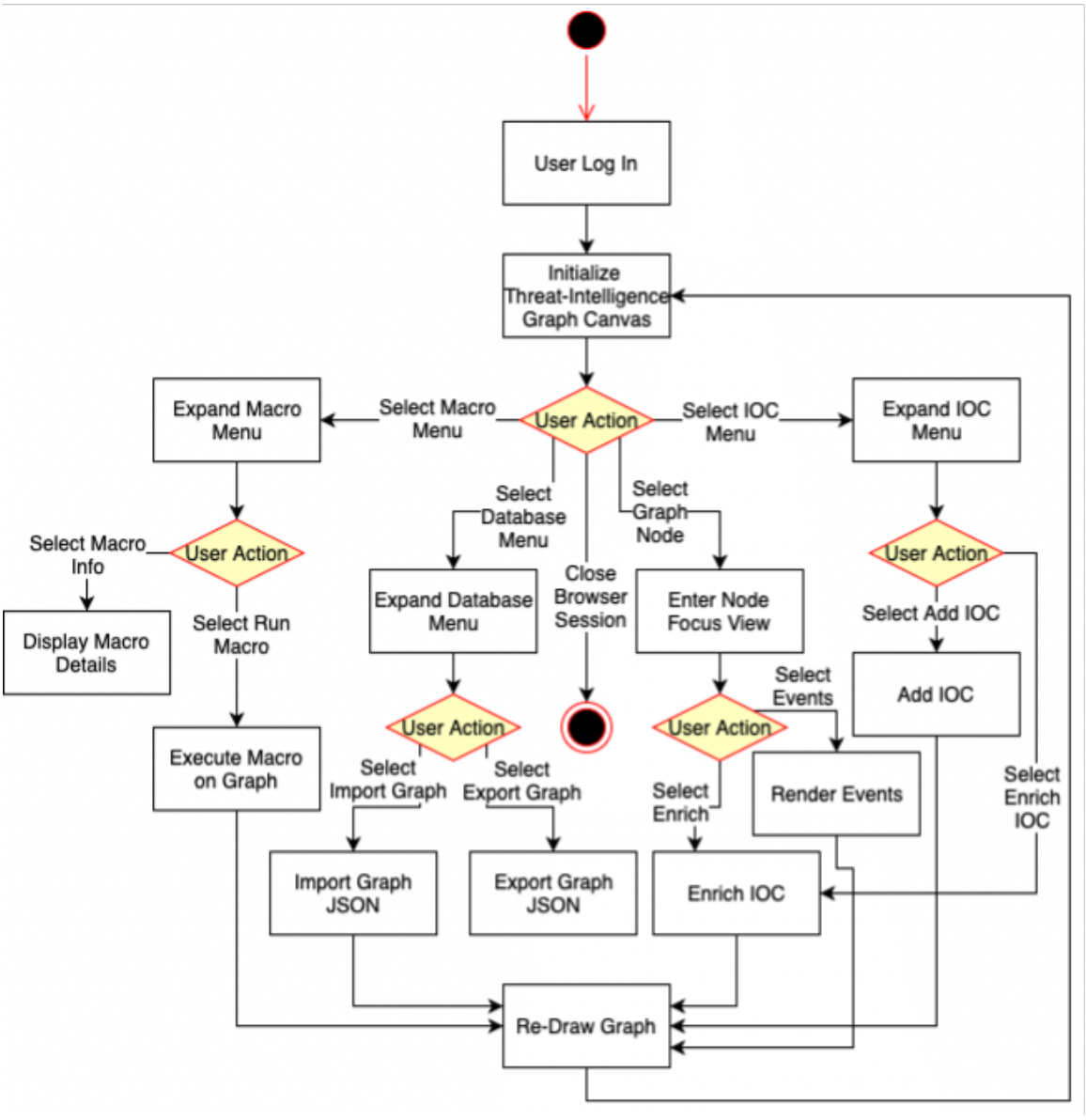


Figure 6.19: Activity diagram illustrating the expected behavior of the product.

6.2 Scenario A: Large Graph Construction

The first scenario resembles some exploratory graph construction, rather than specific analysis. This approach may be taken by analysts who simply want to get a big-picture view of a particular network graph. In this scenario, they are mostly concerned with visualizing connections between IOCs of interest and any adjacent items. This is a case where the investigator doesn't have any particular leads; rather they would simply like to picture their entire network topology.

This scenario begins with the same set of nodes pictured in Figure 6.9. These nodes are all of the known IOCs in the network the analyst monitors. From this starting point, the analyst runs the 'Standard Lookups' macro to return as much lookup data as possible for all nodes. This macro includes all of the non-CYBEX enrichments listed in Table 6.2. Figure 6.20 shows what the screen looks like while this macro executes. Figure 6.21 shows the result. The investigator decides to run this macro recursively, calling it a couple additional times. This expands the graph space with multiple layers of additional nodes, growing the scope exponentially each time. The investigator wanted to get a big-picture view of their network, so this approach helps them quickly achieve that. Figures 6.22 and 6.23 show the results of the second and third macro runs, respectively.

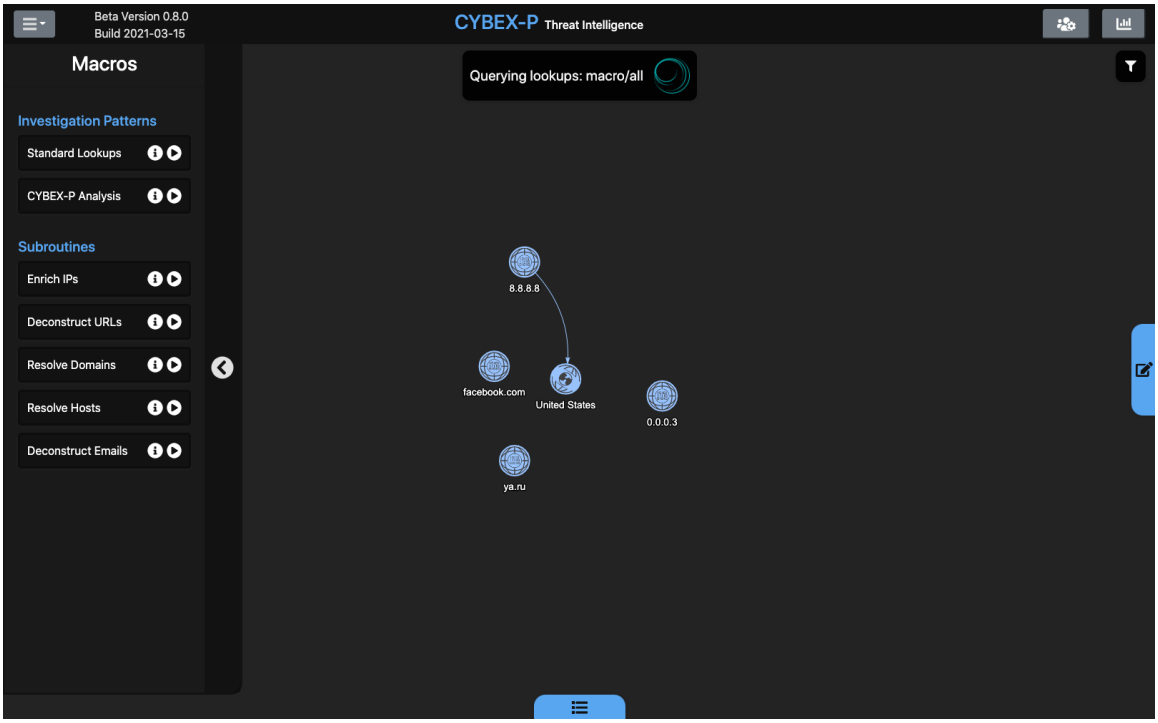


Figure 6.20: The 'loading' state for the 'standard lookups' macro. Users can continue using the graph while the process completes.

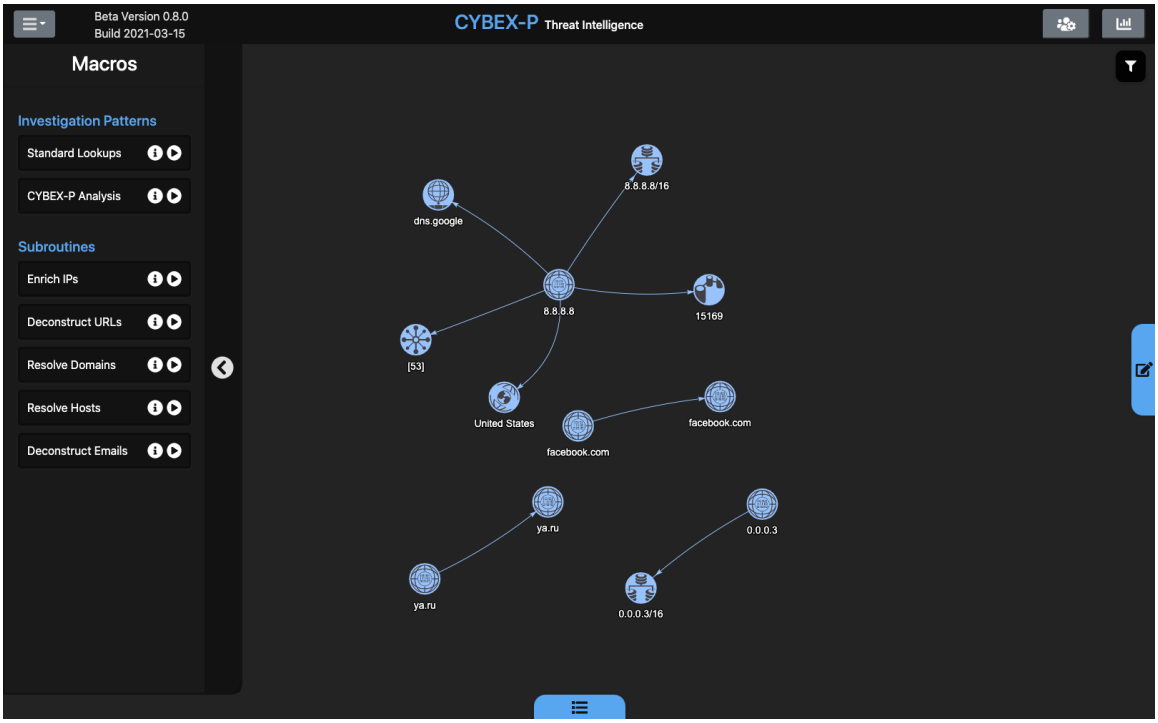


Figure 6.21: The result of a single run of the 'standard lookups' macro.

