IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 July 2023, accepted 7 August 2023, date of publication 14 August 2023, date of current version 18 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3304640

RESEARCH ARTICLE

Cyberattack Graph Modeling for Visual Analytics

MATEJ RABZELJ[1], CIRIL BOHAK[2,3], LEON ŠTEFANIČ JUŽNIČ[1], ANDREJ KOS[1], (Senior Member, IEEE), AND URBAN SEDLAR[1], (Member, IEEE)

[1]Laboratory for Telecommunications, Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia
[2]Visual Computing Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia
[3]Laboratory for Computer Graphics and Multimedia, Faculty of Computer and Information Science, University of Ljubljana, 1000 Ljubljana, Slovenia

Corresponding author: Matej Rabzelj (matej.rabzelj@fe.uni-lj.si)

This work was supported in part by the Slovenian Research and Innovation Agency and Government Information Security Office of Slovenia through the Research Project "Exposure of Modern Information and Communication Infrastructures to Cyberattacks" under Grant V2-2125 and the Research Program "Decentralized Solutions for the Digitalization of Industry and Smart Cities and Communities" under Grant P2-0425, and in part by the University Foundation of ing. Lenarčič Milan at the University of Ljubljana through their Ph.D. Scholarship Program.

ABSTRACT Cybersecurity research demands continuous monitoring of the dynamic threat landscape to detect novel attacks. Researchers and security professionals often deploy honeypot networks to intercept and examine real attack data. However, due to the volume and variety of the collected data, it is very challenging for security analysts to investigate the attacks, compare their characteristics and infer their potential connections. To this end, we propose a novel graph-based cyberattack model for storing, analyzing, and visualizing honeynet-captured attacks as the main contribution of our work. Our model enables attack graph analysis and presents the attack data analogous to the Cyber Kill Chain framework to enable intuitive visualizations. We construct the attack graph by decomposing the intercepted attacks into a set of unique entities (represented as nodes) and actions (represented as edges) and merge them into a global attack graph. We develop a user-centric, interactive attack analysis and visualization tool that leverages the proposed model to aid the heuristic cyberattack investigation. We describe the design and technical implementation of the developed model and visual-interactive tool in detail. Finally, we demonstrate the developed tools and validate the model in an analysis of real-world attack data captured on our own distributed honeypot platform. We use the attack model and (sub)graph visualizations to depict attack topologies, identify recurring attackers, and quantify detected malware types. We also leverage graph data science algorithms to uncover and rank malware distribution networks, reveal hidden links between the attackers, and cluster the attack entities to identify potential botnets.

INDEX TERMS Attack modeling, attacker links, big data, botnet detection, cyberattack, cybersecurity, cyberthreat intelligence, graph data science, graph modeling, honeynet, honeypot, malware distribution networks, threat modeling, visual analytics, visualization, cyberattack analysis.

I. INTRODUCTION

In order to secure online systems against known threats and discover new attacks, malicious behavior must be continuously monitored and actively studied. While known attack methods can be simulated in penetration testing scenarios to assess system security, novel attack techniques are more challenging to study. They can either be investigated during a post-incident forensic analysis on compromised systems or intercepted by specialized services, known as honeypots, to lure the attackers and collect data about their actions. Security researchers then analyze the acquired data to understand the mechanics of attack operations and attempt to model the attacks to aid their future detection and prevention. During the analysis, data visualization tools are often used to build visual attack representations and help reduce data complexity, discover prominent features, and reveal hidden links.

Honeypots are dedicated systems that resemble the functionality of production environments and emulate the behavior of real systems yet expose enough information or

The associate editor coordinating the review of this manuscript and approving it for publication was Claudia Raibulet.

computing resources to appear attractive to attackers [1]. They include specialized software to log the actions of the adversaries and alert administrators. Advanced honeypot systems also offer highly interactive sandbox environments inside which the attacker is monitored during the post-exploitation phase of the attack [1]. Experts then process and analyze the collected data to assess threats and learn about potential new attack mechanisms. However, data analysis can be challenging due to the ever-increasing volume of recorded events, most of which are performed by automated bots; this problem is further exacerbated in large distributed honeypot systems with large numbers of targets.

Since manual examination of massive amounts of recorded events is tedious and error-prone, automated attack identification using anomaly detection algorithms is often proposed [2]. This method relies on service or user models to distinguish between normal and abnormal events or user behavior and alert of a potential cyberattack. However, to build reliable attack detection models while minimizing false positives, a more profound, human-driven investigation is often required first to understand the objectives of the cyberattack and model its structure and mechanics [2]. Nevertheless, the volume and variety of the collected data often make it difficult for security analysts to investigate the attacks, compare their characteristics, or infer their potential connections. With this in mind and upon consultation with the stakeholders in the cyberdefence domain, we propose a novel attack graph model paired with a modern attack analysis tool suitable for heuristic cyberattack investigation using a visual-interactive approach. Our model stores honeypot-collected attack data in the form of a mathematical graph, while our attack analysis tool enables human-oriented visualization of logical attack structure, interactive discovery of cross-attack entity relations, and automation of attacker community detection using graph analysis algorithms.

The primary contribution of this paper is a novel graph-based cyberattack model for storage, analysis, and visualization of captured real-world attacks, either from a system of network honeypots or from real production systems. Our model stores the attack data in the form of attack entities, their metadata, and attacker actions and enables direct attack visualization on a graph that is consistent with Lockheed Martin's Cyber Kill Chain framework [3], depicting its chain of events on actual data. We construct the proposed model by decomposing the captured attacks into a set of unique attack entities and attacker actions. We model the individual attack entities as nodes and use edges to demonstrate actions or entity relations to form the attack graphs for each recorded attack. We then merge all individual attack graphs from multiple honeynet nodes into a single global attack graph to reveal data links between various attack entities, identify attack patterns, and uncover attacker networks.

The secondary contribution of our paper is a modern proof-of-concept tool leveraging our proposed cyberattack model for visual-interactive, heuristic analysis of the captured attack data. We implement the proposed attack model in a web-based cyberattack visualization and analysis tool, enabling interactive data exploration, analysis, and topological classification of individual attacks. We then validate the usefulness of the proposed cyberattack model and the developed visual analysis tool by performing a practical analysis of captured real-world attacks. We believe the results obtained using our tools and methods are significant as they demonstrate the power of interactive visual analytics designed with end users in mind, for the detection, analysis, and mitigation of potential cyberthreats.

The structure of this document is as follows. First, we review related work on cyberattack data collection, analysis, modeling, and visualization. We highlight the importance of continuous monitoring of the dynamic threat landscape and provide insights into the challenges of addressing a wide variety of cyberattacks with a unified model. We discuss the limitations of existing approaches to the design of salient, cognitively effective cyberattack visualizations and analysis tools. Next, we briefly describe our distributed honeypot system, which serves as the source of real-world data for our analyses. Based on that, we propose a novel attack presentation model tailored to real-world datasets and describe its design and implementation in detail. We focus on developing an extensible attack data model based on the Cyber Kill Chain framework and put emphasis on designing user-centric interactive visualizations to address the identified shortcomings of the previous works. Next, we illustrate typical security analyst use cases and use the developed tools and methods to demonstrate the identification and analysis of potential cyber threats. Finally, we conclude with the analysis results, discuss the encountered challenges and limitations of our approach, and outline prospects for future work in this area.

II. RELATED WORK

Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks. It is a broad field of research concerning many highly specialized topics. These range from social engineering to hardware vulnerability exploitation and demand expert knowledge and extensive research due to the complexity and interdependent nature of modern technologies. Therefore, existing cybersecurity knowledge is fragmented across the community of experts and practitioners in industry, academia, and government sectors [4]. For these reasons, it is challenging to address a wide variety of cyberattacks with a unified model, let alone depict them with a single visualization technique. However, several intelligence-sharing initiatives were started to address the growing number of cyber threats, and several security frameworks were developed for attack and defense modeling.

A. CYBERATTACK GRAPH MODELING

Threat and attack modeling are essential parts of cybersecurity research. Security professionals use threat modeling to estimate threats in their specific environments and design

corresponding cyberdefense strategies [5]. Attack modeling attempts to study the actions of threat actors further and understand the details of the attacks. Modeling, therefore, relies on the analysis of adversarial tactics, techniques, and procedures (TTP) [6], as well as their tools, to understand, detect and deflect the attacks. Models allow for the generalization of attacks to aid their detection and help penetration testers simulate or recreate them to assess the security of their systems. Several cyberattack models exist. Some are highly specific to individual domains, whereas others offer a high-level representation of general attack characteristics. Attack modeling techniques (AMT) include various implementations of attack graphs and trees, attack surface modeling methods [7], application of game theory in network security [8], stochastic cyberattack modeling [9], agent-based models [10], error-estimation-based denial-of-service attack modeling [11], differential-equation-based models [12], as well as the Diamond Model of Intrusion Analysis [13] and the Cyber Kill Chain among others. In line with the direction of our research, we primarily focused on attack graph modeling and the Cyber Kill Chain framework.

Lallie et al. [14] argue that graph-based attack modeling is the most popular choice in academic literature. The attack graph model was first proposed in 1998 [15] and has since become one of the most widely used security tools. Attack graphs and trees may simulate attack steps and paths used by the attackers to invade the network [16]. They may depict exploitation co-dependencies and attack steps [17] or display network topology and form a basis for evaluating attack vectors and assessing network security [18]. However, attack data acquisition can quickly accumulate large amounts of information, making it difficult to store and visualize it as a graph. Therefore, approaches toward graph modeling must be scalable [19] and use carefully crafted visualization to deliver salient information and reduce graph complexity [14], [20]. High-level attack representations, such as the Cyber Kill Chain framework, avert this issue by trading details on attack data for a generalized image of attack execution. The Cyber Kill Chain breaks down Advanced Persistent Threat (APT) attacks into seven crucial execution steps (Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command & Control, and Actions on Objectives) to provide a temporal insight into the course of the attack. It has a proven track record, and has been used by the United States Department of Defence for many years [5].

Researchers, therefore, classify modeling methods, security metrics, and attack graphs into different categories. Sheyner [21] describes scenario graphs, attack graphs, and network attack graphs. Idika [22] categorizes primary and secondary network security metrics and discusses condition-oriented, exploit-oriented, and condition-exploit-oriented attack graphs. Recent reviews of attack graph modeling offer even further categorization by graph types and properties (attack trees, fault trees, state graphs, exploit dependency graphs, logical attack graphs, multiple prerequisite attack graphs, Petri nets, Bayes attack graphs) [23] and graph algorithms used for data analysis (graph path algorithms, node importance sorting algorithms, Markov models, Bayesian networks) [14], as well as graph generation methods and their visual syntax [16].

Many studies focus on attack graph generation for cyberattack analysis and network security assessment. Noel and Jajodia [24] used attack graph analysis to aid optimal IDS sensor placement in a network and prioritize IDS alerts. They predicted all possible attack paths to reach critical assets and minimized the number of required sensors while maintaining coverage of the whole network. Kotenko and Stephashkin [18] used network hosts and attack actions to compose combined graph objects, such as routes and threats. They suggested a new approach to network security level evaluation using qualitative risk assessment and quantitative computation on the basis of Bayesian networks. Wang et al. [25] developed an attack graph-based probabilistic security metric for multi-step attacks exploiting multiple vulnerabilities. To do so, they considered the attacker's activity status and modeled the causal relationships between vulnerabilities to form an attack graph. Similarly, Noel et al. [26] addressed the limitations of models relying on individual security metrics and combined vulnerability interdependencies to analyze all potential attack paths through a network and provide a metric of overall system risk. Poolsappasit et al. [27] proposed a risk management framework using the Bayesian Attack Graphs (BAG) to quantify the chances of network compromise at various levels. They based their metrics on the Common Vulnerability Scoring System (CVSS) and used the acquired information to develop a security mitigation and management plan. Further, Sawilla and Ou [28] addressed the size and complexity of the attack graphs, which often prevented humans from fully comprehending the conveyed information. They did so by distilling the amount of information in a graph with the newly proposed AssetRank algorithm, a generalization of Google's PageRank [29], to assess the importance of the graph vertices representing vulnerabilities and attacker privileges. Huang et al. [30] also studied attack graph distillation to achieve a reasonable balance between the completeness of the graph and its usefulness. Their iterative method reduced the dependency attack graphs for moderately sized enterprise networks to an 85% smaller critical attack surface. Wang et al. [31] studied the use of attack graphs to develop a risk assessment metric for exploitation of yet unknown vulnerabilities (zero-day attacks). They developed the k-zero day safety metric and defined a zero-day attack graph composed of known and unknown exploits. Their metric was based on the count of the number of vulnerabilities required to be successfully exploited to compromise a network asset. A higher vulnerability count implied better security due to the lower likelihood of their presence.

Most recent attack graph modeling surveys were conducted by Zeng et al. [16], reviewing attack graph analysis

methods from the perspective of data processing in 2019, Lallie et al. [14] surveying attack graph and attack tree visual syntax approach in 2020, and Tayouri et al. [23] surveying attack scenario coverage of MulVAL [32] extensions in 2022. The authors of the latter conducted a systematic literature review of 938 papers and identified four new major attack graph generation tools in the last 10 years. Namely, in 2013, Holm et al. [33], [34] presented a graph-based tool for quantitative cyber security analysis of enterprises. Their publications reviewed the shortcomings of the earlier security estimation tools, such as NetSPA [35] and MulVAL. They described the developed Cyber Security Modeling Language (CySeMoL) and an attack graph tool tailored for modeling the critical infrastructure in power distribution networks. In 2021, Nadeem et al. [36] presented SAGE, an intrusion alert-driven attack graph extractor for cyber threat intelligence. Their proposal did not rely on vulnerability information for the derivation of attack graphs; rather, it constructed them from the intrusion alerts on a temporal and probabilistic basis. The solution was evaluated on open-source datasets in order to analyze the distributed multi-staged attacks. In the same year, Li et al. [37] constructed a knowledge graph extractor to collect different aspects of attack techniques and enhance attack behavior graphs from cyber threat intelligence reports. Their solution, AttacKG, was heavily based on aggregated data from threat intelligence feeds and the MITRE ATT&CK [38] platform. Moreover, Li et al. [39] recently worked on deep learning approaches to threat detection using system logs. They proposed DeepAG, a framework for the detection of attack sequences, and the construction of attack graphs for attack path prediction. Asvija et al. [40] used BAG to address the security of platform virtualized infrastructures in cloud environments. They used the reported attacks on virtualized systems to encapsulate all possible attack paths on the virtualization stack. They then modeled the identified threats as a Bayesian attack graph and derived the exploit probabilities based on CVSS scores. Similarly, Wang et al. [41] proposed a CVSS-based multi-factor risk assessment model using attack paths to model an attacker's capability and estimate the probabilities for successful vulnerability exploitation. Lastly, Malzahn et al. [42] presented automated vulnerability testing via executable attack graphs in a virtual environment to enhance the reproducibility and consistency of risk analyses.

B. ATTACK GRAPH VISUALIZATION AND ANALYSIS TOOLS

Graphs are used to visualize network elements and their connections [43], display attack paths [44], and visualize malware distribution networks [45]. However, their usability often decreases with the increasing number of network nodes [19]. Geospatial representations allow for IP address mapping to their geographical origin or destination and assist in the real-time visualization of cyber threats [46]. Scatter plots are used for port activity visualization [47], whereas parallel plots may reveal anomalous network traffic patterns [48]. Security visualizations, therefore, encompass a wide range of varied data presentation techniques and may entail fundamentally very different objectives.