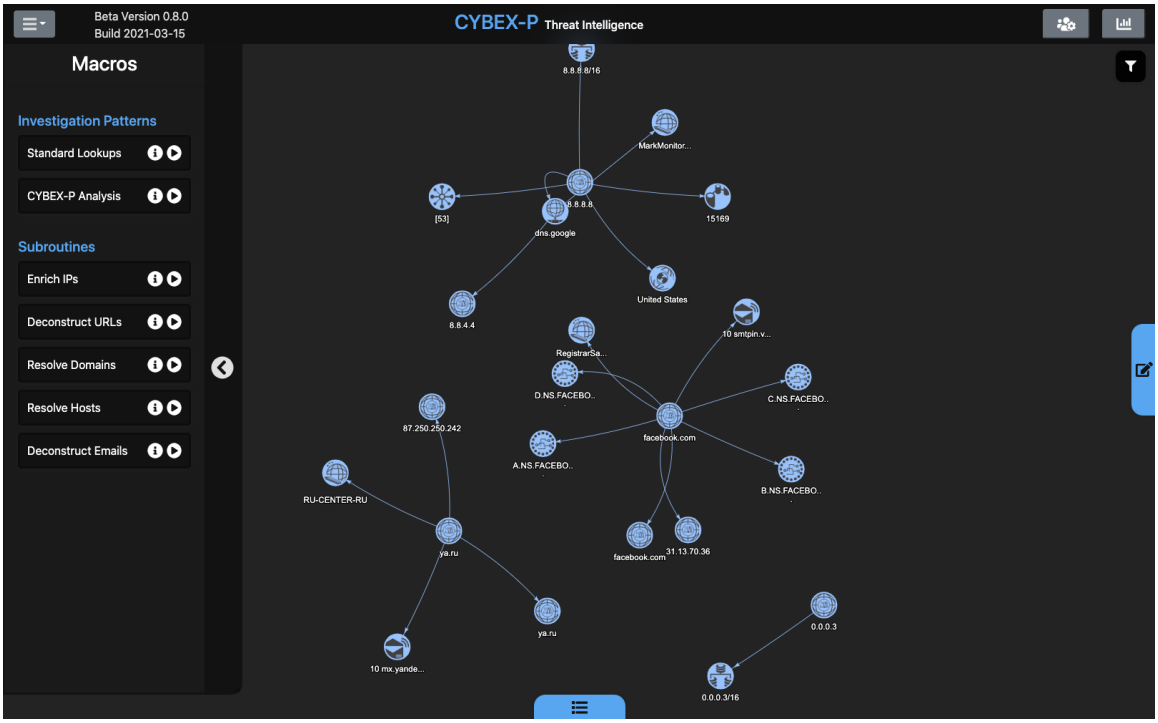


Figure 6.22: The result of two runs of the 'standard lookups' macro.

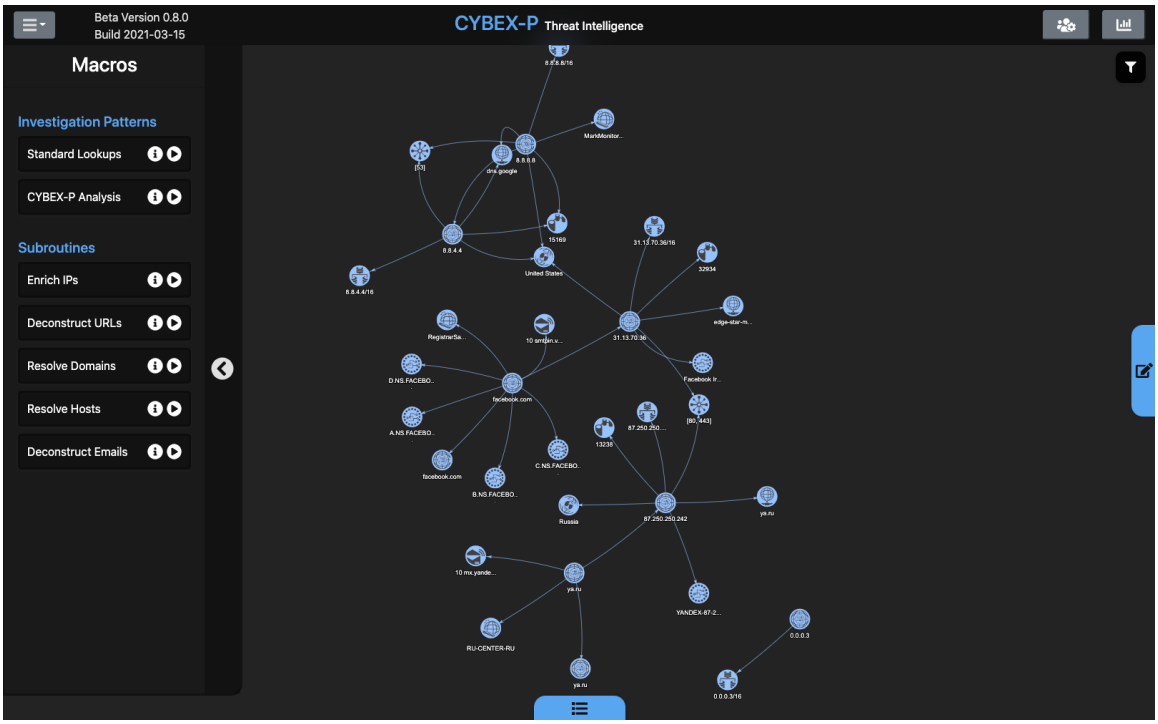


Figure 6.23: The result of three runs of the 'standard lookups' macro.

At this point, the investigator wants to zoom in to part of the graph. They also want to manipulate the graph so that the nodes can still largely be viewed concurrently in this zoomed state. Figure 6.24 shows the graph after the user clicked and dragged one of the bottom node clusters to the upper-right of the canvas. The graph physics also pulls the connected nodes in the same direction as the one the user dragged. When the user lets go, the graph stabilizes and the new positions of all nodes are stored. Figure 6.25 displays a zoomed-in view of this repositioned graph. The investigator is curious about the dual-cluster shape formed in the bottom-left portion of the graph. Zooming in and hovering over the common edge gives a better view of the relationship between facebook.com and the IP address 31.13.70.36. The anlayst wants to eventually figure out if any other IPs are associated with the same host as 31.13.70.36. However, they will need to gather some more data to add to the graph before returning to this question another time. To make it easy to locate later, the node in question is highlighted using the IOC menu controls. Node highlighting of this host name is shown in Figure 6.26.

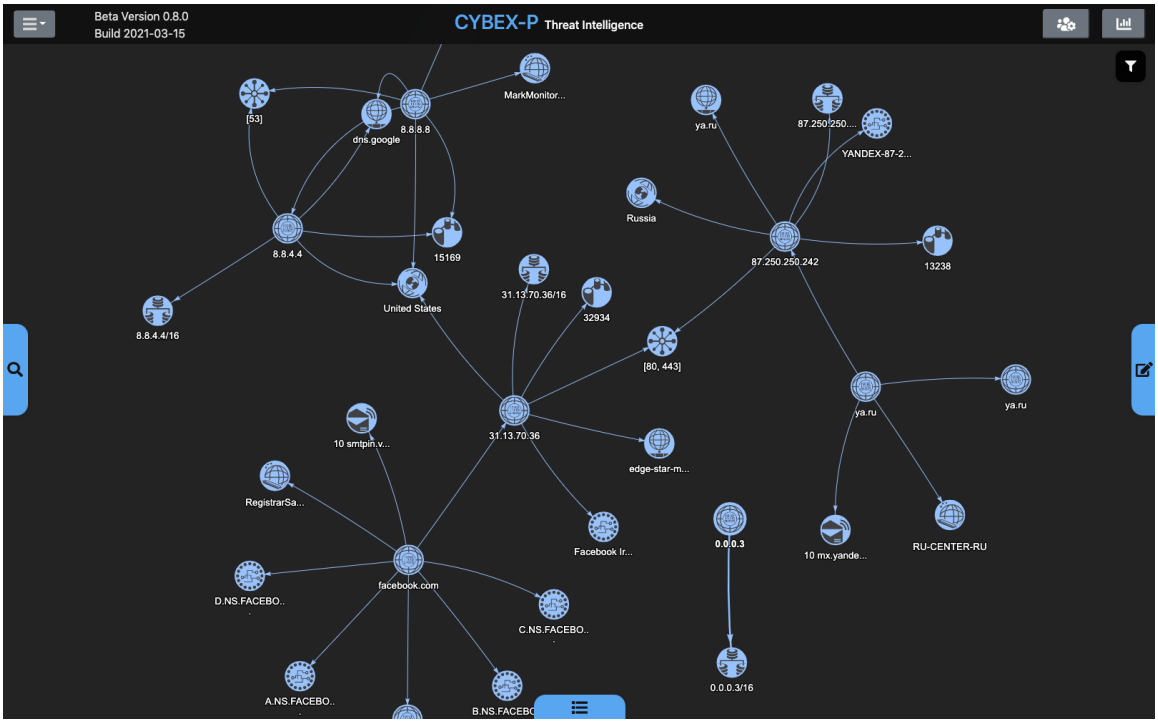


Figure 6.24: Graph nodes can be repositioned however users like, and users can pan/zoom the canvas.

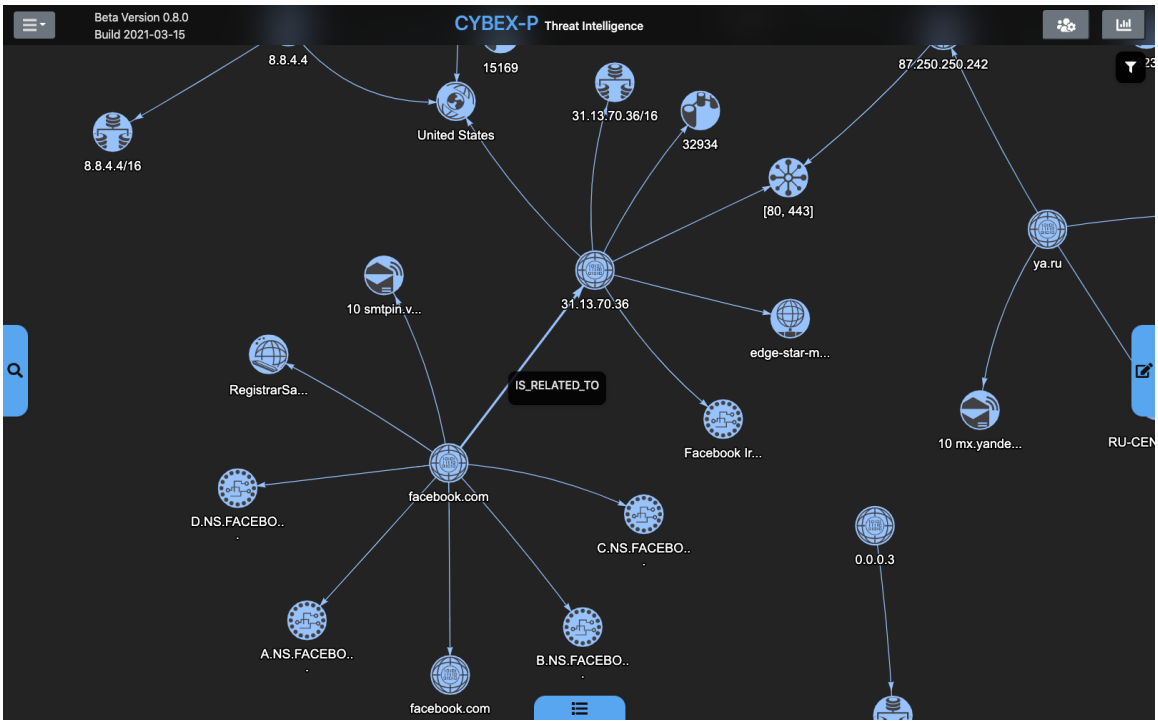


Figure 6.25: Users can pan/zoom the canvas. Hovering over edges reveal information about the relationship between nodes.