Williams et al. [49] presented an interactive tool for simplified tracing of the attackers' paths and an intuitive understanding of the attack graph based on the underlying network topology. Their solution was based on NetSPA for graph generation and a Java application for its visualization. It scaled to networks with thousands of hosts. Xie et al. [50] proposed network security evaluation with two-layered attack graphs. They represented the host access graph in the upper layer and composed the lower layer of host-pair attack graphs. Their proposition addressed the intelligibility of graphs in large networks and focused on reducing the time for their computation. They presented an overall network security evaluation using grayscale images. Chu et al. [51] developed a tool named NAVIGATOR for the visualization of network assets, attacks, graphs, and operational recommendations. The tool allowed for network state visualization, as well as visualization of client-side, server-side, credential-based, and trust-based attacks. In 2015, Angelini et al. [52] presented PERCIVAL, a visual analytics environment for situational awareness and security event monitoring. Their visualization allowed for monitoring of attack progress and evolution. One year later, MITRE Corporation introduced CyGraph [53], a graph-based analytics and visualization system for cybersecurity. Their system captured incremental attack and vulnerability information, as well as security events within a network environment. It correlated the captured data and built a predictive model of possible attack paths to provide an overall picture for situational awareness in cybersecurity. The system was based on TVA/Cauldron [54] and offered layered visualizations of stored attack graphs. In 2019, Angelini et al.presented a new visual analytics solution for multi-step cyber attack detection [55]. The solution was aimed at assisting security operators, improving network security using attack analysis, and identifying suitable mitigations. Furthermore, Leichtnam et al. [56] proposed STARLORD, a data representation model and visualization approach for linked security exploration of heterogeneous sources in a 3D graph. Their solution used clustering to select data for visualizations and highlighted links between malicious events. Peryt et al. [45] used graphs to visualize malware distribution networks based on top-level-domain data collected from Google Safe Browsing reports. Tsigkas et al. [57] leveraged abstract graph representations to visually analyze spam campaigns, while Fowler et al. [58] proposed a graph and heatmap-based detection of security threats by visualizing a large volume of dynamic network data.

Attack graph visualizations were extensively reviewed by Lallie et al. [14] in 2019 and Ikuomenisan and Morgan in 2022 [59]. Lallie et al. [14] reviewed the visual syntax design theory and conducted a systematic literature review of 370 papers on attack graphs to quantify the employed visualization methods and visualized constructs. Their analysis identified 181 attack graph visual syntax configurations. Their reviewed design theories included the use of Bertin's

visual variables [60], Miller's law [61], The Primary and Secondary Notation Theory [62], The Physics of Notations [63], Petre's visual distance [62], and the Gestalt theories [64]. On the other hand, Ikuomenisan and Morgan [59] reviewed visualization practices and commonly used methods for the discovery and communication of attack patterns based on honeypot data [59]. They screened 218 papers using the PRISMA methodology and evaluated 37 papers with higher impact. They reviewed visualization frameworks for forensic honeypot data analysis and visual analysis pipelines for the transformation of raw input data into visual components. Ikuomenisan and Morgan [59] observed that a significant number of honeypot papers conducted summary statistics of static data (IP address, port, packet size), visually analyzed attack data only using simple graphical methods (line, bar and pie charts) and did not follow basic visualization principles and best practices in their use of color [59]. Similarly, Lallie et al. [14] revealed the absence of standardization, ambiguous semantics, and inadequacy of scientific approach towards the visualization design based on cognitive theories, concluding that many AMTs have not undergone an effective design process [14]. Ikuomenisan and Morgan [59] outlined rare implementations of customized visualization techniques (Hilber-curve for honeypot data analysis [65], honeypot driven cyber-incident monitor [66], bipartite graph visualization applied to IDS alerts [67], visualization of actionable knowledge for DRDoS mitigation [68]) as a positive development for pattern discovery and perceptual experience and called for further research in the field. Lallie et al. [14] noted that ineffective design leads to cognitively inefficient systems and attributed the fragmentation of efforts to the immaturity of the research field. They further indicated that methods for formal evaluation of the diverse conceptual models and the effectiveness of their visualizations remain scarce and limited [14]. They emphasized the shortcomings of some of the employed evaluation approaches, citing their statistical insignificance, unclear measures, as well as personal judgment [14]. Moreover, the authors of the study outlined the internal and external semiotic inconsistency as the fundamental problem of AMTs, noting that 34 authors represented identical constructs using multiple shapes in the same papers. In quantitative terms, Lallie et al. [14] report that the most common graph visualization design choices included the top-down representation of event flow (58.6%), no use of color (56.9%), no use of edge texture or line density (79.0%), no mechanisms to aid the perceptible visual distance (87.0%), simple entity labels in the form of text (38.7%), no representation of attack goal (78.5%), representation of precondition nodes for the attack using plain text (24.7%) or ellipses (17.5%), and representation of exploits using rectangles (28.7%) [14].

C. ATTACK DATA ACQUISITION

Cybersecurity professionals rely on intelligence-sharing platforms, data feeds, and open-source tools to acquire the latest information on cyberattacks. These include threat intelligence feeds with known indicators of compromise (IOC), malware sharing uniform resource locators (URL), malware files and their signatures, IP addresses of malicious e-mail servers, botnet information, and more. Besides data feeds such as Open Threat Exchange (OTX) [69], which often provide application programming interface (API) access, knowledge databases intended for heuristic incident analysis, such as MITRE ATT&CK and Malware Information Sharing Platform (MISP) [70], provide comprehensive information on the operation of malware, known threat actors, their methods, and tools. Furthermore, programs like Common Vulnerability Exposure (CVE) [71] and National Vulnerability Database (NVD) [72] identify, define, and catalog publicly disclosed cybersecurity vulnerabilities in a common format. At the same time, Common Weakness Enumeration (CWE) categorically lists software and hardware weaknesses, while open-source exploit databases [73] offer ready-made, proof-of-concept solutions for penetration testers and hackers alike. Most common formats, protocols, and description languages for cyber threat intelligence sharing include the Structured Threat Information Expression (STIX) [74], Trusted Automated Exchange of Intelligence Information (TAXII) [75], Cyber Observable eXpression (CybOX) [76], Malware Attribute Enumeration and Characterization (MAEC) [77], and YARA [78]. Besides external data sources, cybersecurity analysts also rely on internal data sources, such as system, firewall, and IDS logs, as well as network captures to discover malicious events [56].

However, to study the latest methods used by adversaries and examine the details of novel attacks, security research often requires interception of actual attacks and direct acquisition of their data. For this purpose, network honeypots are used to mimic domain-specific infrastructure and intercept incoming threats. They are commonly deployed as virtualized software appliances or implemented using containerization technologies [79] and are generally classified by the service they are emulating and the level of interactivity they provide [1]. Low-interaction honeypots typically only simulate services, such as Secure Shell (SSH) or File Transfer Protocol (FTP), but do not offer actual functionalities and operating system-level (OS) access [1]. While they are helpful for quantitative threat estimation, they cannot provide a deeper introspection into the attack's course and the attacker's post-exploitation actions (e.g., payload download, privilege escalation, actions on objectives). Therefore, high-interaction honeypots were developed as sophisticated cybertraps replicating production systems and allowing the attacker nearly unlimited target access [80]. However, such systems require significantly more effort for set-up and maintenance. They generate large quantities of data and require a thoughtful data acquisition, storage, and processing pipeline design.

Data processing becomes even more difficult when operating a network of honeypot systems (called a honeynet), where data heavily vary in volume, velocity, and variety. However, honeynets can mimic a wide array of vulnerable services,

replicate real IT (information technology) or OT (operational technology) networks and allow for detailed monitoring of the attackers' lateral movement across the deployed network services. Furthermore, a geographically distributed network of honeypots with a centralized data collection system allows for the observation of threat actors with a global presence, identification of their assets, and revealment of their malicious campaigns [81]. Bar et al. [82] studied attack propagation patterns using Markov chains on honeypot data in 2016. They performed complex network analysis on 167 million observed attacks and revealed patterns of attack correlations between the honeypots, identified central honeypots which propagated the attacks and profiled the attackers according to the attacking country. Studiawan et al. [83] focused on graph-based forensic analysis of web honeypots. They proposed attack-type analysis from honeypot logs represented as graph vertices and applied agglomerative clustering to categorize attacks based on PHP-IDS rules. Fernandez et al. [84] modeled malware-drive honeypots using recent IOCs to adapt honeypots for malware execution. Similarly, Wagener [85] studied self-adaptive honeypots for assessing attacker behavior. Durkota et al. [86], as well as Anwar et al. [87] studied honeypot allocation strategies using attack graphs and game theory. Inversely, Gao et al. [88] proposed a dynamic deployment strategy of virtualized honeypots based on intelligent attack path prediction to enhance their trapping capability. Kaâniche et al. [89] used honeypots to statistically model attack processes, while Ikuomenisan and Morgan [59] recently conducted a detailed systematic review of graphical visual methods in honeypot attack data analysis.

III. PROPOSED SOLUTION

We propose a novel graph-based attack modeling technique and present a new visualization tool developed for cyberattack analysis. The model structures the captured attacker actions, participating attack entities, their properties, and relationships as a graph and leverages visualizations to provide insight into the course of cyberattack execution and reveal potential attack patterns and attacker communities.

In the implementation phase, we develop a proof-of-concept solution based on web technologies that enable interactive visualizations of attacks. Its user interface is displayed in Fig. 1. It utilizes a graph-native database for data storage and retrieval and relies on our own distributed honeypot platform for direct attack data acquisition. It is intended for use by cybersecurity researchers and professionals.

The developed proof-of-concept tool can visualize individual captured attacks as a series of attacker actions analogous to the critical steps of the Cyber Kill Chain framework. It reveals individual attack topologies and characteristics and enables further data exploration using the visual-interactive discovery of links between the participating entities and metadata nodes. The implemented attack graph model enables node involvement detection in multiple attacks on the distributed honeypot network. Combined with the captured attack session data and enriched node metadata, it allows for the analysis of attacker hosts, external file servers, and malware files, revealing their potential hidden links, networks, and interdependencies. The web-based visualization tool can aid security analysts with charted statistical information on the aggregated attack data, identifying the most frequent returning attackers, most commonly deployed payloads, utilized distribution URLs, and similar details. It can plot the detected node communities and generate lists of malicious hosts, URLs, and file signatures, as well as export selected attack graphs into third-party open-source software for further network analysis.

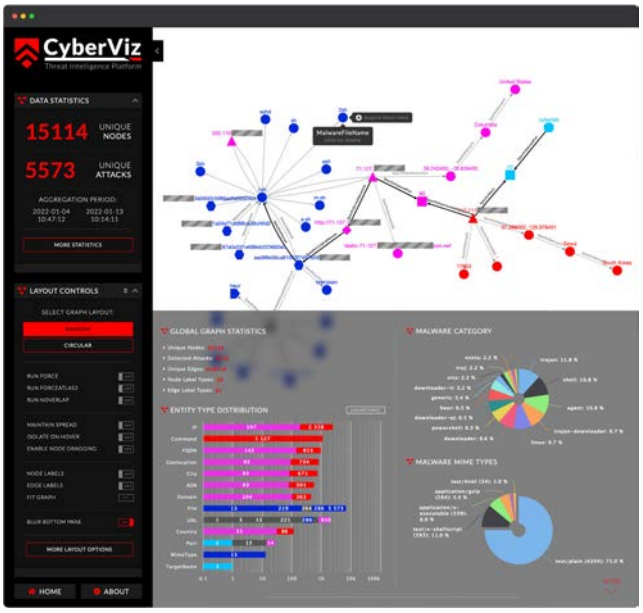


FIGURE 1. User interface of the developed proof-of-concept cyberattack visualization and analysis tool. The interface is divided into three general display sections: a layout and data controls pane on the left side, a graph display area on the top right side, and a details pane at the bottom. The graph display area demonstrates the attack browser, depicting a visualization of a particular cyberattack captured on the honeypot system. The details pane shows statistical information for all captured data in the selected time frame. More details are available upon further interaction with the interface elements. Part of the displayed data are redacted.

A. REQUIREMENTS ENGINEERING

Based on the reviewed literature, we identified both the good practices of the attack graph modeling and visualization design, as well as the opportunities for their improvement. We conducted expert consultations with the potential users of our solution in the cybersecurity domain. We consulted security operations center (SOC) operators, incident response analysts, and cybersecurity experts on their visual attack analysis approaches when dissecting a large number of attacks. We proposed an early idea of the interactive graph-based attack visualization tool with visual mapping of the attack data to the Cyber Kill Chain model to better aid the attack cognition. We performed task analysis and conducted cognitive walkthroughs with the potential users to identify

TABLE 1. Non-functional requirements of the proposed solution.

ID	Category	Requirement	Description
N1	Attack model	Common and extensible data format	The cyberattack model should store and represent participating attack entities and their properties in a common format. The format should be flexible enough to allow extension with new entities and properties on the fly.
N2	Attack model	Attack model scope definition	The attack graph should hold data on all captured attacks in the specific timeframe of the honeypot operation. It should model both the detailed course of execution of the individual captured attacks, as well as enable overview, comparison, and relationship discovery between all recorded attacks.
N3	Attack visualization	Attack intellegibility	Visualization of individual cyberattacks should be easily comprehensible by the users. It should depict attack stages and structure by portraying the crucial attack steps analogous to the Cyber Kill Chain model. The attacks should be visually decomposed and communicate entity relations.
N4	Attack visualization	Reduction of cognitive load and assuring semiotic consistency	Visualization design should be based on visual syntax theories and best practices in an attempt to reduce the amount of cognitive load on the analysts. Attack graph visualizations should make use of select visual variables (shapes, colors, labels, sufficient contrast, visual distance, size, weight, etc.) to form distinct visual constructs representing various entities, relationships, and attack steps. Visualized constructs should make use of a consistent design system to retain symbolic meaning throughout the whole solution.
N5	Attack analysis	Preconfigured visualizations and support for custom analysis using external tools	The proof-of-concept solution should offer multiple preconfigured visualizations of the attack data and display its statistics. It should offer interactivity to aid data exploration and analysis. It should also enable data export into generic graph analysis tools to allow analysts further control over graph projections and rendering.

their needs and investigate their workflow. We compiled a list of requirements for the design and development of our proof-of-concept solution. The following tables outline the top-level functional (Table 1) and non-functional requirements (Table 2).

We framed the scope of our work by outlining the necessary attack coverage of our model to meet these requirements. To do so, we focused on the three critical aspects of cyberattacks on internet-connected systems: network security, system security, and malware detection. We identified a subset of potential attacker actions for each of the segments and aligned them with the crucial attack steps represented by the Cyber Kill Chain framework. Therefore, we addressed the further design of our solution mainly in the context of the Cyber Kill Chain attack steps 3, 4, and 5, as shown in Table 3.

IV. SYSTEM ARCHITECTURE

The overall system is composed of two major platforms: the honeypot-based data acquisition platform (CyberLab) and the interactive attack visualization platform (CyberViz). Both systems consist of multiple interconnected microservices that collect, store, transform and visualize cyberattacks. This paper only briefly reviews the actual data acquisition platform and primarily focuses on the design and technical implementation of the attack modeling and visualization. The high-level system architecture is depicted in Fig. 2.

A. DATA ACQUISITION PLATFORM

CyberLab is a distributed, high-interaction honeypot platform. It exposes inadequately protected SSH honeypots at its forefront and lures the attackers into fully monitored and almost full-featured Linux containers. The platform is distributed over multiple points of presence on a wide range of IP addresses. It supports simultaneous monitoring of multiple attackers and logs incoming and outgoing network activity, records complete terminal sessions, monitors filesystem changes and snapshots the attack evidence. Acquired attack data are stored in various database systems, processed, and exposed via internal APIs.

Captured data are then imported into the CyberViz system in JSON format via an internal API and enriched with open data from publicly available cyber intelligence feeds. The resulting data structure hierarchically describes each intercepted cyberattack with key-value records. Its data values include attack identifier and timestamp, target IP address and entry vector (port), attacker's IP address and its detailed information, such as autonomous system number (ASN), the fully qualified domain name (FQDN), and complete IP geolocation data (city, country, latitude, longitude). Other data fields also include a complete log of commands from interactive SSH sessions, a network log of outgoing and incoming connections to and from (malware distribution) servers and their exact IP address, port and protocol information, hostnames and URLs, as well as a list of service and filesystem changes, together with file details, such as file hash, file name, Multipurpose Internet Mail Extension (MIME) type, and malware scan results, including classification by malware detection engines.

B. CYBERATTACK MODEL

The reviewed literature on attack graph modeling techniques comprises numerous fundamentally different approaches. Their common goal often includes the estimation of the general level of network security or identification of (potential) attack paths within the system. We observed that many of the proposed attack graphs focused on modeling states of known network assets (e.g., hosts, security controls, and network devices under the operator's control). Oftentimes the models' primary represented constructs included the cyberattack preconditions, exploits, and postconditions [14]. The vulnerabilities of such systems were often known in advance, whereas the information of used exploits was retrieved from external intelligence feeds. This allowed the researchers to focus on the model of their network and investigate attacker

TABLE 2. Functional requirements of the proposed solution.