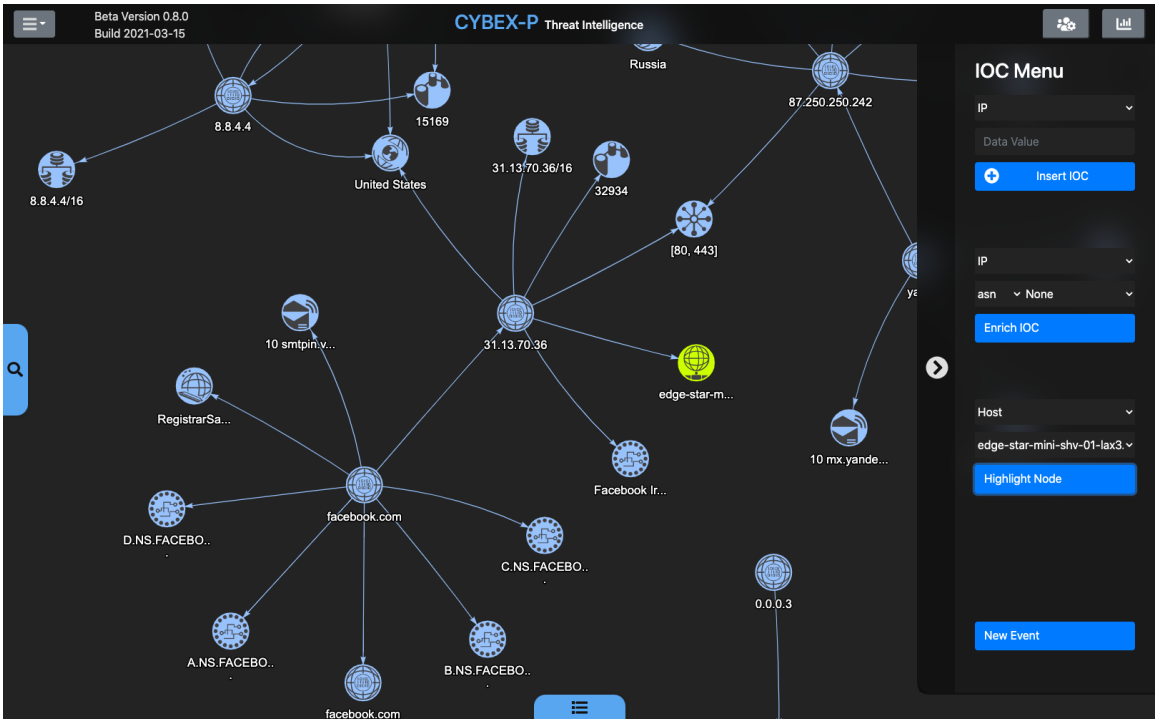


Figure 6.26: Users can mark a certain node to remain highlighted and easily traceable during an investigation.

Next, the analyst wishes to add even more data to the graph. They have already run the standard lookups a few times, but they haven't yet tapped into the CYBEX-P database. They decide to run the cybexRelated enrichment on a few key IOCs. As pictured in Figure 6.27, additional context has now been provided for facebook.com. The previously related data is connected with solid lines, and the new data related by CYBEX is connected with dashed lines. This clearly shows the level of extra context CYBEX-P can provide, relating additional items that were found in common event data. To understand exactly how facebook.com was related to one of these nodes, the user hovers over the edge.

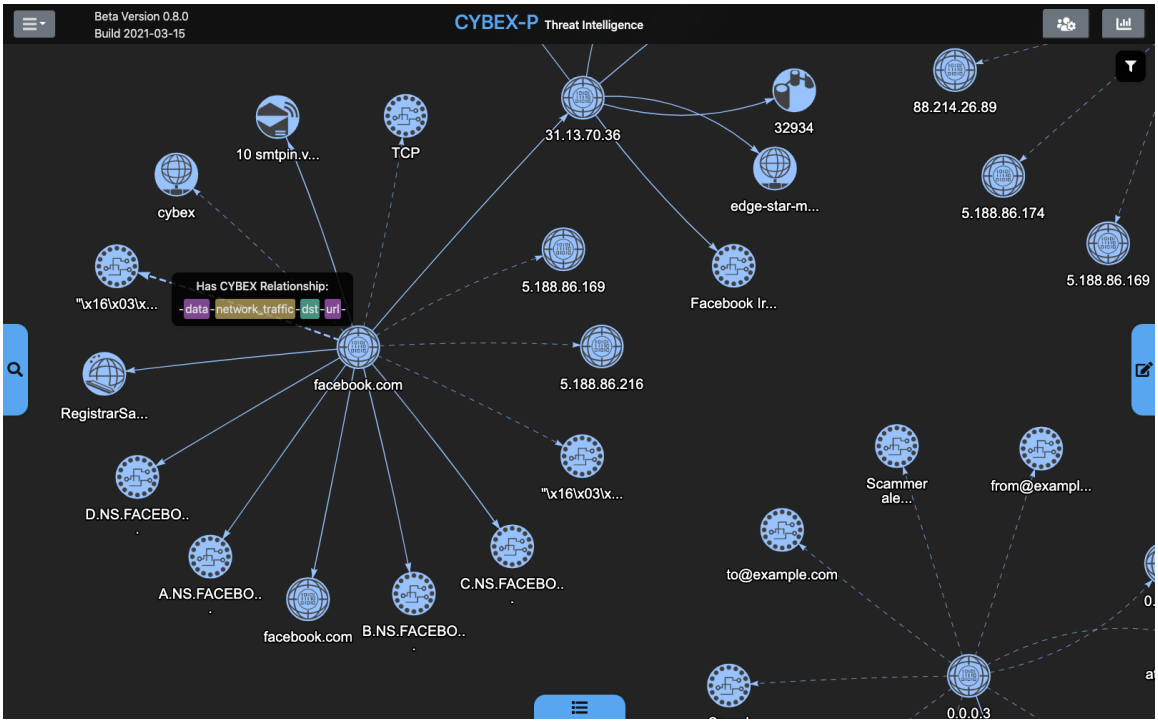


Figure 6.27: The graph is further expanding by using cybexRelated enrichments. The results of these are connected by dashed edges. Context from event data is provided in edge tool-tips to communicate how two objects were connected.

The analyst decides that the graph is now properly visualizing the desired scope of their network. However, there are a few things they would like to trim before sharing this graph with others. For example, the analyst notices that some IP addresses in the graph are not helpful, and instead are a distraction. This could be for many reasons, such as an IP having no meaningful significance to their organization's assets. Perhaps they would like to simplify the graph with only the most pertinent connections so that an executive can review it easily. One IP the user would like to remove is 0.0.0.2, and Figure 6.28 shows them selecting it. They use the delete button in the bottom-left portion of the screen to remove the node and all its connected edges. Figure 6.29 displays the result of this operation.

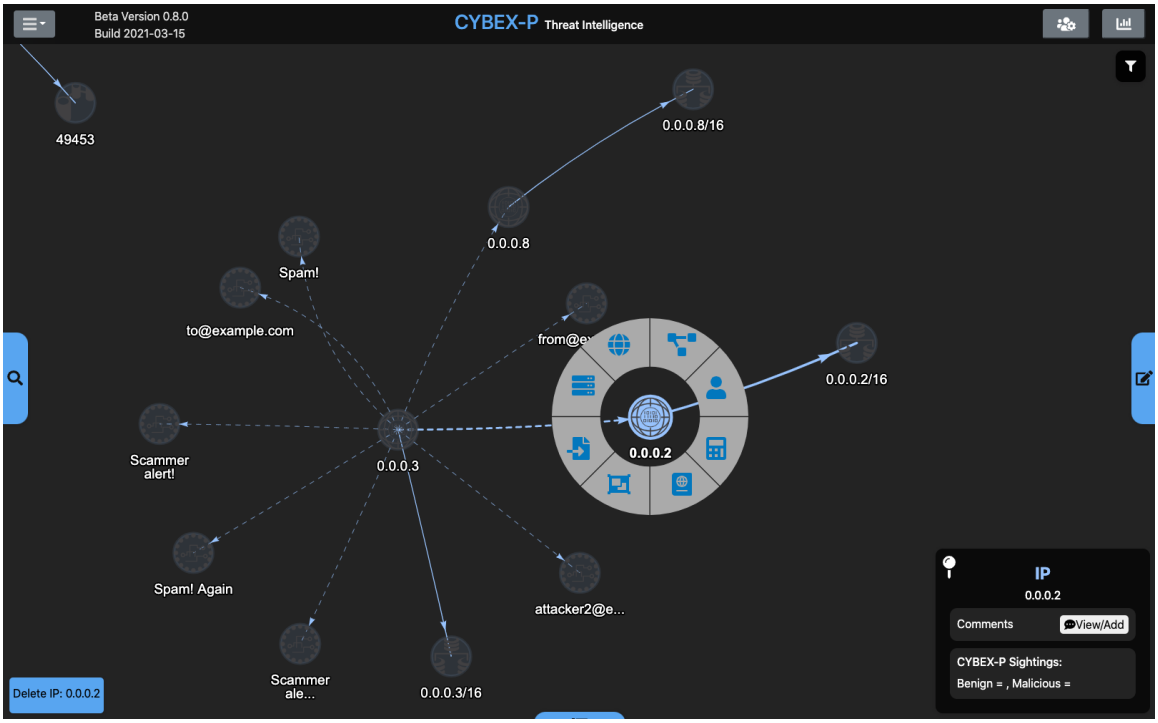


Figure 6.28: To delete a node, the user can select it and then click the bottom-left 'delete' button.

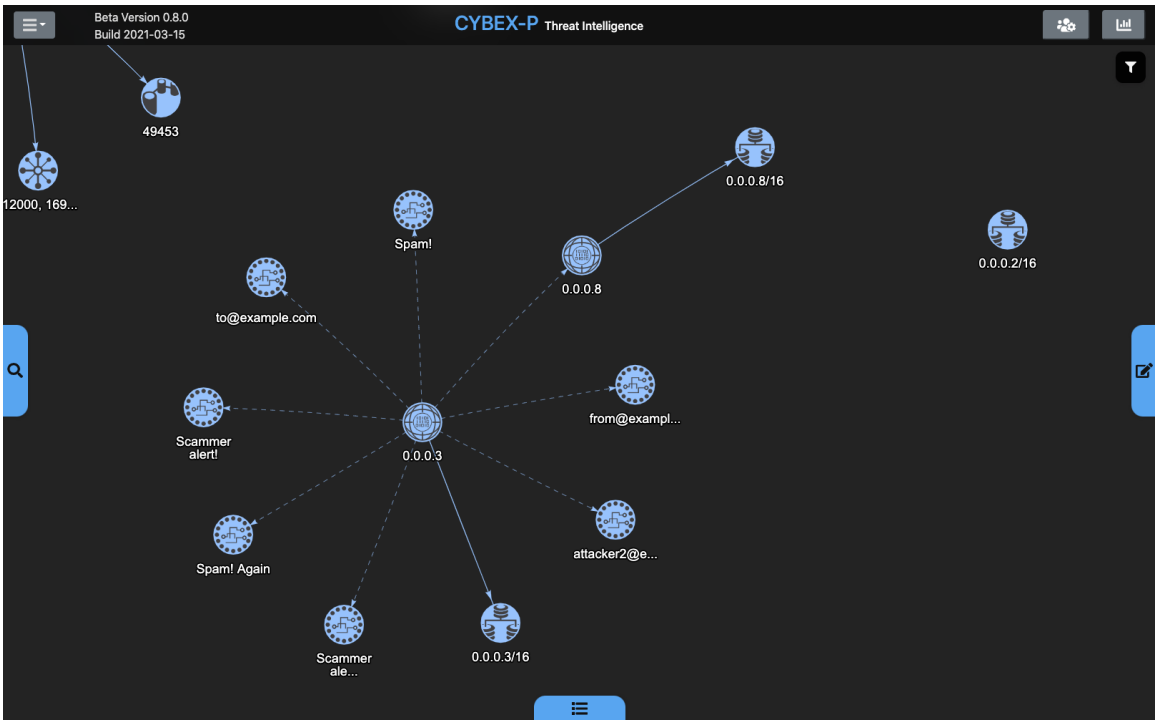


Figure 6.29: The results of deleting a node. Note that all connected edges are removed and related nodes are re-rendered appropriately.

This scenario was entirely exploratory, and focused on building a big-picture visual representation of a particular network. This graph can be referenced when explaining the network's topology, or can even serve as a starting point for threat analysis. Every participating organization of CYBEX-P may want to construct neutral graph templates representing their network assets. Then, when needed, investigators can load these graphs and quickly start running threat analysis. The scenario in the next section focuses specifically on a threat analysis use-case.

6.3 Scenario B: Focused Threat Analysis

For the second scenario, the investigator is specifically concerned about just two IOCs. They have already received information that these IOCs are possibly suspicious, so they want to start from these two nodes rather than some larger network topology. The main goal in this case is to simply display as much threat context about these items as possible. Figure 6.30 shows the two suspicious nodes that the investigator starts with. They immediately run the CYBEX-P Analysis macro, and the result of this operation is shown in Figure 6.31. It is already apparent that the two original IOCs are part of data clusters with very different threat signatures. The IP address 0.0.0.3 itself is colored yellow and classified as moderately malicious. Several other items are found to be related to this IP through CYBEX event data. Almost all of them are colored red, indicating they pose very strong threats (having greater than 50% malicious scores). Meanwhile, the other suspect IOC, a URL called ya.ru, seems to be benign. Most of its related IOCs are classified and colored green.

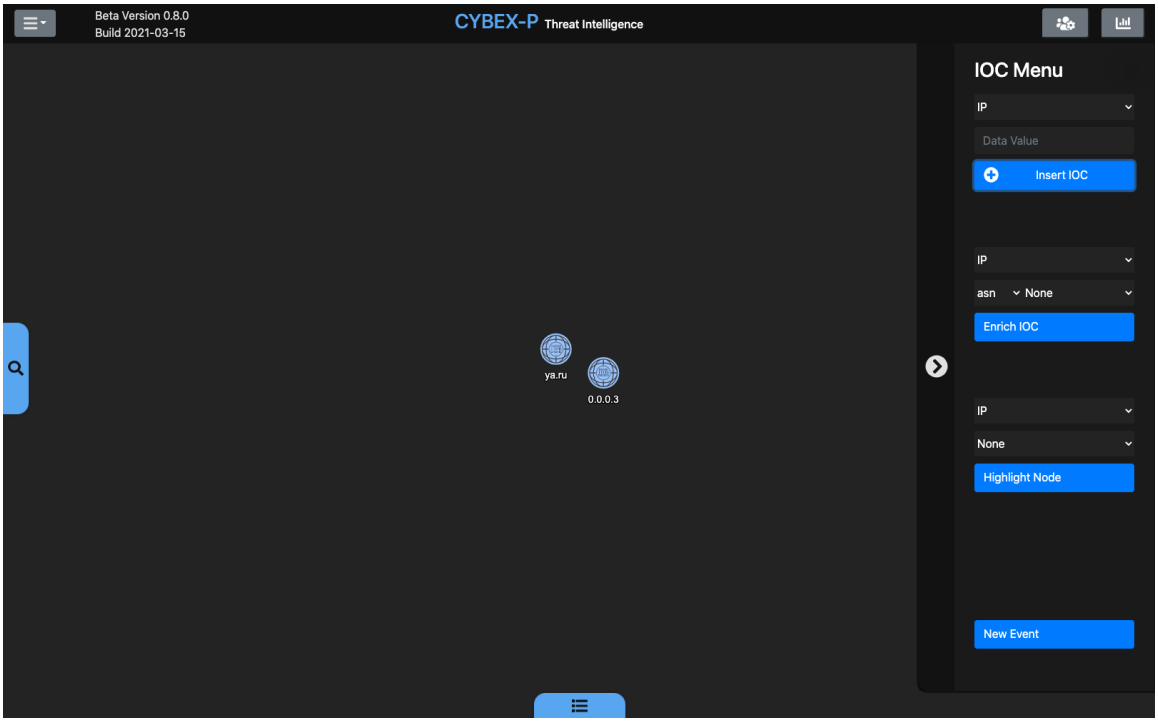


Figure 6.30: The user adds two items of interest to to the graph. The intention is to perform focused threat analysis on these IOCs.

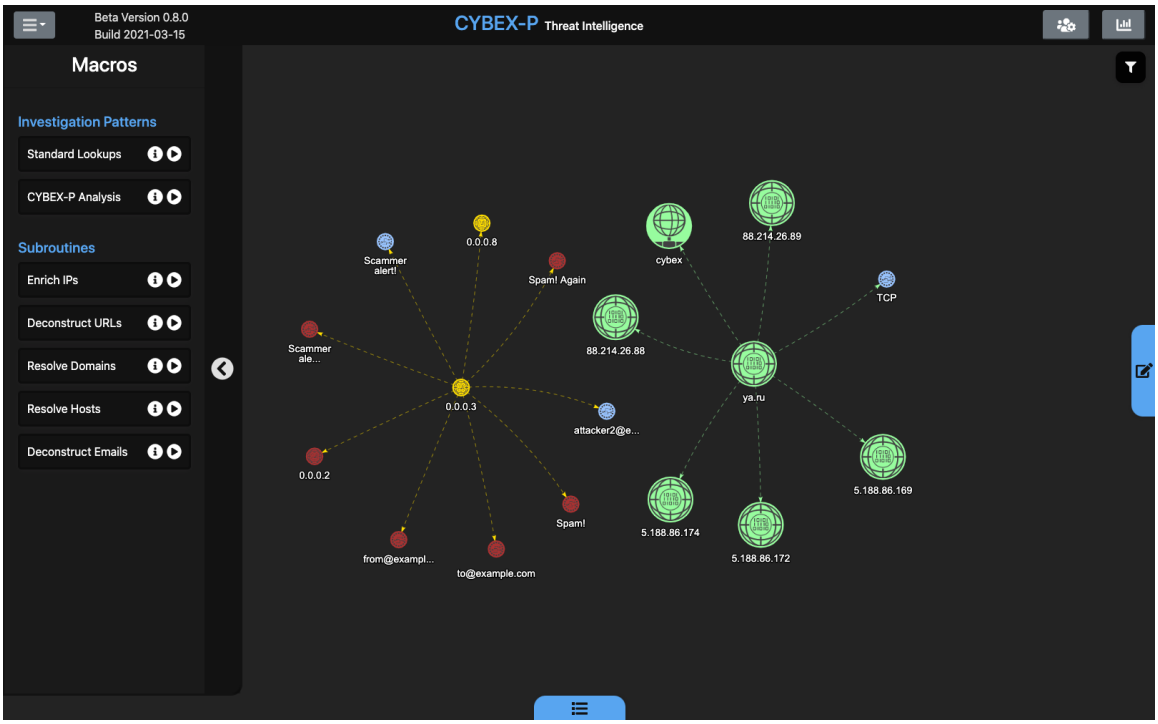


Figure 6.31: The user runs the 'CYBEX Analysis' macro. Related IOCs and attributes are added, alongside automatic threat classification.

Before jumping to conclusions, the analyst wants to perform a few subsequent operations to rule out a threat near ya.ru. They decide to expand the graph scope by running one iteration of the Standard Lookups macro. The result of this is shown in Figure 6.32. This does two things. First, it helps determine if ya.ru has any basic network relationships with the malicious cluster. In this case, no such connection is drawn. Second, there is a new layer of peripheral data from which to run further threat analysis. The investigator decides to run the CYBEX Analysis macro again on this newly expanded graph. Figure 6.33 shows this final graph stage. Some additional data is added via CYBEX event relationships, but no significant new threat classifications are made. The investigator has not been able to show any connection between the benign cluster and the malicious one.

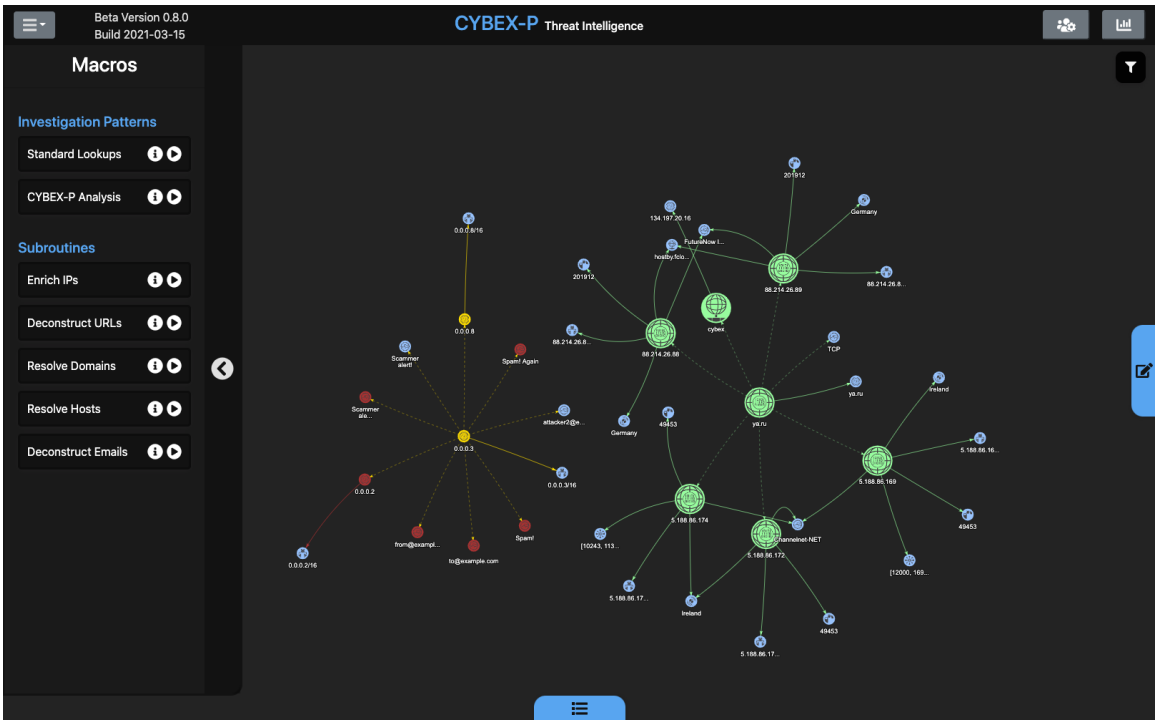


Figure 6.32: The user expands the graph scope by next using the 'Standard Lookups' macro. New data is added that can also be subjected to threat analysis.

The results of this analysis could, at the investigator's discretion, be sufficient to conclude that ya.ru is not a threat. Meanwhile, they can confirm that 0.0.0.3 a threat that should be addressed. The investigator may save this graph and share it

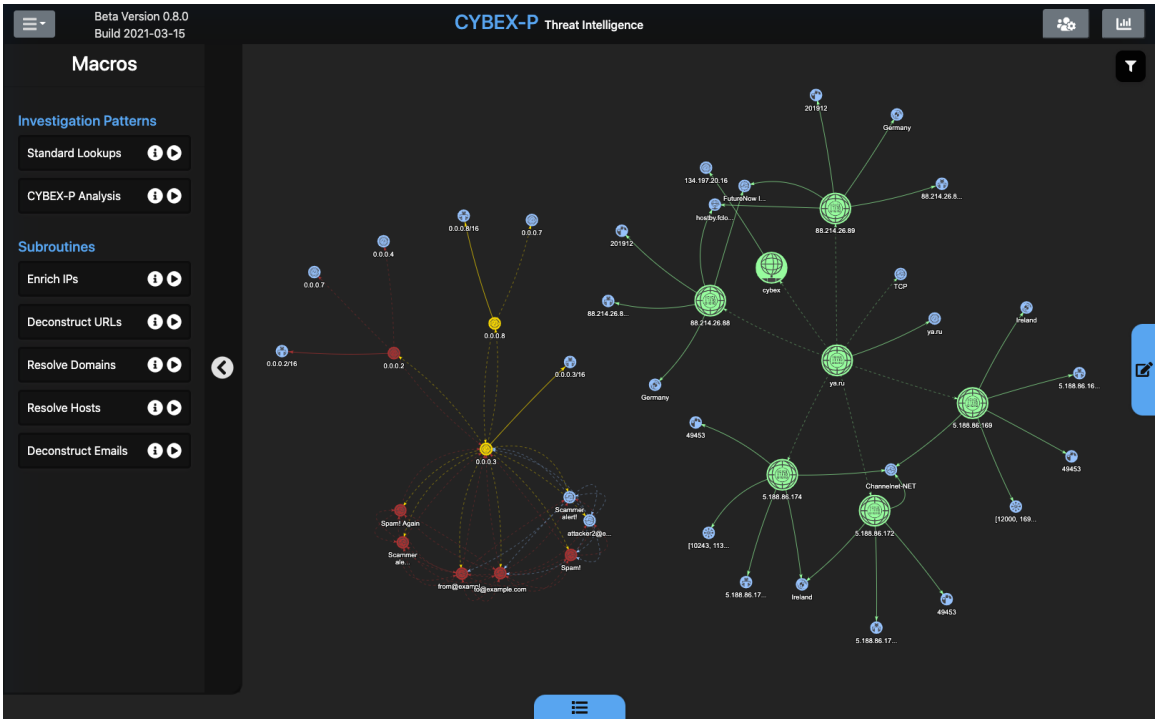


Figure 6.33: The user runs the 'CYBEX Analysis' macro once more. The user can now see the full scope of available threat knowledge and a comprehensive number of important relationships. Each relationship contains on-demand event context that the investigator can selectively inspect.