ID	Category	Requirement	Description
F1	Attack model	CyberLab data import compatibility	The model should be able to ingest raw captured attack data from the developed CyberLab system and convert it into the proposed attack graph model, stored in a graph-native database.
F2	Attack model	Import data supplementation and parameter calculation	The ingested attacker and server data should be enriched with detailed IP information (GeoIP, ASN). Transferred files should be accompanied by malware category tags and attributed with mime types. Scripts should be decomposed into individual commands, while URLs should be decomposed into URL parts (protocol, domain, port, path, anchor, etc.). DNS resolution and reverse DNS lookups should be performed for all FQDN and IP entries. Filename similarity should be calculated on import and attributed to relevant node relationships. Select global graph projections should be prebuilt during import time to enable community detection. Member nodes should be attributed with community IDs.
F3	Attack visualization	Implementation of attack graph visualization and its controls and features	Graph visualizations should employ interactivity (respond to clicking, hovering, panning, scrolling, etc.) to reveal detailed data, such as entity and relationship properties and labels, without unnecessary overplotting. The solution should offer visualization controls over graph layout and animations, selection of predefined graph projections, and time-consecutive browsing between the captured attacks.
F4	Attack visualization	Visualization and display of non-graph data	Attack graphs should be complemented with non-graph-based data, including the charted statistical information on all captured attacks and malware type distribution, textual representation of node and edge properties, as well as attacker shell sessions and descriptions of the used symbols and notation. The data should be displayed dynamically while maintaining graph visibility and interactivity. Charts portraying top-k entities and features should be used to enable prompt insight into the captured sample.
F5	Attack analysis	Implementation of advanced graph analysis features	Graph analysis tools should be implemented in multiple distinct control panels, each addressing a specific set of functionality. Graph layout controls panel should offer application node layouts, such as simulated attraction and repulsion between nodes with weighted relationships to reveal frequent actions and visually cluster similar data. It should control node and edge label display, as well as node spread control on canvas to maintain adequate visibility. It should also control node dragging and graph fitting to active canvas size. The data controls panel should offer a selection of graph data for visualization. It should enable consecutive browsing between specific attack graphs and display the global attack graph, as well as its subgraph projections. Predefined projections should isolate crucial entities and reveal their relations without overplotting graphs with unnecessary data. Additional graph analysis features should be displayed on a separate pane and allow node searching by attribute keywords, enable clustering by arbitrary attribute, visualize clusters node clusters with color, and display node ranks.
N6	Attack analysis	Advanced graph analysis functionality on global attack graph	The global attack graph should implement search functionality to reveal entities matching arbitrary search queries. Searching should be enabled by any entity property. Graph filtering mechanisms should be implemented to allow the filtering of nodes and edges based on their labels. Attack data and metadata should be displayed upon node or edge selection and linked with additional views. Specific attack terminal sessions should be visualized as text using shell commands and displayed as nodes on the graph. Select attacks should be visually highlighted upon hover or selection on the global attack graph. Entity IP addresses should be clickable to reveal IP activity history.
F7	Attack analysis	Data export functionality	The solution should offer data export functionality to transfer the attack graph into external network analysis tools or generate raw TSV files. It should also provide the option to extract a list of detected malicious IP addresses, domains, and malware data for use in network and application-layer firewalls. Exported graph data should contain node assets to maintain semiotic consistency between graph visualization tools.

actions in its context, assessing its security, discovering attack paths, and raising intrusion alerts. However, we observed a shortage of studies that focused on obtaining attack session data and modeling it in such a manner to allow for the investigation of the attackers' behavior outside of the controlled perimeter, enabling the discovery of attackers' network assets and mutual relations, as well as revealing their coordinated cyberattack campaigns.

Therefore, we propose a new attack graph model tailored to the centralized data collection pipeline of a globally distributed honeypot network. The model is capable of representing the individual attack details, as well as inspecting the attacks' common denominators (e.g., shared data, metadata, and actions) for uncovering attacker networks and relations. We enrich the ingested data with externally obtained network information, file metadata, and unique attack identifiers and scan the downloaded malware files for known malware signatures. Like similar works, we do not complement our model with specific vulnerability information from the MITRE ATT&CK platform or the CVE program since knowledge of the exposed vulnerabilities is already built into the design of our honeypot system.

However, we do generalize the captured fragmented attacker actions into a series of macroscopic steps of the Cyber Kill Chain framework to provide a temporal overview of the attack. We base our unified model's technical implementation on scalable graph technologies, proven functional by Noel et al. [53], [90]. We borrow and extend data modeling approaches from Leichtnam et al. [56] We draw inspiration for the analysis of malware distribution networks from Peryt et al. [45]. We employ attack propagation analysis and honeypot platform validation approaches from Kaâniche et al. [89] and Bar et al. [82], We leverage clustering ideas described by Studiawan et al. [83] and aid graph

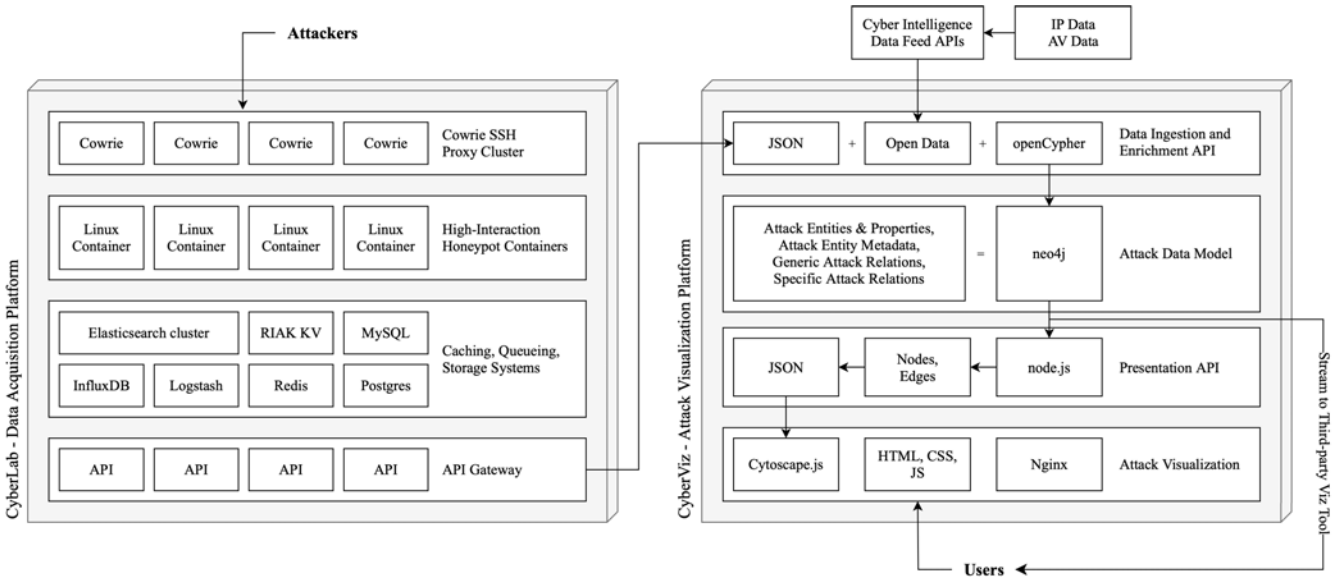


FIGURE 2. High-level architecture of CyberLab and CyberViz platforms. The arrow on the top left of the figure indicates the honeypot attack ingress point, arrows connecting system components denote data flow within the solution, and the arrow on the bottom right shows the CyberViz front-end interface exposed to users. Stacked layers display system topology, whereas individual entities represent specific technologies and data formats.

TABLE 3. Coverage of the Cyber Kill Chain model.

Step	Step description	Implementation strategy
3	Payload delivery over the network	We addressed the attack and payload delivery stage in terms of network security. We proposed investigating attack origin, attacker and target networks, payload servers, and file transport mechanisms. We suggested supplementing the collected network information with metadata and integrating it into the attack graph model for analysis and visualization.
4	The exploitation of the system or service	We addressed the system security by analyzing the attacker's actions on the high-interaction honeypots. We proposed enumerating the attacker's initial access (exploitation) mechanism and logging their actions on the system. We suggested visualization of the attacker commands and attack sessions.
5	Malware installation	We allowed arbitrary file execution on honeypots and suggested the automatic submission of the captured samples to malware detection engines. We categorized the malware files based on their origin, mime type, filename, hash, and detection results (malware tags). We proposed visualization of the detected malware campaigns, file reuse, and filename patterns.

dilution using PageRank algorithm as shown by Sawilla and Ou [28]. We apply the constructed model to our dataset of captured attacks on SSH honeypots and further extend the above proposals with a proof-of-concept visualization system for interactive attack exploration and visual analysis. We also carefully address the shortcomings of the preceding visualization solutions identified by Lallie et al. [14] and employ the most common visualization channels and graphical visual methods for honeypot attack data analysis as reviewed by Ikuomenisan and Morgan [59].

We position the newly proposed data model for structuring and presenting the acquired attack information as the principal contribution of our research. We demonstrate the use of our model to directly represent the attack topology and leverage graph databases to maintain consistency between the data storage and presentation layers. We implement the model by mapping the crucial (unique) entities participating in the attack (e.g., attacker's IP address, target IP address, malware server's IP address, target port, and malware files) into individual nodes and their properties, and model the identified attacker's actions (e.g., a new SSH session, an outgoing network connection) as node relationships. This allows us to describe, access, and visualize the individual attacks as connected graphs rather than storing them as independent objects with potentially duplicate data and metadata.

We deliberately place select metadata information (e.g., IP address geolocation information) on the graph, rather than assigning it to nodes as properties, to enable path-based querying and aid visual exploration of the nearby connected nodes. This complements existing graph edges denoting the attacker's actions with new edges representing metadata relationships. We further generalize the attack data and attackers' actions by complementing the specific stored relationship types and labels with their generic counterparts. To do so, we merge specific node labels (e.g., individual attack target, attacker's IP address, malware server's IP address) into generalized categories and nodes (e.g., all honeypot targets as one node, all IP addresses as one category) and connect them to a global attack graph encompassing all captured attacks. We also link multiple relationships (actions) into virtual higher-level attack steps, following the Cyber Kill Chain framework (e.g., merging the attacker's connection to

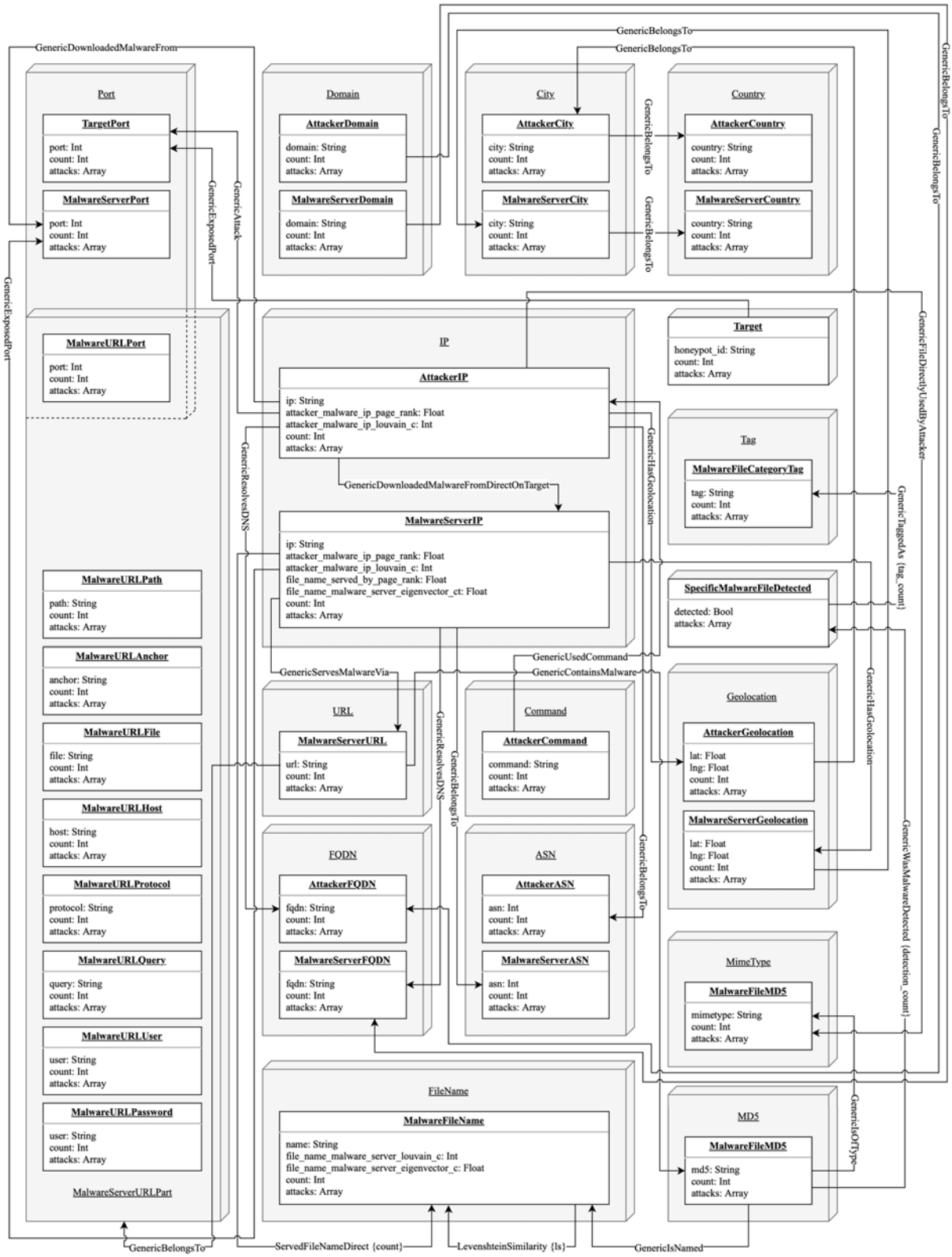


FIGURE 3. Proposed graph data model of an individual cyberattack. The model is displayed for a single attack, while its unique entities are further linked into a global attack graph. The figure only displays generic node relationships to reduce overplotting. Node data and relationship properties are listed. Nodes are displayed with specific labels and grouped into categories. Categories are also represented using (generic) node labels to aid querying and visualization. Identical entities are merged into a single node and linked using relationships attributed with corresponding attack identifiers.

the target via SSH and their malware retrieval via HTTP into a single payload delivery stage). We attribute the unique attack identifier to each specific relationship and append an array of applicable attack identifiers to the generic relationships. Each of the individual cyberattacks is, therefore, a subgraph consisting of nodes and specific attack relationships in the global attack graph. This allows us to observe individual entities in the global attack context and enables the identification of node interactions between multiple attacks. The finalized model includes 34 different node labels, 15 node category labels, and 60 relationship types. It symbolizes our distributed honeypot system as a generic root node connecting all captured attacks. Actual attack targets in individual attacks are represented using unique honeypot identifiers to conceal their true IP address. This design enables simple path traversal across the root target node when analyzing multiple attacks. All attack entities of the model are merged into the graph for each newly imported attack session. This means that previously unseen entities are created as new nodes, whereas already existing entities are linked to the unique identifier of the imported attack. The representation of the proposed attack graph model is depicted in Fig. 3. A sample of the graph-building procedure is shown in Fig. 4.

With further simplification and applied visualization layouts, the proposed model offers a direct representation of the individual attack's structure and its phases of execution. It allows for visual attack classification based on the (sub)graph topology and enables the straightforward development of interactive visualizations. Further, the graph data model allows for applying graph data science algorithms, such as community detection, PageRank, and link prediction, directly to the global attack graph to reveal latent network characteristics.

We implemented the model using the neo4j graph database [91] and developed the data ingestion and transformation API in Python to convert the enriched JavaScript Object Notation (JSON) objects into openCypher [92] graph query language (GQL) statements. We used dedicated graph projections for running graph algorithms and additional virtual relationships for visualizations.

C. VISUALIZATION DESIGN

Authors of the Systematic Review of Graphical Visual Methods in Honeypot Attack Data Analysis [59] observed that scientific researchers often consider data visualization a trivial task and do not follow data visualization best practices, resulting in the ambiguous visual communication of information. Similarly, A Review of Attack Graph and Attack Tree Visual Syntax in Cyber Security [14] reported that many AMTs appear not to have undergone an effective design process and often overlooked the cognitive value of graphical models during their design. Therefore, we approached the visualization design in an attempt to address these findings and the identified solution requirements. We reviewed the relevant visual syntax design theories and design practices in an attempt to produce a cognitively efficient system.