with their colleagues or superiors so that a threat mitigation plan can be developed. Note that there are two key elements of subjectivity that users of this system must always be aware of. First, the thoroughness of each investigation may vary. In the example in this section, the user ran three total macros (CYBEX-P Analysis twice and Standard Lookups once). In high-stakes situations, these macros may need to be run several more times, in case some malicious relationship exists far at the edges of the initial graph scope. The second consideration is that not all IOCs will be known to the CYBEX-P database. In these cases, neutral classifications are given and the default blue color is retained. Investigators can never assume that neutral items are benign. Instead, they must simply conclude that there is "no evidence of maliciousness". These neutral items should be monitored over time, in case some relevant event data eventually gets submitted to the system.

Chapter 7

User Study

7.1 Objectives

CYBEX-P's threat-intelligence graph requires a visualization method that is tailored to its unique approach of deriving threat metrics. To this end, this chapter seeks the best way to visualize CYBEX-P's IOC threat scores. These are calculated based on object relationships found within both benign and malicious event contexts. The backend methodologies are worth very little to analysts if their results cannot be easily understood and trusted. Thoughtful consideration is required for a frontend system that enables accurate and quick threat detection, especially among interrelated entities of varying threat levels.

This chapter details a user study with participants from both academic and industry organizations who are involved in the cybersecurity field. Various graph visualization techniques are tested, seeking a design that is intuitive to understand while prioritizing effective threat identification. The threat-intelligence graph was altered in design to determine the optimal configuration that achieved the above goals. The study consisted of user trials to test out three visualization instances. Fundamental visual properties such as size and color were employed to communicate object threat. One configuration also portrayed additional metrics to evaluate how greater information density influences task accuracy. User actions were tracked and timed to see which technique can facilitate the given task most efficiently. Entry and exit questionnaires were also used to further inform the experiment's results.

The study's structure is described in the next section, followed by results analysis, discussion, and study conclusions. The research in this chapter is contributed in collaboration with other investigators[1].

7.2 Experimental Setup

7.2.1 Participants

To ensure a suitably representative participant demographic, we extended invitations for this user study to individuals at our university and within industry who have cybersecurity backgrounds or interests. Examples of these participants were systems analysts, administrators, security engineers, information officers, faculty, and graduate students who are involved in cybersecurity research. In particular, we recruited participants from the university's Information Technology Office. In total, the study was performed with 13 participants (12 men and one woman) of the above backgrounds. Age ranges were from 20s to 60s.

7.2.2 Apparatus

Hardware and software components were required to perform the user study. The virtual meeting was conducted through Zoom, which allowed participants to remotely control the study computer. All tasks were hosted on the study coordinator's machine. The experiment was conducted online through Google Chrome. Usage monitoring and timers were built directly into the application. The hardware used by the coordinator was a MacBook Pro running macOS 10.15. On the user side, the hardware used was the personal computers of the users, with minimal requirements. Participants simply needed the ability to run the Zoom client, and use a mouse for input. Lastly, digital questionnaire documents were handed out to each participant before and after they took part in the study. Fig. 7.1 illustrates the Zoom interface for conducting the user study.

[1]Tapadhir Das and Zachary Black were co-researchers for this study.

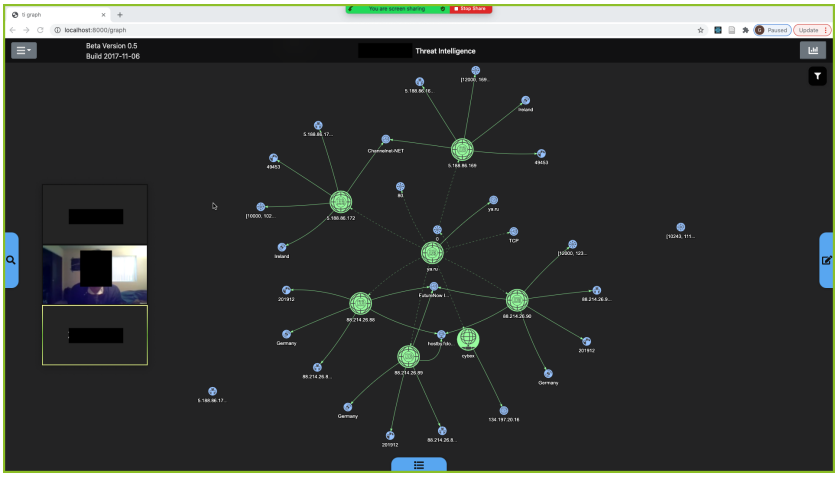


Figure 7.1: Zoom interface for the user study.

7.2.3 Procedure

Due to current COVID-19 restrictions, this user study was conducted fully remotely. When participants joined the virtual meeting, the study coordinator provided the *entry questionnaire*, which consisted of 10 questions. Besides gathering demographics data, one of the primary goals of this questionnaire was to measure initial assumptions about which visualization configurations would be most optimal. Participants were given drawings representing the three conditions to be tested, and were asked to rank their expected effectiveness on a scale of 1 to 5 (1 being highly ineffective, 5 being highly effective). More details on entry survey questions are provided in Subsection 7.3.2.

Once the *entry questionnaire* was filled and submitted, the study coordinator went over the instructions once more, gave a brief overview, and showed a pilot demo of what the user was expected to perform. Once the introduction was concluded, the study coordinator handed over the remote control of their station, so that the participant could commence the user study. The user study was conducted using a modified build of the CYBEX-P web application.

After completing all tasks, the participant was asked to answer an *exit questionnaire*, which consisted of 12 questions. For the most part, the exit questionnaire

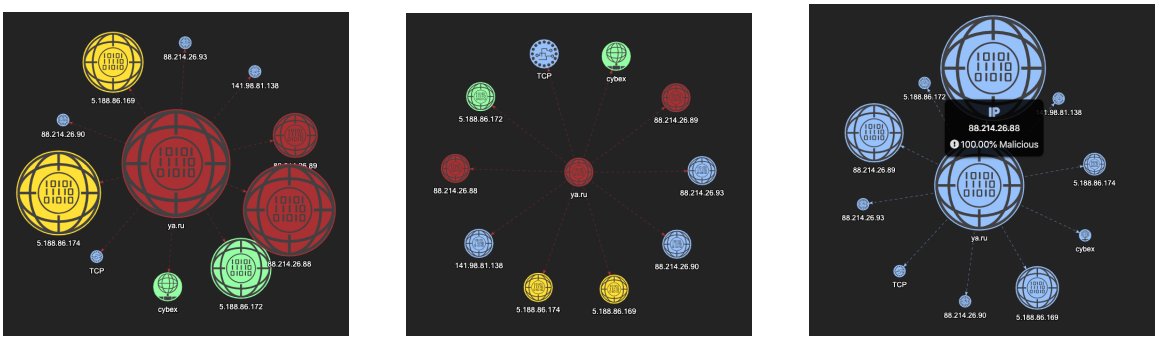
closely followed the structure of the entry questionnaire, however it also included several specific questions. Participants were asked to rank the effectiveness of each configuration again after directly experiencing them. Participants were also given opportunities to leave open-ended feedback regarding their interaction with the visualizations and the tool. More details on the exit survey questions are provided in Subsection 7.3.2.

7.2.4 Design

The user study was designed around assessing the influence of visualization-focused independent variables on measures of threat-detection performance. This research focused on network-graph representations, where nodes are IOCs and edges are the relationships that bind them. Relationships could exist due to standard lookups (such as querying the ASN group of an IP address), or may have been the result of CYBEX-P finding entities within similar event contexts. Threat analysis could be performed on any IOC, no matter which way it was added to the graph.

The study used a within-subjects structure regarding the participants, where users tested each condition. The **independent variable** was *the method of visualization* for communicating an object's threat level. Specifically, the **test conditions** for this independent variable were the following: **Condition A ("color + size")**, which used node color for threat level and node size for 'sightings'. Sightings were defined as the number of times the object was collectively seen within the database (i.e. prominence). The colors in this case were red for high threat, yellow for moderate threat, and green for no threat. **Condition B ("color")** implemented node color only, without the 'sightings' size visualization. **Condition C ("size")** exclusively used node size, but in this case corresponding to threat level rather than sightings. Examples of each of the three configurations are shown in Fig. 7.2.

The three conditions helped assess the role that node color and node size played. The following **dependent variables** were measured: *average task completion time* (the time it took users to complete the threat detection task) and *task accuracy* (how



(a) **Condition A:** *color = threat level, size = # of sightings*

(b) **Condition B:** *color = threat level, # of sightings not displayed*

(c) **Condition C:** *size = threat level, # of sightings not displayed*

Figure 7.2: The three visualization configurations examined in the study.

many node selections a user made before correctly selecting the most malicious node). These metrics could reveal whether additional manipulation was required by the user to succeed in one condition versus another.

Finally, counterbalancing was implemented to account for users being able to 'learn' general dashboard navigation as they moved through the tests. This was done using a Latin square [27], dividing participants into three groups corresponding to each of the test conditions. In conjunction with the aforementioned parametric testing, entry and exit questionnaires provided data for additional non-parametric tests.

7.2.5 Tasks

The participants were assigned the same **two tasks** to complete for each of the three conditions. The first task was to identify the most malicious object in a *small graph* (Fig. 7.3). This tested the visualization's effectiveness in a drilled-down setting in which only a few nodes were examined. The second task was to identify the most malicious object in a *large graph* (Fig. 7.4). This assessed the visualization's effectiveness at an overview level. In order to fit more nodes on-screen, graph elements were scaled-down to create a zoomed-out initial view. Both scenarios were representative of working dashboards used by cybersecurity analysts. Graph size may vary

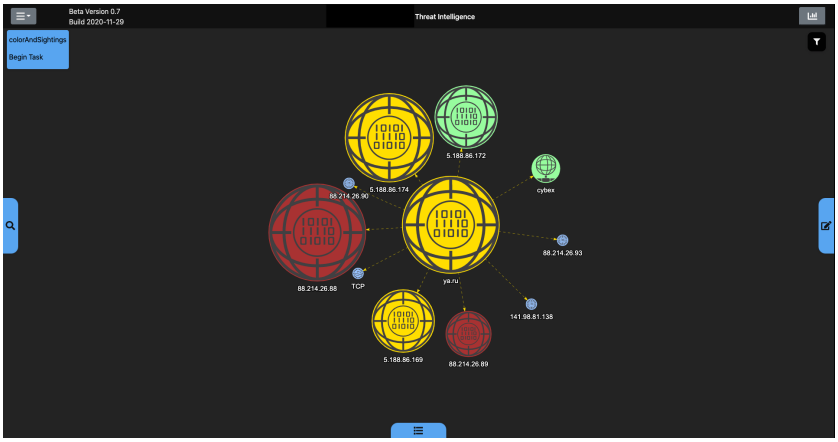


Figure 7.3: CYBEX-P Interface - Example of small graph, with few nodes.

according to investigation stage and type, so this task design helped generalize our results. Note that the order of *small graph* first, *large graph* second was a control.

A built-in timer was used to count the seconds it took each user to find the most malicious node. This was measured as the time differential between the start of the task and the moment the user clicked the node with the highest threat score. Each node had varying levels of threat, and threat scores were represented by a percentage. In all cases, there was only one node with a threat score of 100%, and therefore that was the node that users needed to find. Throughout all components, the software tracked the number of times that the user interacted with the graph (select/hover/zoom/pan). The overall structure (e.g. node/edge count, cluster shapes, and symmetry) of the graphs were the same for all users to provide consistency. However, the malicious nodes' locations were selected as a random variable for every user so that the study was generalized.

7.3 Results

The results from the conducted user study are divided into parametric testing and non-parametric testing.

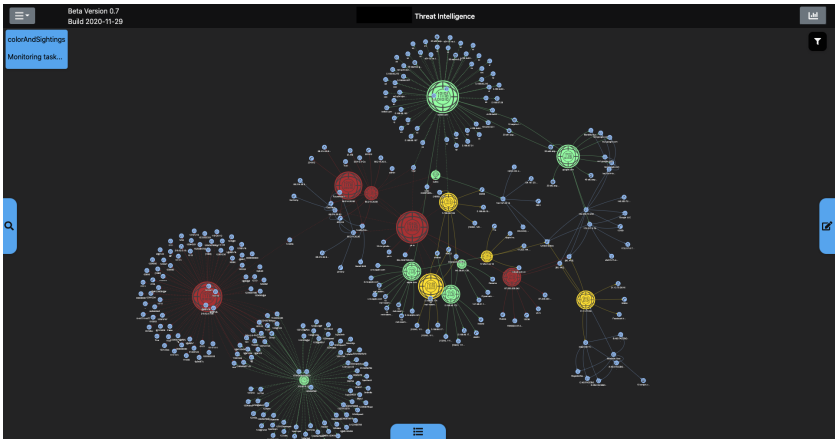


Figure 7.4: CYBEX-P Interface - Example of large graph, with many nodes.

7.3.1 Tests Conducted

Parametric Test

Four Analysis of Variance (ANOVA) tests were conducted using parametric interval data collected regarding each participant's task performance [27]. In this experimentation, we performed the one-way repeated measures ANOVA as our user study was within-subjects [11]. Two dependent variables, *task completion time* and *task accuracy* (number of node selections), were the focus of these tests. Each of these two variables were studied for the separate *small graph* and *large graph* tasks within each test condition (four ANOVA tests in total). All ANOVA tests were run with $\alpha = 0.05$ to enforce a strong threshold of statistical significance. For all ANOVA tests that yielded statistically significant results, further post-hoc comparisons were performed to determine which conditions differ from which other test conditions. For these comparisons, we opted to use pairwise t-tests with $\alpha = 0.05$. The results of these comparisons form the basis of the objective conclusions described later in our discussion (Section 5). The parametric testing consisted of 13 participants who provided task performance on 3 different data visualization methods.

Non-Parametric Test

For our non-parametric testing we chose to perform three Friedman Tests divided between our entry questionnaire and exit questionnaire data [27]. This type of test was chosen because the data gathered from our surveys was ordinal and the groups involved were independent of each other. The χ^2 statistic obtained from each Friedman test was compared to a value of 5.991, according to Friedman's ANOVA by Ranks Critical Value table [27]. Whether χ^2 was greater or smaller than the stated constant determined if there was a significant difference detected in our survey results. Similar to the above, our non-parametric tests were conducted with 13 participants who provided survey information on three different data visualization techniques.

7.3.2 Findings

ANOVA

Of the four ANOVA tests conducted, one was statistically significant, and the other three failed to meet the 5% threshold for significance. First, the test on number of selections (accuracy) is presented for the *small graph* task. This is followed by results for the same dependent variable pertaining to the *large graph* task. Next, results of task completion time on the *small graph* are described. The final test analyzes task completion time on the *large graph*. Task completion time is defined as the number of seconds it took for the participant to select the most malicious node.

The mean number of node selections (our measure of accuracy) for Condition C on the *small graph* was 1. This was 0.15 selections (15%) less than the mean of 1.15 observed for Condition B, and 0.54 selections (54%) less than the mean of 1.54 observed for Condition A. Table 7.1 displays the collected data with means and standard deviations. The means across the three conditions for the *small graph* task are visualized in Fig. 7.5. The difference was statistically significant ($F_{2,24} = 9$, $p<0.05$). ANOVA test results are listed in Table 7.2. Of the three conditions, the difference between Condition A and Condition B was significant ($p<0.05$). The difference between Condition A and Condition C was also significant ($p<0.05$). The other comparison,

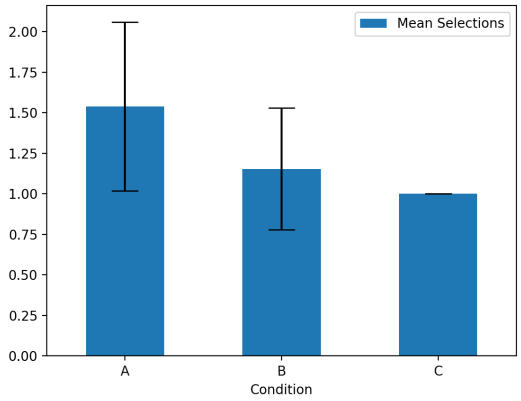


Figure 7.5: Mean number of node selections during each of the three test conditions for small graphs. Note that the standard deviation for Condition C is zero.

between Condition B and Condition C, did not meet the 5% threshold for significance. The three comparisons are displayed in Table 7.3.

Participant	Small - A	Small - B	Small - C	Large - A	Large - B	Large - C
1	1	1	1	1	1	2
2	1	1	1	1	1	1
3	1	1	1	3	1	1
4	2	1	1	1	1	1
5	1	1	1	1	1	1
6	2	1	1	1	5	1
7	1	1	1	1	1	1
8	2	2	1	1	4	1
9	2	1	1	1	1	1
10	2	1	1	2	1	1
11	2	1	1	1	1	1
12	2	2	1	3	2	2
13	1	1	1	1	1	1
Mean	1.54	1.15	1	1.38	1.62	1.15
STD	0.52	0.38	0	0.77	1.33	0.38

Table 7.1: Recorded node selection events with means and standard deviations

The mean number of node selections on the *large graph* was 1.38 for Condition A, 1.62 for Condition B, and 1.15 for Condition C. As there was substantial variation in the observations across participants, the difference was not statistically significant as revealed in an analysis of variances ($F_{2,24} = 0.833$, ns).

Table 7.4 displays the collected data pertaining to task completion times with

Source	Sum of Squares	DF	Mean Square	F-Value	P-Value
Within-Subjects	2	2	1	9	0.001
Error	2.667	24	0.111		

Table 7.2: Analysis of variance table for select events across conditions during the small graph task

Contrast	I	II	T-Statistic	DF	P-Value
conditions	Condition A	Condition B	-2.739	12	0.018
conditions	Condition A	Condition C	-3.742	12	0.003
conditions	Condition B	Condition C	-1.477	12	0.165

Table 7.3: Post-Hoc comparisons of test conditions using Pairwise T-tests

means and standard deviations. The mean task completion time on the *small graph* was 10.08 seconds for Condition A, 16.31 seconds for Condition B, and 6.15 seconds for Condition C. The means across the three conditions for the *small graph* task are visualized in Fig. 7.6. As there was substantial variation in the observations across participants, the difference does not meet the acceptable threshold for significance in an analysis of variances ($F_{2,24} = 2.665$, $p>0.05$). The results of the ANOVA test are shown in Table 7.5.

The mean task completion time on the *large graph* was 17.08 seconds for Condition A, 17.85 seconds for Condition B, and 10.31 seconds for Condition C. As there was substantial variation in the observations across participants, the difference does not meet the acceptable threshold for significance in an analysis of variances ($F_{2,24} = 1.179$, $p>0.05$).

Friedman Tests

Entry Questionnaire Test This test was based on a set of three questions in our entry questionnaire, which gave the participants a brief graphical depiction of the type of dashboard display they would encounter during the study, such as the one shown in Fig. 7.7. The three questions, mapped to the three test conditions (Conditions A, B, and C), were: "Consider an implementation that uses color to represent the threat level of an object. In addition, assume node size is used to reflect prominence of each object (i.e. how often it is seen in event reports). How effective would you

Participant	Small - A	Small - B	Small - C	Large - A	Large - B	Large - C
1	4	6	6	10	11	19
2	4	7	2	3	5	12
3	9	4	3	60	9	10
4	23	15	6	11	11	20
5	2	6	4	9	28	6
6	23	6	6	10	50	9
7	5	71	13	10	5	4
8	5	34	7	11	44	6
9	7	6	5	6	19	7
10	9	23	8	36	22	10
11	13	9	6	12	10	13
12	17	17	6	33	5	11
13	10	8	8	11	13	7
Mean	10.08	16.31	6.15	17.08	17.85	10.31
STD	7.03	18.57	2.70	16.14	14.70	4.84

Table 7.4: Recorded task completion times (in seconds) with means and standard deviations

Source	Sum of Squares	DF	Mean Square	F-Value	P-Value
Within-Subjects	681.692	2	340.846	2.665	0.09
Error	3069.641	24	127.902		

Table 7.5: Analysis of variance for task completion times across conditions during the small graph task

expect this more complex implementation to be for identifying malicious nodes?"; "To detect malicious nodes on a network graph, how effective would you expect color to be as a visualization tool (red = malicious, yellow = moderately malicious, green = safe)?"; and "To detect malicious nodes on a network graph, how effective would you expect dynamic node sizing to be as a visualization tool (where larger node size corresponds to a higher threat level)?".

These questions asked the participants to rank the method of visualization from 1 to 5, with 1 denoting a very ineffective display method and 5 denoting a very effective display method. By analyzing the data collected using the Friedman test, the test statistic was 20.217 with a p-value of statistical significance ($p<0.05$). Table 7.6 displays statistics such as the median and lower/upper quartile ranges of all responses.

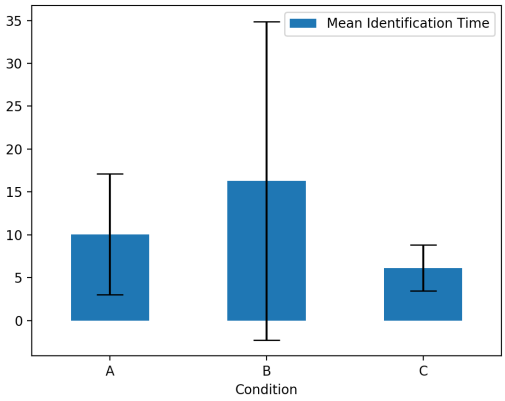


Figure 7.6: Mean number of seconds to complete the small graph task across the three test conditions.

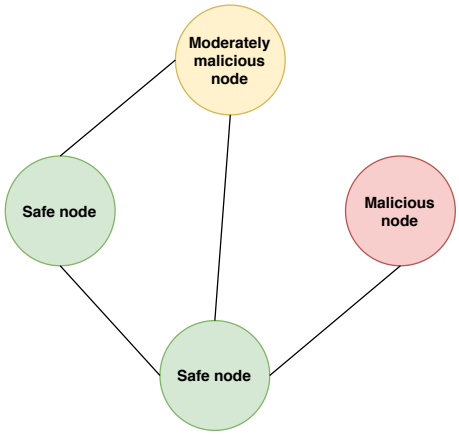


Figure 7.7: Example of one of the graphics shown to participants in the entry questionnaire.