Yang et al. [93] and Ji et al. [94] suggested approaches to effective network traffic visualization using a graph-based representation of complex structures, as well as using data filtration and dimensionality reduction techniques as reviewed by Sorzano et al. [95]. They also proposed the use of multiple coordinated views for better insight into the data. Weissgerber et al. [96] and Ikuomenisan and Morgan [59] suggested basic charts (such as line, bar, and pie) in security visualization be complemented with (interactive) graphical constructs offering higher data density, visual effectiveness, and better visual expressiveness to effectively communicate hidden patterns in data. Keim et al. [97] defined the functional steps of a visual analytics model, consisting of data processing, information visualization, pattern discovery, and knowledge generation, reinforced with visual-interactive data exploration. Bertin [60] described visual variables, entailing position, size, shape, value, color, orientation, and texture, to build the foundation of visual syntax. Miller [61] discussed the limits of human short-term memory in relation to the number of visual stimuli. He proposed an optimal number (7 ± 2) of objects and their properties (e.g., shapes, colors, edges) for effective visual communication. The Principle of Primary and Secondary Notation built a distinction between variables forming the generic diagram structure and objects exhibiting its relationships and aiding the perception of the observer [62]. Visual distance referred to perceptible steps that helped to distinguish between objects in a diagram, aiding object recognition and concept interpretation [62]. The Gestalt principles of visual perception proposed seven factors contributing to the effective diagram design, including simplicity, proximity, similarity, figure-ground relationship, direction, continuity, and closure [64]. The Cognitive Dimension of Notations [98] defined 14 cognitive dimensions, including consistency, abstraction, the closeness of mapping, the difficulty of mental operations, hidden dependencies, and viscosity. The Physics of Notations set the basic visual syntax principles and formed guidelines for the implementation of the design variables [63]. Most notably, these principles included object-to-concept mapping without redundancy and ambiguity (semiotic clarity), dual coding using text and visual notations, the cognitive fit of the developed model for its intended audience, visual expressiveness, and semantic transparency. Moreover, additional visualization design considerations included the logical depiction of events and information flow on the diagrams [14], as well considerations regarding the shape and color selection, leveraging their suggestive powers and perceptive links with hazards [99]. Lallie et al. [14] argued shapes should be used as the primary means of communicating information as they are the primary visual variable for aiding object recognition [63] and can capture important phenomena more powerfully and succinctly than words [100]. They further discussed the use of specific shapes and their suitability for the accommodation of textual labels.

Following the reviewed design theories and visualization practices, we approached the solution design in accordance

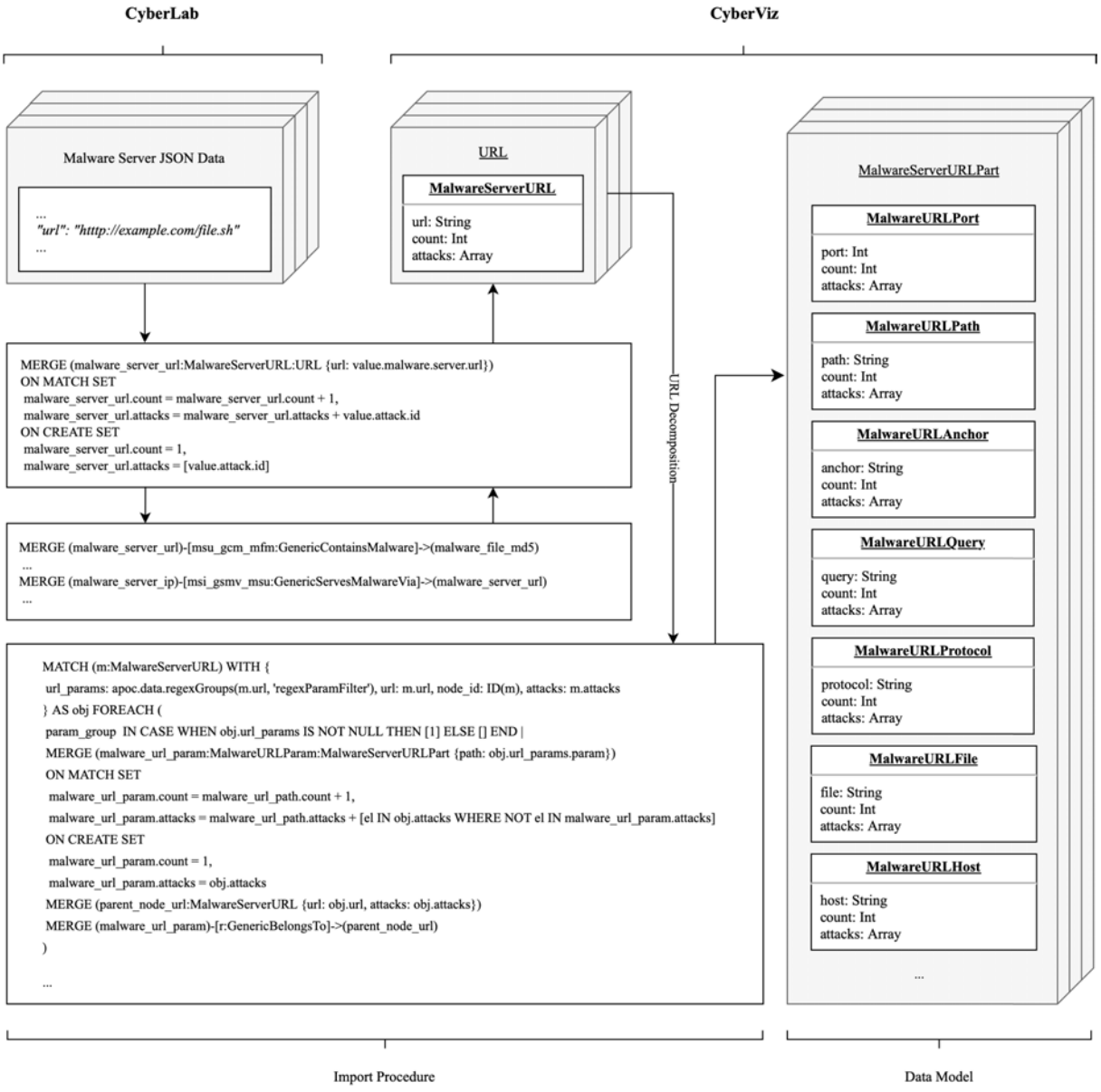


FIGURE 4. Visual representation of data import (graph building) procedure. The figure displays pseudocode samples for the creation of MalwareServerURL nodes and their relationships, as well as the decomposition of URL strings into individual parts, each represented as its own MalwareServerURLPart node. The data are ingested from CyberLab API and merged into unique nodes, as shown on the top left side of the diagram. The nodes are linked with other graph entities and attributed with unique attack identifiers. URL decomposition is performed to enable the identification of attack similarities based on URL parts and patterns.

with the identified requirements. Based on the conducted consultations with cybersecurity experts, we identified two crucial visualization use cases for our attack analysis tool:

1) Visualization-based workflow for the analysis of the global attack graph.

   The tool should allow for visualization of the global attack graph model comprising all captured attacks. It should enable the analysis of entity involvement in multiple attacks to reveal potential attack patterns and identify node clusters and their hidden links. It should define and visualize graph feature projections based on select node relationships and calculated metrics. It should enable node searching and visualization of search results on a graph by matching text-based queries to any selected node properties. It should allow for configurable visualization of node clusters based on node attributes, support cluster coloring, and isolation, as well as display cluster sizes. It should allow graph filtering based on node and edge labels to allow for configurable preview without overplotting. The tool should also support interactivity between the main graph display and additional view panes portraying node and edge (meta)data. It should

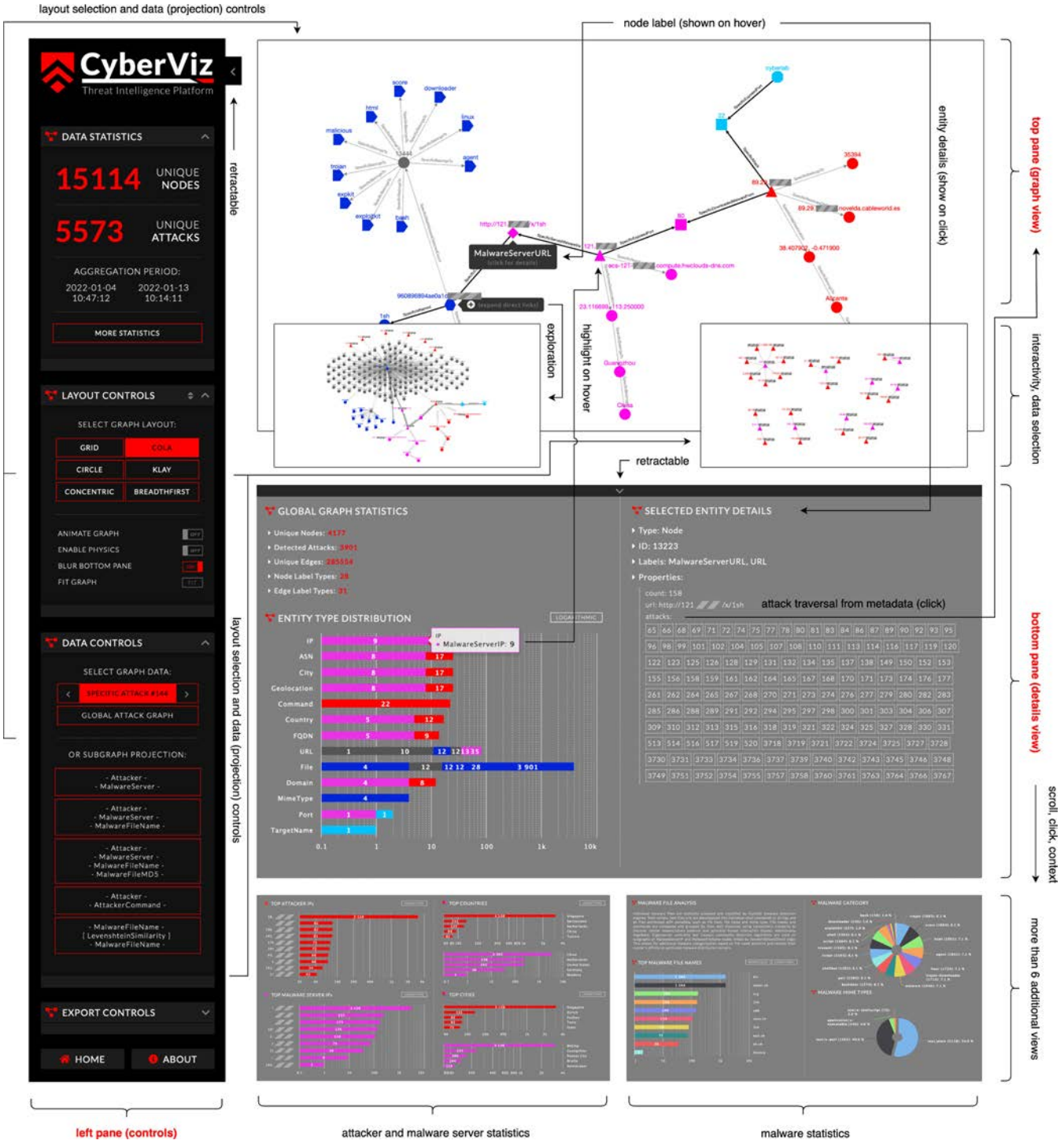


FIGURE 5. Design of the attack visualization tool. The figure displays the main components and features of the proposed proof-of-concept visualization solution. The tool consists of three main panes (labeled in red): the controls pane on the left, the graph visualization display pane on the top, and the details view pane on the bottom. The controls and details pane is retractable to enable fullscreen graph visualization. The controls pane allows the selection of graph data (individual attacks or global graph projections) and its layout, as well as allows for easy data export. The bottom pane displays node and relationship details and statistics. The visualized attack graph is interactive. It displays node labels and graph expansion context menu on hover and expands the graph by visualizing directly related nodes from other attacks on click. Upon node or relationship selection, their details are displayed in the details pane at the bottom. The details include node labels, appearance count, and entity properties (including the calculated metrics). Some of the displayed elements, such as the attack involvement list, are interactive and directly affect the visualization in the graph display pane. This is depicted with black arrows linking crucial elements and views via user actions. Additionally, scrolling the bottom (details) pane reveals additional widgets containing specific attack information and global attack graph statistics (e.g., malware file data, attack origin statistics, top returning attackers, etc.). Sensitive attack data are redacted from the figure.

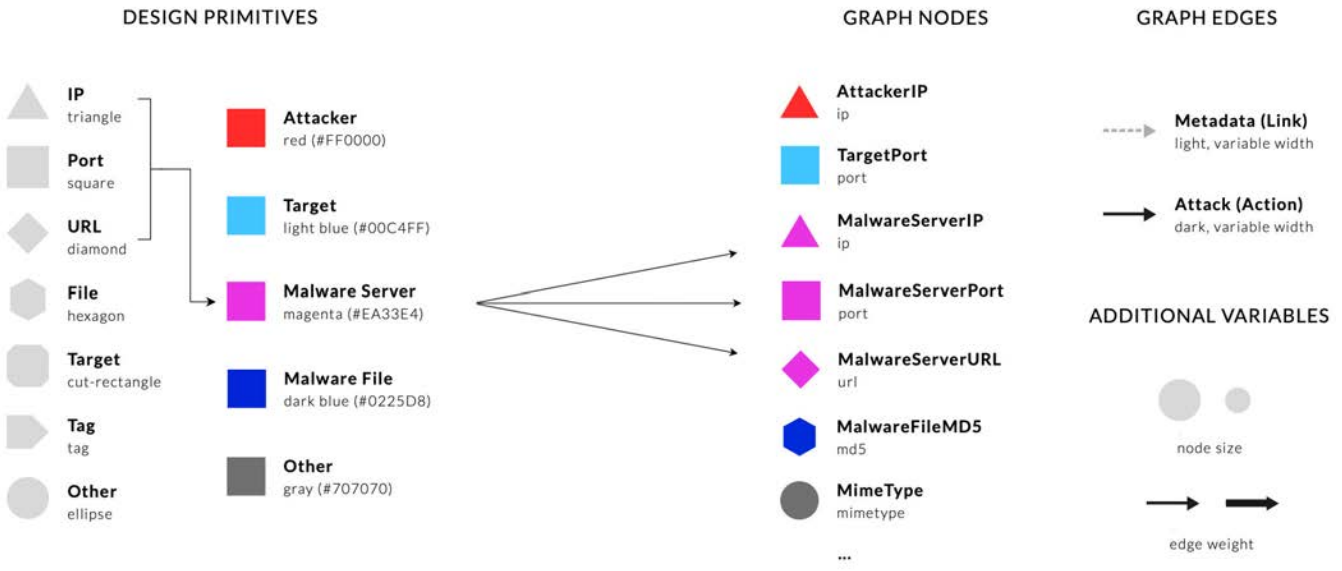


FIGURE 6. The implemented node design comprises shapes and colors to represent generic data types and entity roles. The combined design primitives form easily distinguishable individual attack graph entities. The figure only depicts a sample subset of nodes composed of the design primitives. Node relationships are depicted using dotted edge lines to display metadata links and solid lines to display attacker actions. The defined node design and color scheme are used throughout the complete solution, maintaining semiotic consistency across all graph projections and charts with statistical data. Additionally, select projections to make use of node size and edge weight to represent their relative importance based on node ranks (e.g., page rank) and relationship property weights (e.g., number of connections, node similarity score, etc.).

enable text-based previews of attack shell sessions and visualization of isolated attacks while previewing the global attack graph. Captured IP addresses should be linked to the CyberLab platform to allow users to display the history of their activity and show their malicious score. Lastly, the tool should enable easy model export from the graph database to general-purpose graph analysis tools for external network analysis. It should also enable the export of the detected malicious IP address and malware details. The exported graph data should be complemented with node assets to maintain visualization consistency across all visualization environments. Extracted entity data should be exported in common formats to allow its use for easy configuration of network protection appliances (e.g., blocking service access for exported attacker IP addresses on network firewalls and restricting user access to known malicious URL endpoints on IPS systems).

2) Visualization-based workflow for detailed analysis of individual attacks.

The tool should enable navigation between the captured attack sessions and depict the individual attacks as graph structures to visually portray their phases of execution analogous to the Cyber Kill Chain framework and reveal their characteristics. It should display attack entities using a semiotically consistent node design and utilize interactivity to reveal node and relationship properties in the additional overlayed view panes. The visualization should enable node relationship exploration across multiple recorded attack sessions and implement elementary graph layout controls. The main graph should be complemented with charted statistical data on all captured attacks within the specified timeframe to better paint the context of the specific attack under analysis.

We build upon these findings to design a proof-of-concept interactive tool for the visualization and validation of the proposed attack graph model. We position graph visualizations as the primary means of communicating attack data and follow the proposition of Ji et al. [94] to implement additional coordinated view panes and display (textual) entity properties and details. We borrow the ideas of Ikuomenisan and Morgan [59] and complement graph data with charted statistical information on recorded attacks. We make use of the proposed dimensionality reduction techniques using feature projections [95] and introduce visualization controls to allow the selection of predefined graph projections and offer control over the visibility of graph features. We implement node, edge, and canvas interactivity, capturing hover, scroll, click, and drag events to allow for the accommodation of label data on hover, visual-interactive data exploration on click, and view control using scroll and drag gestures. The individual components of the proposed visualization tool and their features are displayed in Fig. 5.

We designed our proof-of-concept visualization solution in accordance with the identified use cases. Its user interface design is displayed in Fig. 5. It consists of three main display and control panes: the visualization control pane on the left, the graph display pane on the top, and the details view pane on the bottom of the interface. The control pane allows the selection of visualization to show in the graph display pane

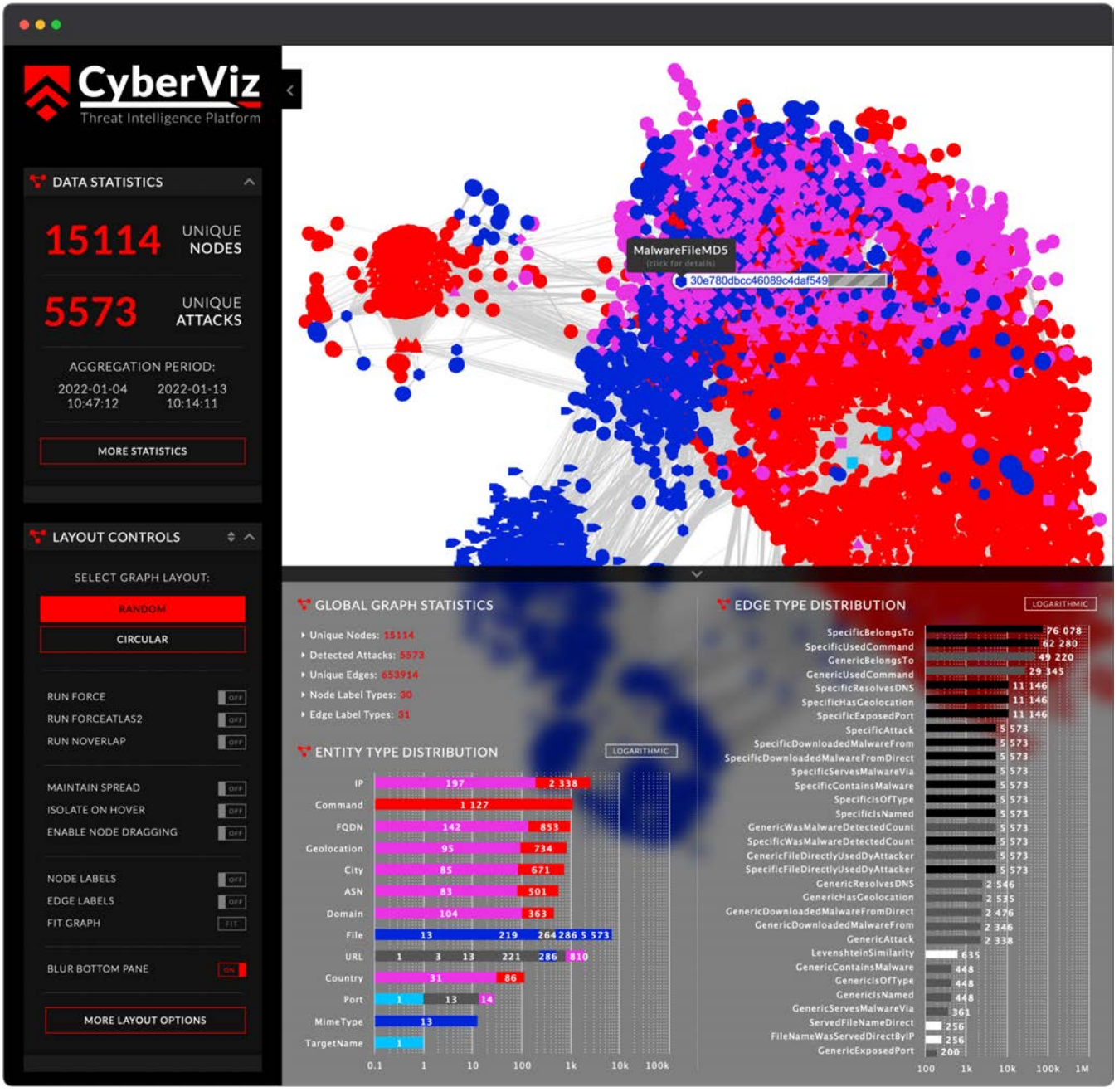


FIGURE 7. CyberViz solution displaying the global attack graph and its details. The graph is initially displayed without a node spread. Node and edge labels were disabled to conceal any personally identifiable information (IP addresses, domain names, command parameters, file names) and reduce label overplotting. The user is hovering over a MalwareFileMD5 node, revealing its hash. The bottom details pane displays the basic information about the global attack graph and its node and edge type distributions. Specific node and edge labels in distribution charts are hidden until the user hovers over their corresponding bar. Additional graph data are revealed by scrolling the bottom panel.

and enables control over its layout and features. Data control options enable visualization of the global attack graph and its five predefined projections (use case 1), including attacker IP to malware server IP address mappings, visualization of the identified malware distribution networks, including payload file names and their MD5 hashes, and the display of the attacker IP addresses, their used commands and command similarities. Additionally, data controls allow isolated specific attack subgraph display (use case 2) and allow browsing between the consecutive captured attacks. The graph display pane on the top shows the selected attack graph visualization. The display defaults to a force-directed, constrain-based layout for attack subgraph display, whereas live physics and animations are initially disabled when visualizing global graph projections to aid browser performance. Layout selection, graph physics, and animations can be manually

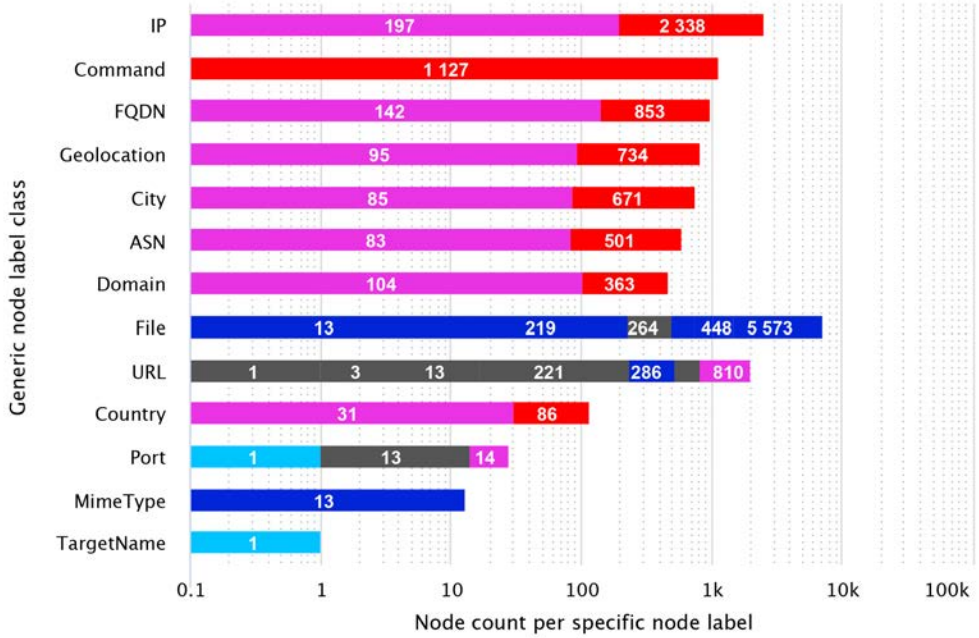


FIGURE 8. Entity type distribution in the global attack graph. The chart portrays the statistical data on generic and specific node labels and their distribution in the dataset. The global network graph for the captured timeframe includes 15114 unique nodes and 5573 detected attack events. The data series color coding in the chart indicates the specific label's role in accordance with the presented node design. Most notably, the chart allows the user to identify the IP address distribution skew, revealing 2338 unique attacker IP addresses and only 197 unique malware IP addresses in the global graph.

controlled using the control pane. The default layout uses node relationships and parameters, such as relationship weights, action frequency, and node similarity, to position nodes on canvas and allow for visual clustering of graph components to aid the human perception of the data. The visualization layout controls pane is dynamically populated with additional controls when switching between a specific attack and a global graph or its projection display. These controls include the implemented static layouts, continuous force simulation algorithms (e.g., Noverlap, ForceAtlas, ForceAtlas2), control of node and edge label visibility, graph fitting to view pane size, hover-based node isolation mode and node dragging support. The displayed graph is interactive and shows node labels and graph expansion context menu on hover. Users can dynamically extend the visualized attack graphs by clicking on the individual node's context menu item to visualize its directly connected neighboring nodes from the global attack graph and enable the discovery of adjacent malware files and connected attackers and servers. Node selection displays its details in the bottom pane and reveals its involvement in other cyberattacks in the collected data sample. Additional details include node labels, a list of attacks, and entity and relationship properties, including their calculated metrics (ranks, community memberships, similarity, etc.). Some of the displayed elements, such as the list of attacks, enable visualization of the related attacks in the main graph display pane with a single click (use case 2), whereas hovering over the attack IDs dynamically highlights the attack topology on a global graph (use case 1). Selecting a specific attack ID during global graph preview (use case 1) also depicts its shell session and isolated structure in an auxiliary display area within the details pane. Scrolling on the details pane reveals additional widgets with specific attack information and global attack graph statistics, including the details on entity label and edge type distribution, malware files, filenames, tags, attack origin statistics, top attacker IP addresses, returning attackers, and the like. Scrolling on the details pane or selecting the *MORE LAYOUT OPTIONS* button reveals global graph search controls, clustering and ranking controls, and node and edge label-based filtering options. Lastly, the solution also enables direct export of the global attack graph data in GEXF format and node design assets in SVG or PNG format to third-party graph analysis tools, such as Gephi [101]. This allows for custom network analysis scenarios and offers fine control over graph rendering but does not provide the same level of graph interactivity, statistical charts, and pre-made visualization tools and projections suited for cyber attack analysis.

Besides tool functionality and user interface design considerations, we took heed of node design for cognitively effective graph visualization. We implemented a node design based on the findings of Lallie et al. [14], attaining to Miller's law [61], following the principles of primary and secondary notation [62], making use of color [99] and utilizing a subset of Bertin's visual variables [60]. We leverage Miller's law to define 7 elementary shapes aiding attack entity differentiation

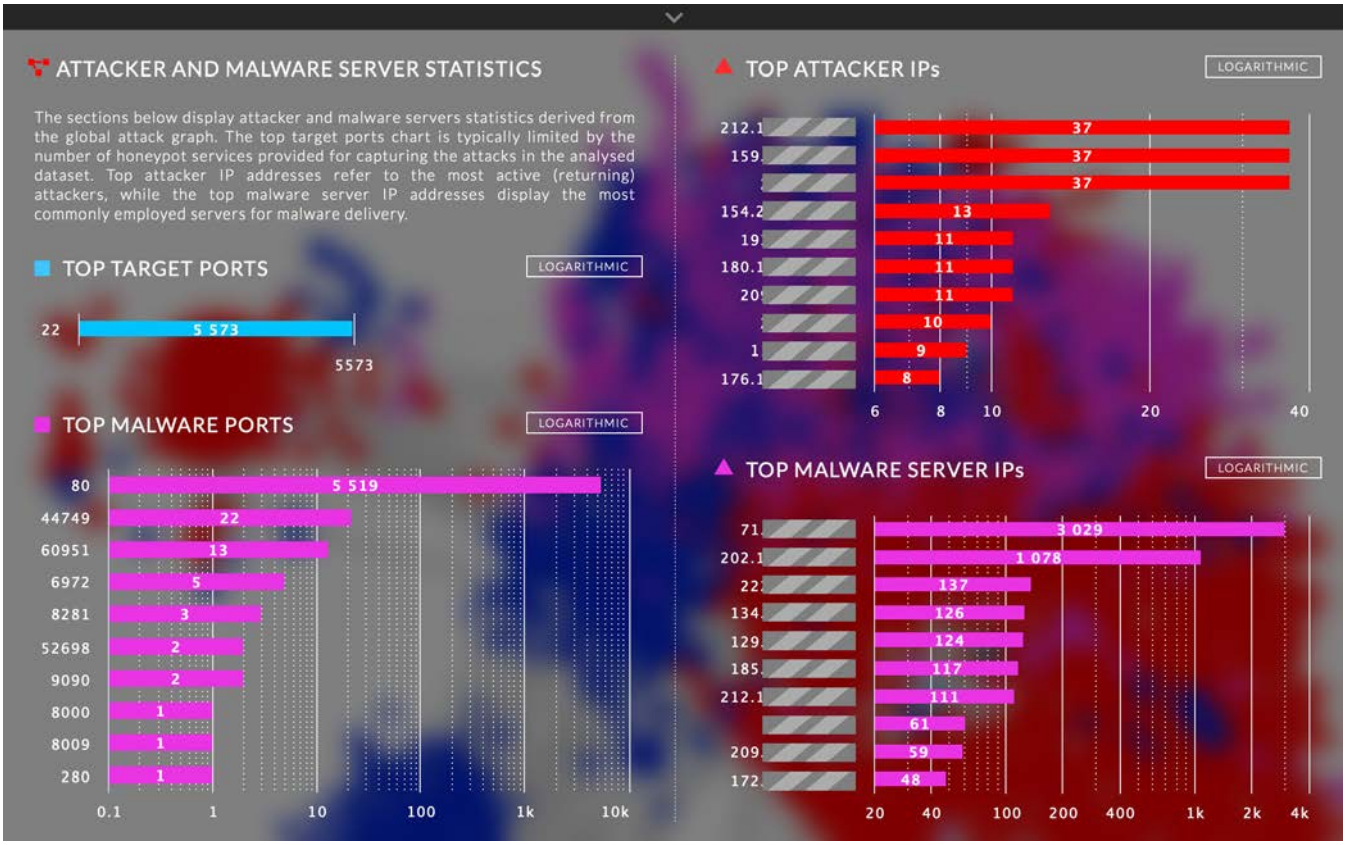


FIGURE 9. The extended entity statistics display focuses on attacker and malware server data derived from the global attack graph. The *TOP TARGET PORTS* section lists the most commonly attacked target service ports (attackers obtaining initial access). In this example, only port 22 is listed, as the loaded data originates solely from SSH honeypots. The remaining charts portray statistical data about the most active attackers and malware distribution servers They depict a relatively small number of returning attackers (*TOP ATTACKER IPs*) compared to two exceptionally common malware distribution servers (*TOP MALWARE SERVER IPs*). Out of the 5573 captured attacks, one of the malware servers was used 3029 times for payload delivery. The most common malware delivery port is HTTP service port 80 (*TOP MALWARE PORTS*). Sensitive data has been redacted from the figure.