Exit Questionnaire Test One This test was based on a set of three questions in our exit questionnaire, which asked the participants how effective each graph-based visualization technique (Conditions A, B, and C) was for the *smaller network graphs*. The three questions, mapped to the three test conditions were: "To detect malicious nodes on the smaller network graph, how effective was the color AND size configuration (color = degree of maliciousness, size = prominence of node)?"; "To detect malicious nodes on the smaller network graph, how effective was the visualization that only used color to reflect threat level (red = malicious, yellow = kind of malicious, green = safe)?"; and "To detect malicious nodes on the smaller

	Condition A	Condition B	Condition C
Median	4.5	5	4
Lower Quartile	4	4.75	3.75
Upper Quartile	5	5	5

Table 7.6: Median, lower, and upper quartile range statistical information for the entry questionnaire Friedman test

	Condition A	Condition B	Condition C
Median	4	4	4
Lower Quartile	3	4	4
Upper Quartile	5	5	4

Table 7.7: Median, lower, and upper quartile range statistical information for the first exit questionnaire Friedman test

network graph, how effective was the visualization that only used node size to reflect threat level (larger size corresponds to more malicious nodes)?".

Similarly to the entry questionnaire test, the questions asked the participants to rank the methods from 1 to 5, with 1 denoting a very ineffective display method and 5 denoting a very effective display method. By analyzing the data collected using the Friedman test, the test statistic was 16.329 with a non-significant p-value of $p>0.05$. Table 7.7 shows statistics such as the median response of participants as well as the lower/upper quartile ranges of all responses.

Exit Questionnaire Test Two Similar to the first exit questionnaire test, this test was based on another set of three questions that asked the participant how effective each graph-based visualization technique (Conditions A, B, and C) was for *larger network graphs*. These questions were identical to those in the *exit questionnaire test one*, but referred to the larger graphs. These questions also asked the participants to rank the methods from 1 to 5. By analyzing the data collected using the Friedman test, the test statistic was 17.831 with a non-significant p-value ($p>0.05$). Table 7.8 displays additional statistics pertaining to the analysis of these questions.

	Condition C	Condition B	Condition A
Median	3	3	4
Lower Quartile	3	4	2
Upper Quartile	5	5	5

Table 7.8: Median, lower, and upper quartile range statistical information for the second exit questionnaire Friedman test

7.3.3 Sample Comments from Participants

Besides questions with answers on a Likert scale of 1 to 5, our exit questionnaire also included the following two open-ended questions: *"What improvements would you recommend to make the graph visualization more effective?"* and *"If applicable, how could this software be best tailored to your work?"*. Some of the most salient comments received are provided here.

"Color is better for categorical variables. Maliciousness is generally not a categorical thing for IP addresses because they are elastic and can be reallocated. Finally, with respect to size, that's best for a continuous variable. In general, the more connections and more nodes, the less readable a visualization is."

"A companion tool that mapped similar data on an actual map that could be viewed side-by-side with the graph would make visualization richer. I am doing comparisons of malicious data based on locality, so a way to visualize geographic data would be helpful to me."

"I would be most interested in finding the top 10 most malicious nodes listed in a dashboard along with a quick reference to the threat indicators which make those nodes so malicious. Then if I could perhaps click on one of those malicious node items in a list and then have that action navigate me to a graph where that particular node is highlighted or put into center focus, this might help with the more immediate triage steps to discover threats which would impact my environment. I would use this tool to help my tier-1 infosec analysts, who generally need an alert or highlighting, to take advantage of the low-hanging fruit to begin an investigation."

"The large network graphs were very busy. It might be useful to have the initial

few show only suspected malicious activity nodes (red or yellow) then other nodes could be added as a drill down, when the user wanted to explore that item in more detail. A lot of nodes representing non-malicious activity or details that aren't needed on the initial assessment look cool, but they can be distracting."

"I noticed that using size to represent prominence can be confusing for trying to determine how large the threat is."

These comments and suggestions, together with feedback received from other participants, helped us formulate our key qualitative findings and determinations, especially in the following aspects: [i] greater effectiveness of size-only versus color-only for threat identification; [ii] careful use of information density (for example adding extra sightings data could actually decrease threat detection accuracy); and [iii] the users' needs for more context on IOCs.

More details pertaining to our most significant findings and determinations are provided in Section 7.4.

7.4 Discussion

7.4.1 Survey Insights

In the scope of the non-parametric tests performed, our data reveals that participants had an innate bias going into the study about the effectiveness of a color-only dashboard. The color-only question had a median response of 5 and a lower quartile of 4.75. These values were significantly higher than the other responses and our distribution for these three different questions were found to be statistically significant ($p < 0.05$, $\chi^2 = 20.217$)). These results suggest that there was a measurable difference in assumed preference between the three visualizations. These data serve as a baseline of participant sentiment before exposure to the visualizations. In the following, we compare these initial opinions with post-study sentiment as well as with separate empirical measurements.

Unlike the entry questionnaire results, the exit questionnaire results had no sta-

tistical significance. These tests asked users to rate each configuration using the same 1 to 5 Likert scale as the entry survey questions. We found that the threshold for significance was not met when observing both the *small graph* tasks ($p > 0.05$, $\chi^2 = 16.329$), and the *large graph* tasks ($p > 0.05$, $\chi^2 = 17.831$). This suggests that there is not provable preference among the visualizations after participants have interacted with them. As further detailed in this section, additional data must be examined to draw significant conclusions.

7.4.2 Task Analysis

Given the variability regarding self-reported preference, we next analyzed our recorded task metrics for additional insights. The *small graph* tasks tested participants' immediate responses to visual stimuli, representing split-second decisions influenced by the visualization. This may explain why more significant results were observed from these tests than from the *large graph* tests. Before examining some key conclusions from the small graphs, it is worth first explaining the variability that was observed for large graphs.

Large Graph Variance

The *large graph* tasks saw very high variability in all recorded metrics. These tasks required participants to consider a larger amount of information and make more decisions. Such scenarios were susceptible to a greater magnitude of outside factors. These included some differences in participants' existing habits regarding process of elimination. Some users would act swiftly and select the most malicious node once it was first discovered. Others, however, would be more methodical in their approach. A few participants would mentally note the correct node, but still feel a need to scan the rest of the graph before returning to the correct choice. When asked about such behavior, these participants cited existing investigative tendencies. Many security professionals are accustomed to acting cautiously and thoroughly when conducting their real-world analyses. Among these participants, we observed a reluctance to

rely on immediate visual representations without first building more context. Understanding the IOC types of certain nodes and the details of their relationships were examples of such context. In the *large graph* scenarios, there were more opportunities for these factors to contribute to variance.

Other factors that may have contributed to interaction metric variability were limitations pertaining to conducting the study remotely due to COVID-19 guidelines. Performance of remote screen-control software was better for some participants than others. If poor network performance was encountered during the study, this could cause some delay between users' actions and visual response. This was more disruptive during the *large graph* tasks, where user actions such as zoom and pan were more often necessary.

Small Graph Observations

Task accuracy was defined as the number of node selections made before completing the task. A value of 1 for this measure was considered a perfect score, as the very first node selection must have been correct. For the *small graph* tests, parametric testing of selection events suggests that using node size alone had the best accuracy. In addition, visualizing additional sightings data alongside threat level (Condition A) reduced task accuracy. This was true when comparing to both threat-level only configurations (Condition B and Condition C). The comparison between Condition A and Condition B makes this the most obvious. Threat-level is communicated using the same tiered color system in both. However, introducing the additional sightings metric via node size clearly distracts from the core task of threat detection.

Because Condition A led to the most incorrect selections for the largest threat, a potentially dangerous consequence is revealed. If information density (the amount of information communicated in a given space) is increased without care, the user may be easily misled. Such complex designs can also increase decision fatigue for users, negatively impacting performance. This was observed despite all participants being clearly pre-briefed on meanings of each property within each configuration.

In Condition A, many users tried using the size of the node to differentiate relative threat levels between all the red nodes. The group of 'malicious nodes' was easily filtered down to just the red ones, but picking between them raised issues. Only one of these nodes was the 'most' malicious, and participants could have relied on the tool-tips to confirm which one was. Instead, many participants' immediate assumptions were that larger nodes were more malicious, prompting them to make an incorrect choice. Given that node size equated to an object's sightings and not threat level, this should not have driven decisions. Prior exposure to Condition C (where size *does* equal threat level) was accounted for as a potential cause of this observation.

Thanks to counterbalancing in our study design, task order was unlikely to influence results. The key takeaway is that higher information density can be detrimental to investigative accuracy in graph-based visualizations. In such cases, participants were more likely to default to internal biases as to what they expect visual properties to communicate (i.e. big node = greater threat). The complexity of graph properties should be balanced carefully to avoid limiting the effectiveness of such tools.

Our tests were insufficient to statistically prove which configuration leads to the quickest task completion time. However, our results do show some limited evidence ($p<0.1$) that Condition C had the quickest threat identification for *small graph* sizes. This helps support the stronger evidence above regarding the limitations of using node color for threat level. Overall, node size was shown to be an effective indicator for threat, but misleading when used for separate metrics. If both color and size were to indicate the same metric (threat level) rather than separate metrics, perhaps that implementation would be most optimal. Sightings and other secondary information could alternatively be limited to less prominent elements, such as tool-tips. Visualization of these metrics may also be included as non-default options, toggled according to current task priority.

7.5 Study Conclusions

In this chapter, we described the primary frontend usability goals for CYBEX-P, our collaborative cyberthreat analysis tool. In particular, we discussed the need for effective threat visualization within network representations. Next, we outlined our methodology for researching visual properties of graphs that facilitate efficient threat identification. We then presented results from our user study conducted with participants from cybersecurity backgrounds. Finally, we discussed our findings and their implications for graph-driven cyberthreat visualization.

Multiple key determinations were reached regarding the core graph visualization. One such determination is that participants expected node color to be the most effective characteristic to convey threat. However, visualizations that relied on color rather than node size were actually less accurate in practice. We also found evidence to suggest they were slower to navigate. Another determination was that introducing additional graph variation for supplemental information significantly reduced threat detection accuracy. Notably, we also determined that cybersecurity professionals often had initial hesitations relying on visual representations of threat data. Our results lead to the following suggestions for future work.

First, small-scale graphs should consider representing threat-level of objects through node size to achieve reliable results. Additional research may prove whether this holds true across other graph sizes and investigative contexts.

Second, future work should be done to address the information density issue within threat intelligence graphs. This study has revealed some significant pitfalls when using common combinations of basic graph properties (like node size/color). Rather than approaching the visualization as the sum of all data that can be displayed, it should instead incorporate clear levels of priority depending upon context. More separated and hierarchical visual methods should be used for displaying additional uncorrelated dimensions (such as sightings within CYBEX-P and other non-mission-critical metrics). Each dimension of data should be prioritized depending on

the user's current task. Providing options to the user to switch between contexts should exist when user priorities are not universal. Further work is needed to investigate how to implement a model that combines the best of hierarchical and relational visualization. Cyberthreat investigations often have clear motivations and objectives. Thus, effective visuals should optimize for precision rather than pure density or generalizability.

Third, the design of cyberthreat analysis tools should cater to investigative caution, rather than try to eliminate it. Future designs should improve user confidence by embracing the importance of context. The interface should include obvious mechanisms for conveying why and how each object's threat is classified. Note that the event context feature on CYBEX relationship edges (demonstrated in Chapter 6) was not yet implemented at the time of this study. This is an example of a modification that provides supporting evidence as to why a node is related. Features like this are important for visual systems to be readily trusted and adopted.

Lastly, future studies should also focus on overall workflow optimization for graph-based security tools. More work is needed in this domain to streamline tasks like general graph construction and event tracing.