on the graph. We further dual-code attack entity types with color to denote their categories and role affiliations. We use red color to indicate the attackers, as it is commonly associated with danger and often used in the cybersecurity domain to indicate the adversaries (e.g., red team). Conversely, we use light blue to denote the target systems analogous to the blue teams. Other most common entities, such as the detected malware servers and the captured malware files, are colored magenta and dark blue, respectively. We attempted to distinguish malware files in other colors to avoid potential misinterpretations with light blue target systems. However, we found dark blue to best maintain the satisfactory visual contrast on both light and dark backgrounds of the implemented view panes while avoiding the potential positive connotation of green and insufficient contrast of yellow on light backgrounds. Composed elementary design primitives, consisting of shapes and colors, therefore, represent data types (e.g., IP addresses, port numbers, files) and roles or categories (e.g., attacker, target, malware) to form graph nodes. The shapes were selected on a subjective basis to be as associative with the represented constructs as possible. While some shapes share a degree of similarity, the entities they represent are often painted in a specific context or in small enough numbers to minimize their ambiguity. We do not consider node shape a limiting factor for label accommodation, as data labels are positioned directly next to the nodes. At the same time, type labels are superimposed only upon mouse hover. Further, we use two types of edge lines to visualize node relationships. In the context of individual attack visualizations correspondent to the Cyber Kill Chain framework, we rely on solid edges to reveal important node relationships and depict attack actions, whereas we use light, dotted edges to display select metadata links. This helps to create visually distinctive attack segments when visualizing an individual attack's structure or analyzing larger amounts of data. In some projections, additional features such as node size and edge width are used to display node importance (e.g., displaying node rank) and relationship weights (e.g., number of connections, node similarity score, etc.). We rely on the developed design system throughout the whole application, as well as the remainder of this document, with the only exception of intentionally using a completely different color palette for a visual representation of clusters. However, for the latter to apply, the cluster coloring option has to be explicitly

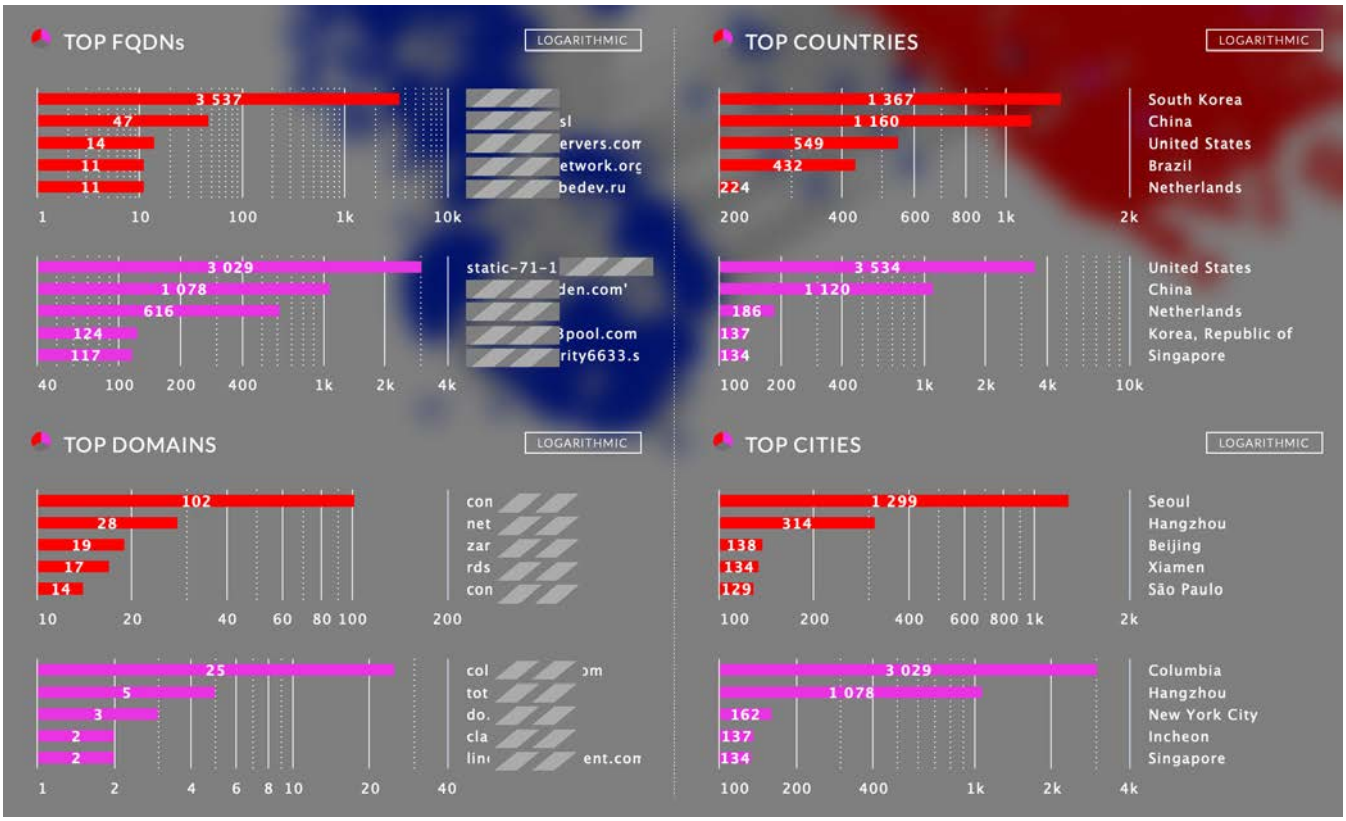


FIGURE 10. Additional attacker and malware server data are displayed upon scrolling the graph details pane. The display lists attacker and malware server domains, fully qualified domain names, and IP geolocation statistics. The most common attacker country is South Korea, and the most common malware server country is the United States. Sensitive data has been redacted from the figure.

enabled in the additional layout options pane. In such cases, cluster colors are also visualized in cluster legend alongside their absolute sizes. Lastly, we also enable the export of node designs as static assets when exporting graph data to third-party visualization solutions. This allows us to maintain semiotic consistency in all attack graph projections and statistic charts across the complete solution. The described node design primitives and a subset of composed nodes are visualized in Fig. 6.

The visualization tool was implemented as a web application. We used standard web technologies, such as JavaScript, HyperText Markup Language (HTML), and Cascading Style Sheets (CSS), to build a responsive user interface and visualization controls. Specific attack graph visualizations are based on Cytoscape.js [102], a graph theory library for visualization and analysis, offering great customisability. Global attack graph and their projections are visualized using Graphology [103] and Sigma.js [104], a JavaScript library aimed at visualizing graphs of thousands of nodes and edges. Large graphs are visualized using WebGL renderers to aid visualization performance by leveraging the computer's Graphics Processing Unit. Individual node designs are implemented as GLSL shader code, while continuous layout algorithms are run in separate threads using Web Workers to avoid reducing the browser's and, therefore, the solution's GUI performance (main thread). Graph data are acquired from node and edge descriptions using asynchronous web requests to an intermediary presentation API. The latter is developed in Node.js and functions as a concealment, transformation, and caching layer, preventing direct access and heavy load on the graph database from the web. The overall web application interface is shown in Fig. 5, whereas the designed attack graph visualizations are displayed in the Validation section.

V. VALIDATION

We used the developed visualization tool in an attempt to validate the design of our cyberattack model. To do so, we demonstrate the use of our solution from the user's perspective by following the outlined use case scenarios to review the captured attack data, navigate and display attack sessions, and reveal attacker communities and their attack characteristics.

Due to the sensitive nature of the collected data, our solution is presently deployed in an on-premise private cloud environment for internal testing and evaluation. The implemented continuous data import mechanism resembles the sliding window algorithm. Captured honeypot attack data are aggregated for a seven-day period and automatically imported into the graph database on a daily basis. The import procedure utilizes the developed data ingestion,

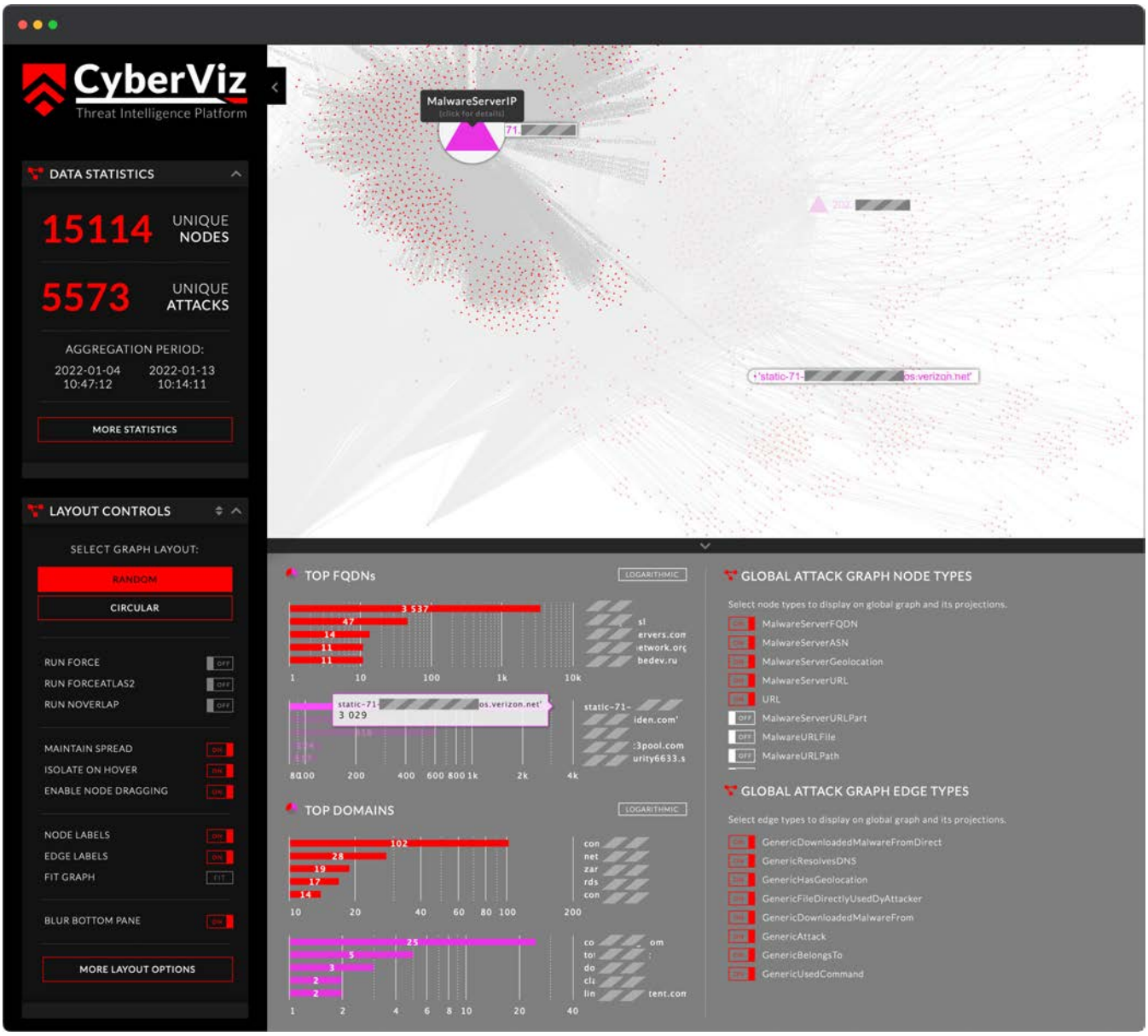


FIGURE 11. Interactive analysis of the most commonly employed malware server on the diluted global graph. The global attack graph is filtered by selecting the desired node and edge types to display. In combination with the enabled node spread, node dragging, and node isolation, this allows for the visual identification of attackers connecting to a particular malware server. The node isolation functionality reveals the malware server's FQDN metadata link and highlights all attackers referencing this server's IP or FQDN in their attacks. Sensitive data has been redacted from the figure.

transformation, and enrichment API to transform raw cyberattack data artifacts collected on CyberLab honeypots into the proposed CyberViz model. The demonstrational attack data analysis conducted below was performed on a data sample of 5573 real-world SSH attacks with payload delivery phases captured between January 4, 2022, and January 13, 2022. The attack analysis and visualizations were conducted using the developed CyberViz tool.

A. GLOBAL ATTACK GRAPH DATA EXPLORATION

Following our first use case, the user utilizes the developed visualization tool to preview the global attack graph and perform an initial exploratory analysis of the collected attack data. To do so, the user selects the *GLOBAL ATTACK GRAPH* option from the data controls section on the left pane of the interface. The global attack graph, consisting of all captured attack entities and relations, is then drawn in the main graph display pane. The user also proceeds to toggle the *RUN FORCEATLAS2* switch to run an arbitrary number of iterations of the selected layout algorithm and better position entities on the graph rather than allocating them random canvas coordinates. The applied force algorithm then forms node clusters based on relationship weights signifying the attack action frequency, metadata relationship count, and the attacker command similarity.

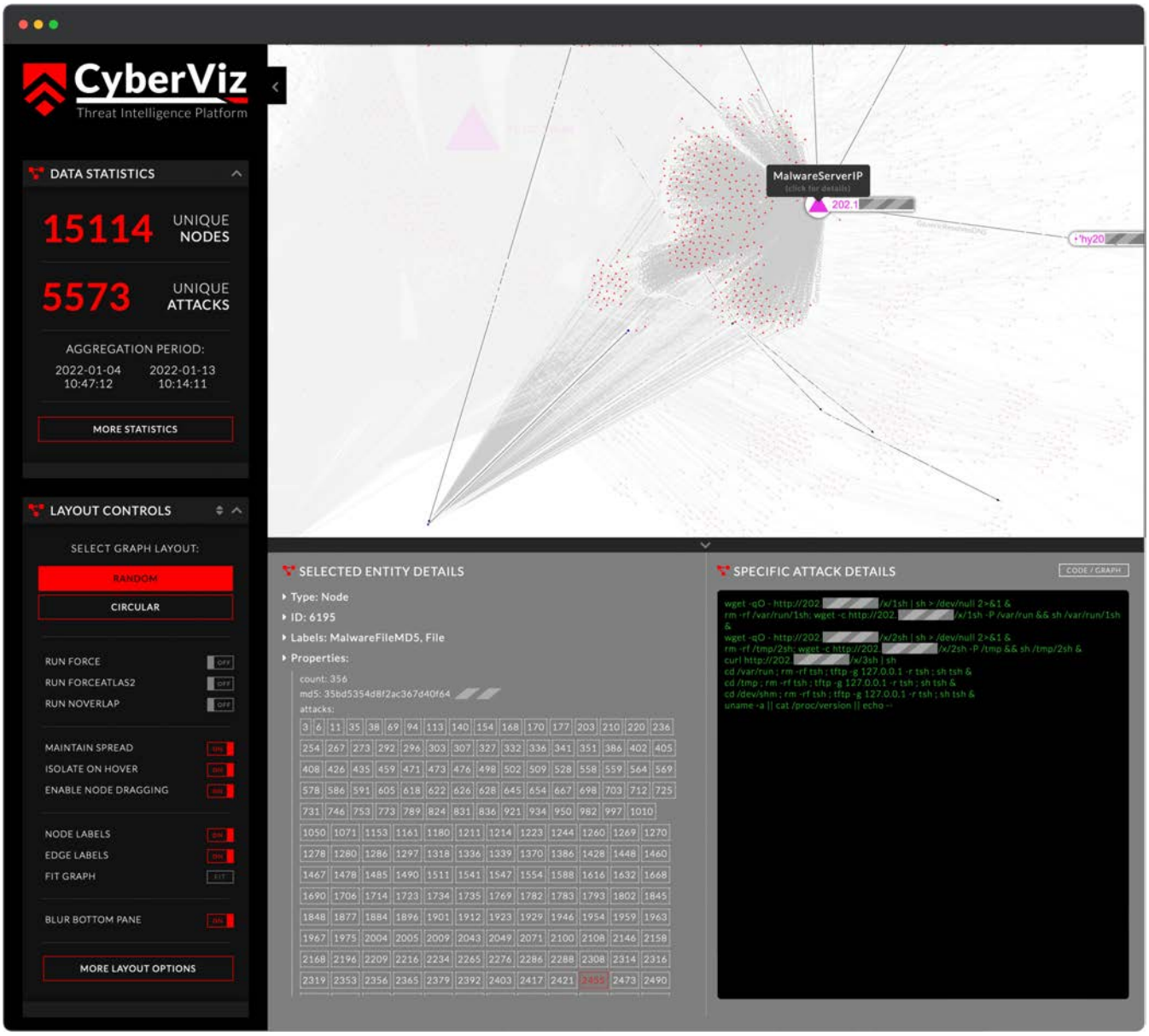


FIGURE 12. Interactive analysis of the second most commonly used malware distribution server. The user hovers over the second top malware server's IP node to visualize its direct links. They then click on the node to freeze the isolated view and explore the highlighted nodes to identify attack characteristics. They then select a particular attack ID to visualize its shell session and display the properties of the deployed malware. Sensitive data has been redacted from the figure.

The global attack graph is accompanied by basic information about the sample under analysis (displayed in the data statistics section of the left pane), including the data aggregation timeframe, the number of unique graph nodes, and the number of unique captured attacks. Since the user intends to acquire more data on the collected sample, they select the *MORE STATISTICS* button to reveal global graph statistics, as well as entity and edge type distributions. These details are displayed in an overlayed pane on the bottom half of the interface. The described user actions form an initial state shown in Fig. 7 and serve as a starting point for attack data analysis.

Without the applied *MAINTAIN SPREAD* and *ISOLATE ON HOVER* options and without any applied entity or relationship filtering, the global attack graph remains severely overplotted for the time being. Therefore, the user rather initially shifts their attention to the entity type distribution chart plotted in the details pane of the interface, as it may help the user to quickly deduce initial observations regarding the sampled attacks. To allow for better readability, we have extracted the chart in question from its web-based form into a suitable paper format in Fig. 8. Nevertheless, the chart's web implementation allows the user to hover over the individual stacked rows and reveal category labels,

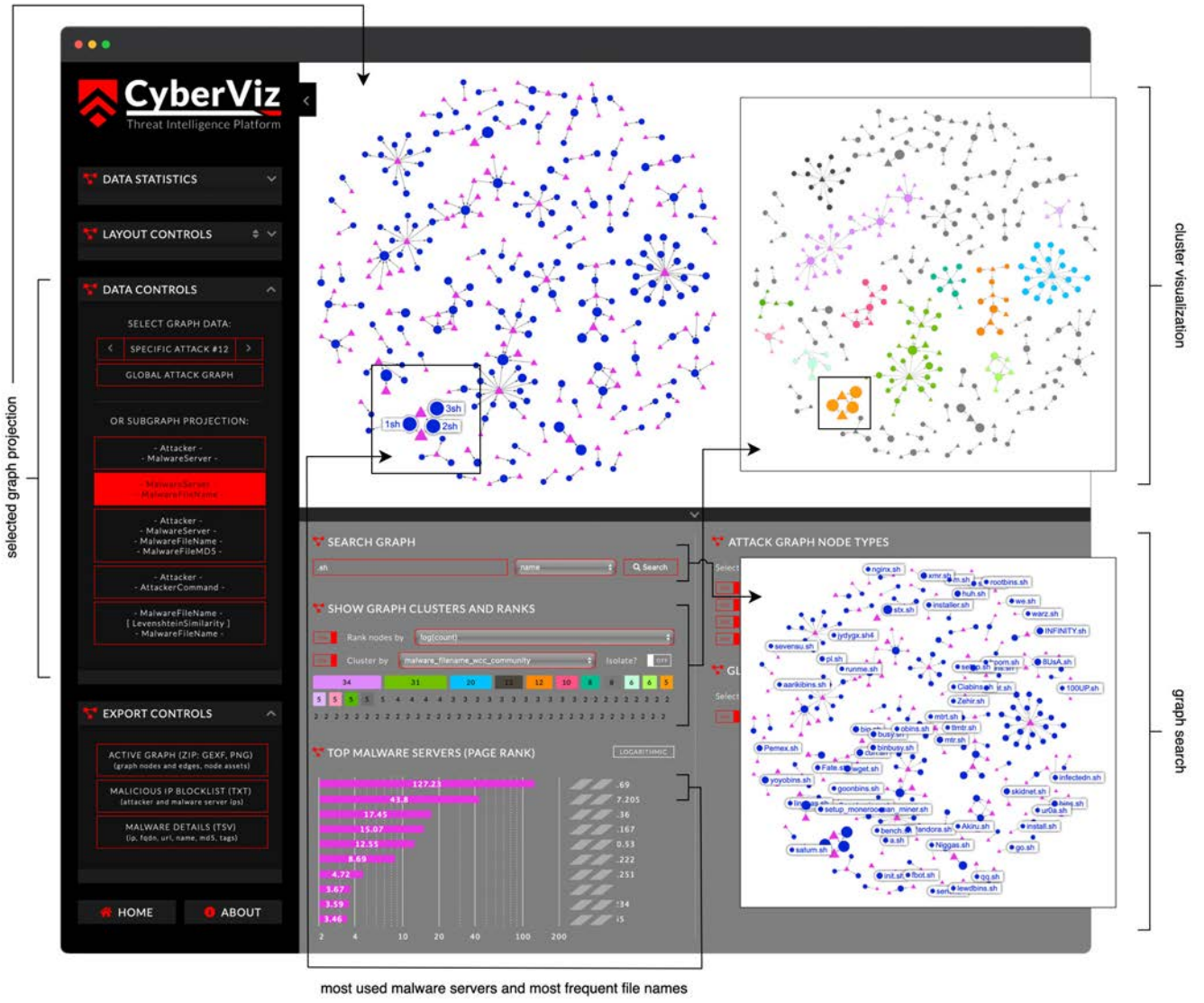


FIGURE 13. Visualization of malware communities based on direct virtual relations between malware server IP addresses and malware file names. Malware servers are represented with magenta-colored triangles, whereas malware file names are depicted as blue dots. Node sizes represent their PageRanks for the given network, whereas edge width is kept constant and does not depend on the event count. The top subfigure visualizes malware communities by coloring the 15 largest connected graph components. The bottom subfigure visualizes graph search functionality by outlining the detected malware files with shell script extensions. Sensitive data has been redacted from the figure.

sizes, and details of entities constituting the global attack graph.

Fig. 8 depicts the entity type distribution in the global attack graph. It shows the dataset includes precisely 5573 captured attacks represented by 15114 unique nodes. However, the user might immediately notice a prominent skew of node counts representing the IP address data. The chart conveys there are 2338 nodes representing the unique IP addresses of the attackers (specific label AttackerIP, colored in red, represented with triangular shape in the graph) and only 197 nodes representing malware server IP addresses (specific label MalwareServerIP, colored in magenta, also represented using triangles in the graph). Moreover, hovering over the stacked rows in the chart reveals that the captured dataset only includes 448 unique malware file hashes and 264 unique malware file names, while there are 5573 recorded payload download actions. The detected skew is also similar for other generic node classes (FQDN, ASN, geolocation) and is indicative of significant reuse of malware distribution servers and their malicious payloads. In other words, the data sample in question likely includes a large number of very similar or repeated attacks conducted by many attacker IP addresses relying on a small number of reused malware file servers.

To further confirm this hypothesis, the user scrolls on the bottom details pane to reveal additional displays portraying the extended entity statistics. The latter include the top attacker and malware server IP address, top malware ports, most frequent attacker and malware IP geolocations, as well

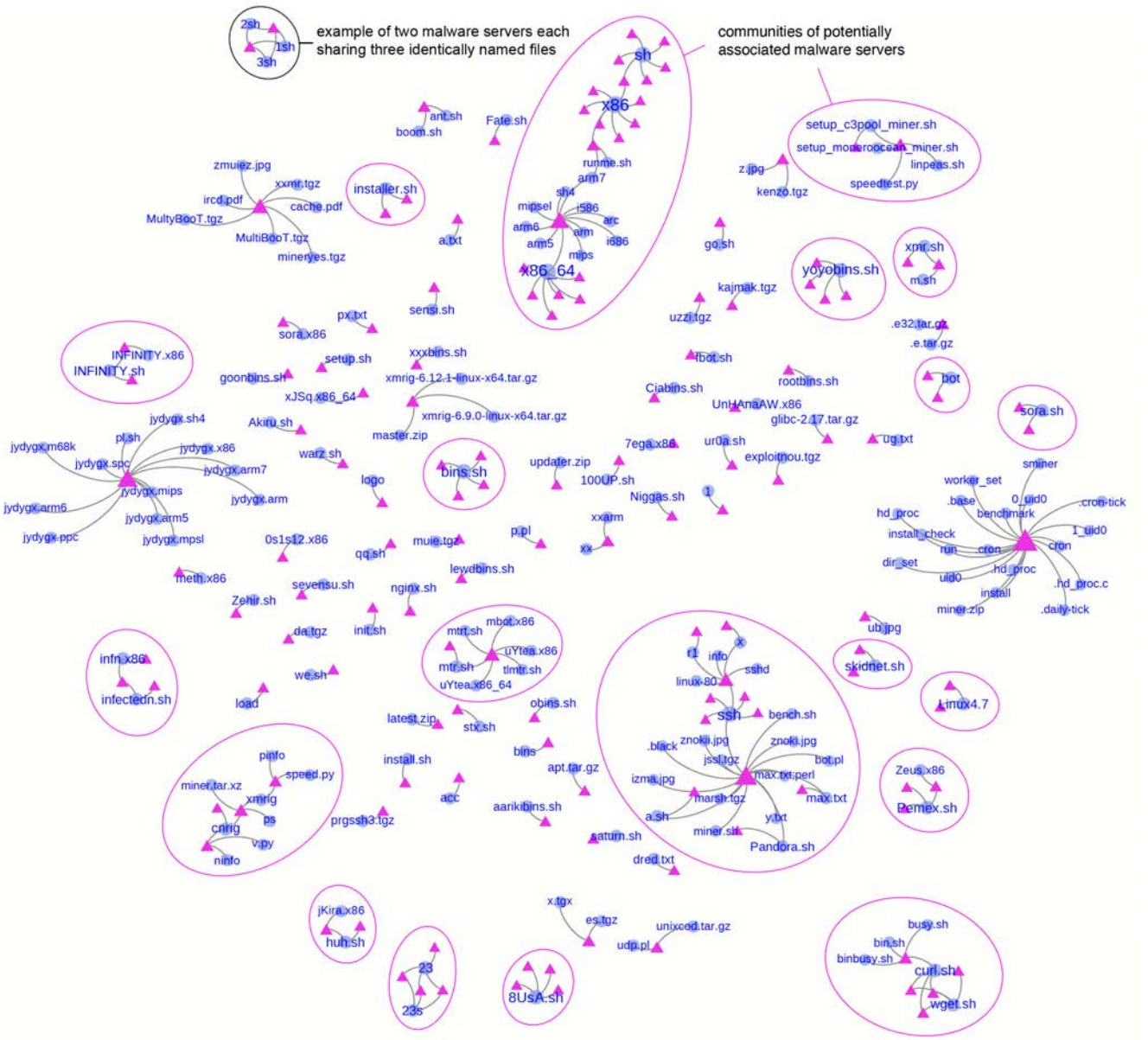


FIGURE 14. Malware distribution communities with displayed file names. Malware servers are represented with magenta-colored triangles, whereas malware file names are depicted as blue dots. Node sizes represent their PageRank for the given network. Edge width is constant and does not depend on the event count. Note the connected communities, where multiple distinct hosts serve identical file names.

as their reverse IP resolutions disclosing their domains or fully qualified domain names. These data are portrayed in Fig. 9 and Fig. 10.

Fig. 9 and Fig. 10 convey additional statistical data about the most active attackers and the most used malware server IP addresses and ports, as well as their geographical origin. As evident in Fig. 9, the most commonly employed port for malware delivery is HTTP service port 80, while the only available target port is port 22, as the loaded data sample only contains SSH honeypot attacks. Moreover, Fig. 9 reveals that two of the most active malware distribution servers constitute nearly 74% of all captured attack traffic and certainly spark further research. Therefore, the user's next prediction might be that these servers are, in fact, well-known public repositories hosting penetration testing tools or malicious code. However, since statistical results only offer a brief overview of the captured attack data and do not capture the complete dynamics of the recorded attacks or portray realistic malware reuse patterns, the user decides to employ interactive graph analysis to uncover complex entity relations.

In order to reduce the global graph complexity, the user navigates to the layout controls section of the left pane and selects *MORE LAYOUT OPTIONS*. The graph details pane is then replaced with additional controls enabling graph search, clustering, and node and edge type filtering. The user disables node display for entity types they consider unnecessary and

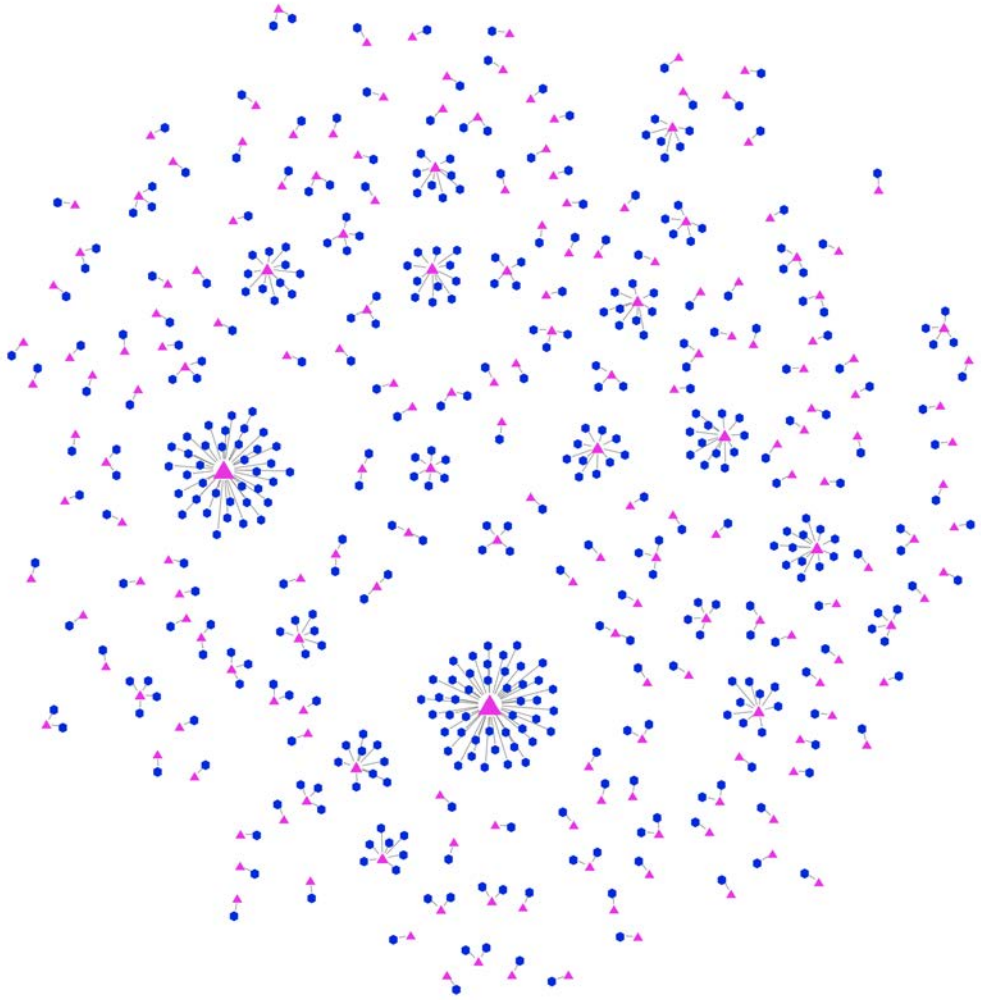


FIGURE 15. Visualization of malware distribution communities based on file hashes. Malware servers are represented with magenta-colored triangles, whereas blue hexagons depict unique malware file hashes. Note the complete absence of connections between communities.

uses layout controls to enable *NODE LABELS* and *EDGE LABELS*. They also enable the *MAINTAIN SPREAD* toggle, as well as *ISOLATE ON HOVER* and *ENABLE NODE DRAGGING* options to aid graph navigation. Lastly, the user runs a desired layout algorithm to reduce node overlap. The diluted graph then enables them to easily visualize the highest-ranking malware servers and interactively display their direct links by hovering over their nodes, as shown in Fig. 11. Moreover, the user may click on the individual node to display its properties, visualize attack terminal sessions, and display attack patterns by hovering over the node attacks table (Fig. 12).

The simple graph analysis conducted above disproves the user's hypothesis of attackers relying on well-known public repositories for malware distribution. While the attack shell sessions reveal short and repetitive malware deployment scripts, the malware servers in the graph still appear directly unconnected to each other. They do not link to the same metadata nodes, and their properties do not reveal DNS resolutions to any of the well-known public service domains. In fact, the two most used malware servers are situated in Verizon Business and China Telecom networks, with FQDN records indicating their intended use by a business or residential customers. The user may therefore anticipate many scenarios, including that these servers could have been compromised and repurposed as malware serving proxies by remote attackers.