Security professionals see promise in using graph-driven tools compared to existing common-practice. However, these methods are not without limitations, which must be well understood to be truly effective. The findings of this study suggest tangible improvements that CYBEX-P can make to its visualizations in the future. The conclusions presented here can also be applied more broadly to any graph visualization approach used for cyberthreat analysis.

Chapter 8

Conclusions and Future Work

8.1 Summary of Contributions

This thesis has made two primary contributions that are highly relevant in an increasingly connected world. The first is a graph visualization approach for understanding cyberthreats within network topologies. Results from this method have implications within cybersecurity as well as the broader field of large-scale, interconnected data analysis. The second contribution is a fully-featured threat investigation tool that is centered around the aforementioned visual network graph structure. The CYBEX-P web application provides a human-centered interface for the emerging domain of cybersecurity information sharing. The software serves as a functional presentation of significant research and development outcomes. The result is a more intuitive solution that cybersecurity analysts can use to better understand threats to digital assets.

To achieve the original research goals, a methodical approach was undertaken. This began with establishing foundational background in the field of visual analytics. Early motivations for visualization were explored, and some historical milestones examined. Next, the current landscape for visualizing complex data was surveyed broadly. These basic underpinnings were then distilled into visualization considerations that are especially pertinent to cybersecurity. Some background in relevant cybersecurity concepts were also introduced. These included defining the field of cyberthreat intelligence, and describing emerging research in cybersecurity information sharing. Specifically, the CYBEX-P platform was introduced, which serves as

the data source for the primary contributions of this thesis. Lastly, tools and technologies were introduced that were critical for the research and development of the CYBEX-P web application.

Key related works were summarized next in order to identify insightful examples of cybersecurity data visualization. Existing work is examined within the context of distinct use-case classifications. Learnings from these works led to important determinations about the eventual web application design. There are different requirements for applications that are designed for general exploration versus specific goals. Because the CYBEX-P web application bridges the gap between these use-cases, examples of each one served as important inspiration. Additionally, current approaches to analyst workflows helped inform the design of CYBEX-P's investigation experience.

The primary motivation for the CYBEX-P web application was to enable advanced visual workflows so that analysts can more efficiently identify threats. This required a detailed understanding of the intended users, including their backgrounds, common behaviors, and existing challenges. Development required a hybrid approach that utilized both human-centered design and software engineering processes. A practical plan was executed that transitioned from the former to the latter as the software matured.

Use cases were modeled that were central to helping the application and visualizations achieve their high-level goals. From these, detailed requirements were specified, leading to a comprehensive feature set in the final product. A software architecture was presented that considers the application's back end, front end, and intended deployment.

The primary design challenges for the graph visualization included the display of identifying metadata, prominence and threat level, and entity relationships. Thoughtful consideration was also given to the implementation of the investigation workflow, including data manipulation and focused data observation. The solutions to all of these challenges were presented in deep detail, and were applied in example investigation scenarios.

Lastly, a user study was conducted to evaluate the application and some candidate visualization configurations. The application's default approach was compared against two other approaches. The alternatives tested the use of color and size exclusively for threat identification, omitting the sightings metric from the visual. Results indicated that using node size for threat rather than color yielded more accurate investigation conclusions. There was also some evidence to suggest node size allows for faster performance. Non-threat metrics, such as each node's number of sightings, hindered both task accuracy and completion time. It follows that graphs with less information density performed better for security-critical tasks. Qualitative feedback suggested some additional areas of improvement to strengthen analysts' trust in visual workflows.

Overall, investigators see promise in integrating these types of visual tools into their daily toolbox. Using a human-centered approach to cyberthreat analysis software helps extract insights from complex event data. The ultimate result is a more approachable way to keep our society and its collective data better protected.

8.2 Future Work

There are multiple avenues for future work to build on the contributions of this thesis. First, the user study results are expected to further inform future iterations of the CYBEX-P web application. Contextual event data has already been included as one such improvement. However, adjustments to the display of secondary metrics should be explored further. As mentioned in the study conclusions, this could include research on how to best combine elements of hierarchical and relational visualization. This thesis prioritizes crowd-sourced threat scores in its visualization; however, there are other network/security metrics to consider. The definition of a clear priority hierarchy for different investigation scenarios would allow for a generalized or customized visual model. More enhancements to non-graph workflow components should also be integrated. These include integrating alerting and reporting tasks, supporting more data sources, and facilitating real-time collaboration. Additional user studies that

include the full application feature set would help support continued development. This thesis has taken thorough first steps towards addressing several challenges; it also opens many doors to exciting future work in this promising space.

References

[1] Babel. URL: https://babeljs.io/docs/en/.

[2] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

[3] Sean Barnum. Standardizing cyber threat intelligence information with the structured threat information expression (stix). *Mitre Corporation*, 11:1–22, 2012.

[4] Nicholas Becker, Ayush Dattagupta, Eli Fajardo, Prem Gali, Bianca Rhodes, Bartley Richardson, and Bhargav Suryadevara. Streamlined and accelerated cyber analyst workflows with clx and rapids. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2011–2015. IEEE, 2019.

[5] Konstantin Berlin and Ajay Lakshminarayanarao. A simple and agile cloud infrastructure to support cybersecurity oriented machine learning workflows. *arXiv preprint arXiv:2002.11828*, 2020.

[6] J Clement. Us data breaches and exposed records 2019, 2020. URL: https://www.statista.com/topics/1731/smb-and-cyber-crime/.

[7] Kristin A Cook and James J Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2005.

[8] R Jordan Crouser, Erina Fukuda, and Subashini Sridhar. Retrospective on a decade of research in visualization for cybersecurity. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–5. IEEE, 2017.

[9] Michel Cukier. Study: hackers attack every 39 seconds, 2018. URL: https://eng.umd.edu/news/story/study-hackers-attack-every-39-seconds.

[10] José M de Fuentes, Lorena González-Manzano, Juan Tapiador, and Pedro Peris-Lopez. Pracis: privacy-preserving and aggregatable cybersecurity information sharing. *computers & security*, 69:127–141, 2017.

[11] Joseph Dien and Alecia M Santuzzi. Application of repeated measures anova to high-density erp. *Event-related potentials: A methods handbook*:57–81, 2004.

[12] Django. URL: https://www.djangoproject.com/.

[13] Django rest framework. URL: https://www.django-rest-framework.org.

[14] Docker. URL: https://www.docker.com.

[15] Lyndsey Franklin, Meg Pirrung, Leslie Blaha, Michelle Dowling, and Mi Feng. Toward a visualization-supported workflow for cyber alert management using threat models and human-centered design. In *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2017.

[16] Carrie Gates and Sophie Engle. Reflecting on visualization for cyber security. In *2013 IEEE International Conference on Intelligence and Security Informatics*, pages 275–277. IEEE, 2013.

[17] Joel Glanfield, Stephen Brooks, Teryl Taylor, Diana Paterson, Christopher Smith, Carrie Gates, and John McHugh. Overflow: an overview visualization for network analysis. In *2009 6th International Workshop on Visualization for Cyber Security*, pages 11–19. IEEE, 2009.

[18] Cristin Goodwin, J Paul Nicholas, Jerry Bryant, Kaja Ciglic, Aaron Kleiner, Cornelia Kutterer, Alison Massagli, Angela Mckay, Paul Mckitrick, Jan Neutze, et al. A framework for cybersecurity information sharing and risk reduction. *Microsoft*, 2015.

[19] Gunicorn. URL: https://gunicorn.org.

[20] VN Harikrishnan and GT Kumar. Advanced persistent threat analysis using splunk. *International Journal of Pure and Applied Mathematics*, 118(20):3761–3768, 2018.

[21] Jeffrey Heer. The Barnes-Hut Approximation. en. URL: https://jheer.github.io/barnes-hut/ (visited on 04/26/2021).

[22] Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky. A tour through the visualization zoo. *Communications of the ACM*, 53(6):59–67, June 2010. ISSN: 0001-0782. DOI: 10.1145/1743546.1743567. URL: https://doi.org/10.1145/1743546.1743567.

[23] Jeffrey Heer and Ben Shneiderman. Interactive dynamics for visual analysis. *Communications of the ACM*, 55(4):45–54, 2012.

[24] Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. Mastering the information age: solving problems with visual analytics, 2010.

[25] Valérie Lavigne and Denis Gouin. Visual analytics for cyber security and intelligence. *The Journal of Defense Modeling and Simulation*, 11(2):175–199, 2014.

[26] Matthew Lincoln. Adjacency matrix plots with r and ggplot2, 2014. URL: https://matthewlincoln.net/2014/12/20/adjacency-matrix-plots-with-r-and-ggplot2.html.

[27] I Scott MacKenzie. *Human-computer interaction: An empirical research perspective*. Morgan Kaufmann, 2013.

[28] Michael McNeese, Nancy J Cooke, Anita D'Amico, Mica R Endsley, Cleotilde Gonzalez, Emilie Roth, and Eduardo Salas. Perspectives on the role of cognition in cyber security. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 56 of number 1, pages 268–271. SAGE Publications Sage CA: Los Angeles, CA, 2012. DOI: 10.1177/1071181312561063. URL: https://doi.org/10.1177/1071181312561063.

[29] Martin Jose Hernandez Mediná, Cristian Camilo Pinzón Hernández, Daniel Orlando Díaz López, Juan Carlos Garcia Ruiz, and Ricardo Andrés Pinto Rico. Open source intelligence (osint) in a colombian context and sentiment analysys. *Revista Vínculos: Ciencia, tecnología y sociedad*, 15(2):195–214, 2018.

[30] Neo4j graph database. URL: https://neo4j.com/product/neo4j-graph-database/.

[31] Nginx. URL: https://www.nginx.com.

[32] Npm. URL: https://www.npmjs.com.

[33] Heldiney Pereira. Building Purposeful UI Using Pre-attentive Attributes. en, August 2018. URL: https://medium.com/design-at-zoopla/building-purposeful-ui-using-pre-attentive-attributes-9c5ee5dcc25c (visited on 10/13/2020).

[34] React. URL: https://reactjs.org.

[35] Farhan Sadique, Khalid Bakhshaliyev, Jeff Springer, and Shamik Sengupta. A system architecture of cybersecurity information exchange with privacy (cybex-p). In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0493–0498. IEEE, 2019.

[36] Matthias Schäfer, Markus Fuchs, Martin Strohmeier, Markus Engel, Marc Liechti, and Vincent Lenders. Blackwidow: monitoring the dark web for cyber security information. In *2019 11th International Conference on Cyber Conflict (CyCon)*, volume 900, pages 1–21. IEEE, 2019.

[37] Aneesha Sethi and Gary Wills. Expert-interviews led analysis of eevi—a model for effective visualization in cyber-security. In *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2017.

[38] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson, 6th edition, 2016. ISBN: 013438038X.

[39] Hossein Siadati, Bahador Saket, and Nasir Memon. Detecting malicious logins in enterprise networks using visualization. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2016.

[40] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2015. ISBN: 0133943038.

[41] Spiderfoot. URL: https://www.spiderfoot.net/.

[42] Splunk: cloud-based data platform for cybersecurity. URL: https://www.splunk.com/.

[43] Sqlite. URL: https://www.sqlite.org/index.html.

[44] Supervisor. URL: http://supervisord.org.

[45] Iman Vakilinia, Deepak K Tosh, and Shamik Sengupta. Privacy-preserving cybersecurity information exchange mechanism. In *2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–7. IEEE, 2017.

[46] Vis js. URL: https://visjs.org.

[47] Oli Warner. Django-multifactor. URL: https://pypi.org/project/django-multifactor/.

[48] Dana Zhang, Kotagiri Ramamohanarao, Steven Versteeg, and Rui Zhang. Rolevat: visual assessment of practical need for role based access control. In *2009 Annual Computer Security Applications Conference*, pages 13–22. IEEE, 2009.