B. MALWARE DISTRIBUTION NETWORK VISUALIZATION

While the interactive data exploration enables attack session visualization, discloses entity details, and allows metadata link analysis in the global context, such graph visualizations still suffer from overplotting and require the user to manually filter and navigate the graph to uncover interesting attack characteristics. Since the user is likely interested in visualizing relationships (attacker actions or attack steps

FIGURE 16. Malware file statistics. File name statistics reveal the most frequent malware file names used in the attacks and their most common name collisions. They show that several distinct files (files with different MD5 hashes) were identically named. Malware category statistics reveal more than 15 classes, as reported by malware detection engines. Multiple attackers were downloading identical payloads during the attacks. Three of the most used malware files were used by a very large number of attackers, more than twice as many as the next three most commonly used files. Most of the files originated from malware servers in the United States and the Netherlands. Malware transfer mime-type statistics report that the most common file types include plaintext and executable shell script files. The color scheme in the charts differs from the general color scheme used for node design to better differentiate chart data. The visualizations were extracted from the web solution to better fit the paper format.

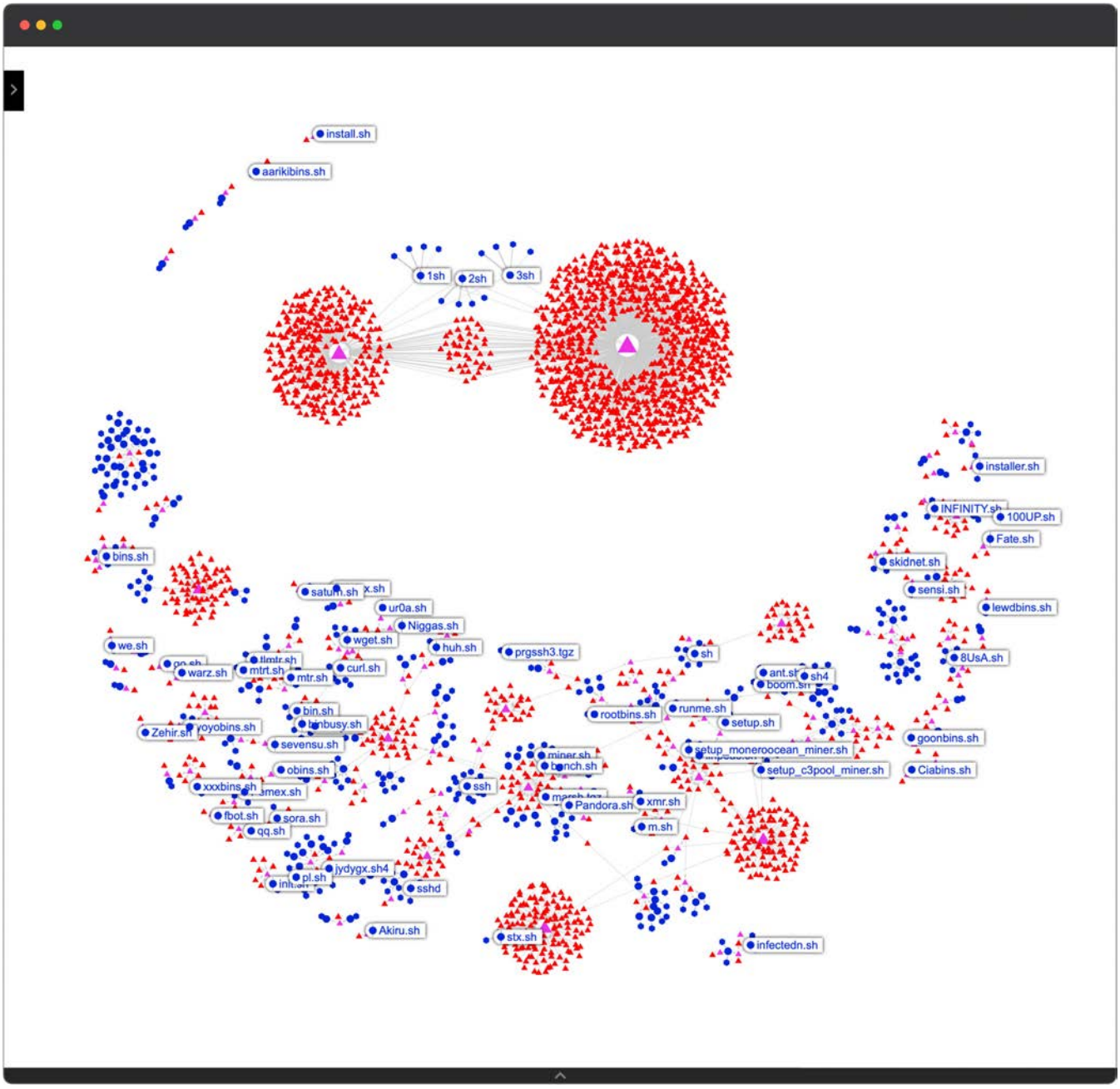


FIGURE 17. Visualization of attacker clusters linked by malware servers and served files and their filenames. Individual attacker nodes are directly connected to malware servers used for conducting cyberattacks. Three of the largest attacker communities are seen transferring three distinct payloads, each with three different names. Visual clusters are created using a force-directed graph layout. Triangle shapes depict unique IP nodes, hexagons depict unique MD5 hashes of transferred files, and circles depict file names. Attacker nodes are colored red, malware servers are colored magenta, and malware files and their names are colored blue. Node sizes represent their PageRank. Highlighted file names indicate shell scripts.

and data links) between the crucial attack entities, such as the AttackerIP, MalwareServerIP, MalwareFileName, MalwareFileMD5, and AttackerCommand, we used graph data science algorithms and dimensionality reduction techniques to further dilute the data around those labels.

We prepared multiple graph feature projections from the global attack graph, encompassing the select relationships between the nodes. One of such projections visualizes malware distribution networks. We reduced data dimensionality using a new virtual relationship, directly connecting unique malware server IP addresses with the unique file names of the downloaded payloads without intermediary nodes or decomposed attacker actions. Such a high-level virtual relationship roughly corresponds to the payload delivery step of the Cyber Kill Chain. The user can display the desired graph projection by selecting it in the data controls

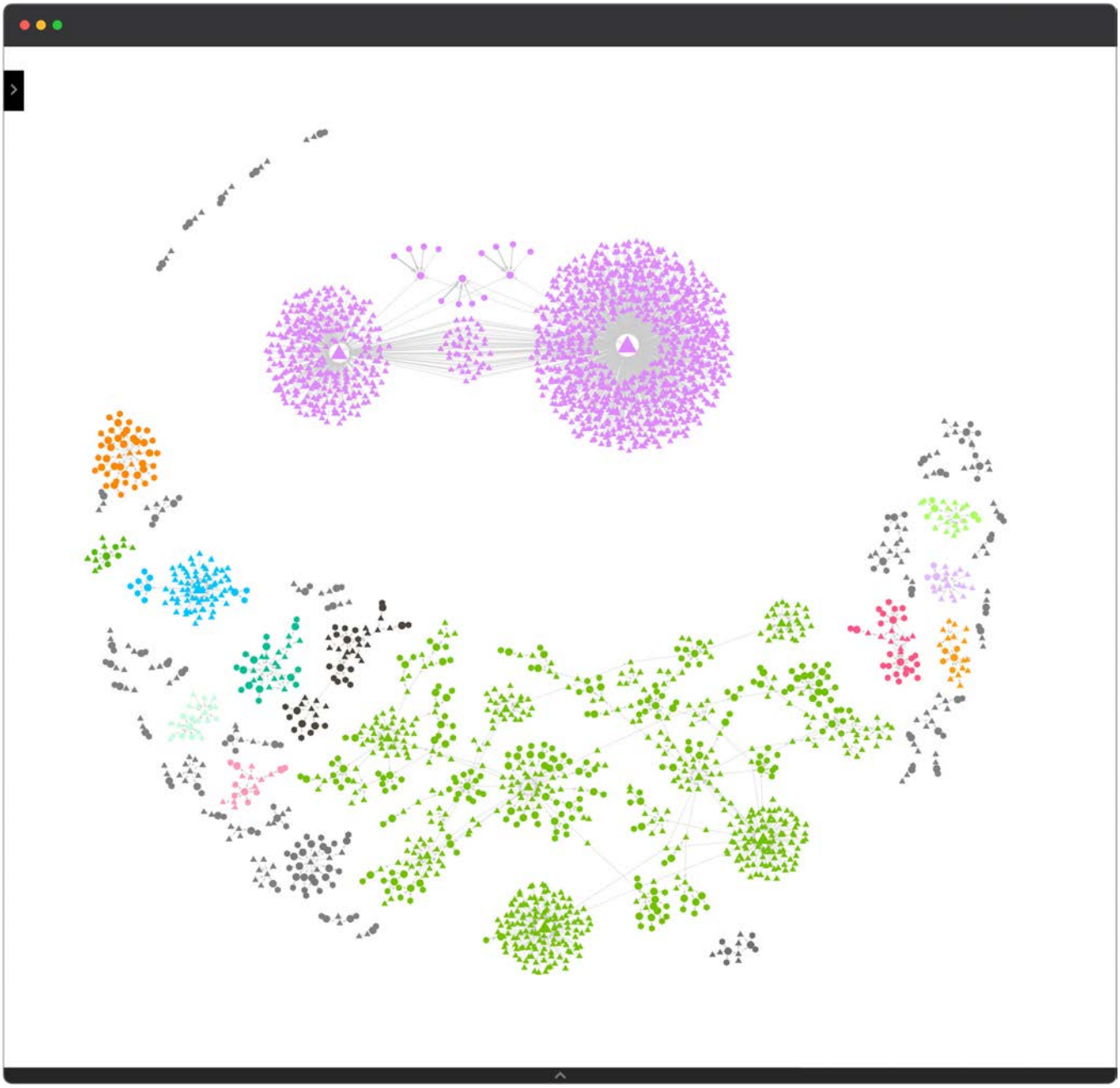


FIGURE 18. Botnet detection based on connected component search in an attack graph projection. Color-coded node communities represent detected botnets (connected networks of attackers). Individual botnets may consist of several smaller communities, indicating attackers' preferential malware distribution server. The largest 15 communities are color-coded in an alternative color scheme corresponding to the bar colors in community size charts. Node sizes represent PageRank, and edge widths represent connection count.

section of the left pane. The resulting malware distribution communities are then displayed in the main graf preview area, as shown in Fig. 13. The user may utilize additional visualization controls to search, cluster, and rank the nodes by arbitrary properties. In Fig. 13, the user first ranks the nodes by their log normalized download counts (due to the high variance of the number of download events) and then colors the clusters based on their assigned community identifier (obtained using the Weakly Connected Components search algorithm). Moreover, the user may scroll the details pane to reveal additional statistical data on captured malware files or match a file name pattern.

Note: All of the available graph projections, node and relationship rank, and community identifiers, as well as statistical data on a global graph and malware files, are automatically calculated during each data import event.

The identified malware distribution communities reveal potential hidden links between malware servers

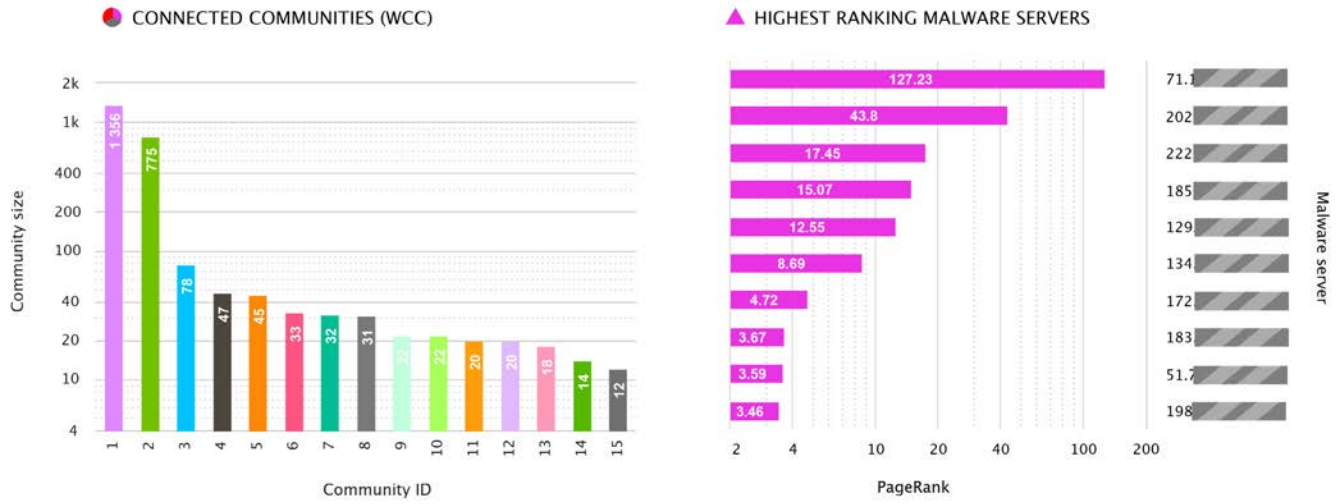


FIGURE 19. Exported charts displaying suspected botnets (detected attacker communities) and their bot count, as well as malware servers' PageRank. Sizes of detected communities are color-coded following the color scheme used for visualization in Fig. 18.

(purple triangles) based on the assigned malware file names (blue circles) in the analyzed attack sample. To better visualize file name patterns served by individual distribution servers, the user may decide to export the active graph from the web solution. To do so, they use the *ACTIVE GRAPH* button in the *EXPORT CONTROLS* section of the control pane. The exported data are contained within a ZIP file and includes a GEXF graph description file and image assets for all exported entities. The user may open the graph file in their graph visualization suite of choice and apply custom rendering settings to generate an output such as the visualization of all served malware file names in Fig. 14.

In Fig. 14, the user also applied a weighted PageRank algorithm to the given network to measure the importance of individual nodes in relation to the number and weight (event count) of their edges. Rescaled node PageRank is then presented as node size in graph visualization.

However, since each unique file name is only stored in the global attack graph once, Fig. 13 and Fig. 14 may not accurately depict malware distribution communities in cases where two different payloads coincidentally share the same name but vary in attackers or contents. Since several malware files seem to be semantically named, there is a relatively high potential for name collisions. Therefore, the user employs a different graph projection based on a new virtual relationship between the malware server nodes and the MD5 hashes of the files they host. The user once again prepares the graph using the web interface and exports it in a suitable format to visualize the malware distribution communities more precisely.

While Fig. 14 displays several connected communities, indicating that multiple unique malware servers host files with identical names, Fig. 15 illustrates that no two malware servers actually hosted identical files. This might be due to the common practice of malware obfuscation, file binding, or polymorphic encoding of payloads by the attackers. While the latter visualization is more appropriate for quantitative malware analysis, the former better illustrates potential associations between the malware servers. The likelihood of true positive associations increases with the increase in the number of files and the uniqueness of their names.

C. MALWARE FILE ANALYSIS AND BOTNET DETECTION

After visualizing malware distribution networks, the user might be interested in the details of the captured malware files. For this reason, our solution statically analyzes and classifies all captured malware files using multiple malware detection engines. Shell scripts, text files, and URLs are decomposed into individual shell commands or strings. All files are attributed with metadata, such as file hash, file name, and mime type. File names are correlated and grouped by their edit distances using the Levenshtein similarity algorithm to discover similar nomenclature patterns and potential human interaction (typographical errors). PageRank and Louvain community detection algorithm are used on subgraphs of MalwareServerIP and MalwareFileName nodes linked by ServedFileNameDirect edge. This allows for additional malware categorization based on file name patterns and reveals their cluster's affinity to particular malware distribution servers.

Fig. 16 lists malware file statistics. The most common transferred payload types include plain text files and executable shell scripts. Three of the most common file names in the captured attack sample are '1sh', '2sh', and '3sh'. They were used in a comparable number of attacks, and their file names constitute the majority (74%) of all transferred payloads, thus confirming the user's initial hypothesis of a high number of similar attack repetitions. All files with these three file names were distributed by two servers, as outlined in the upper left corner of Fig. 14.

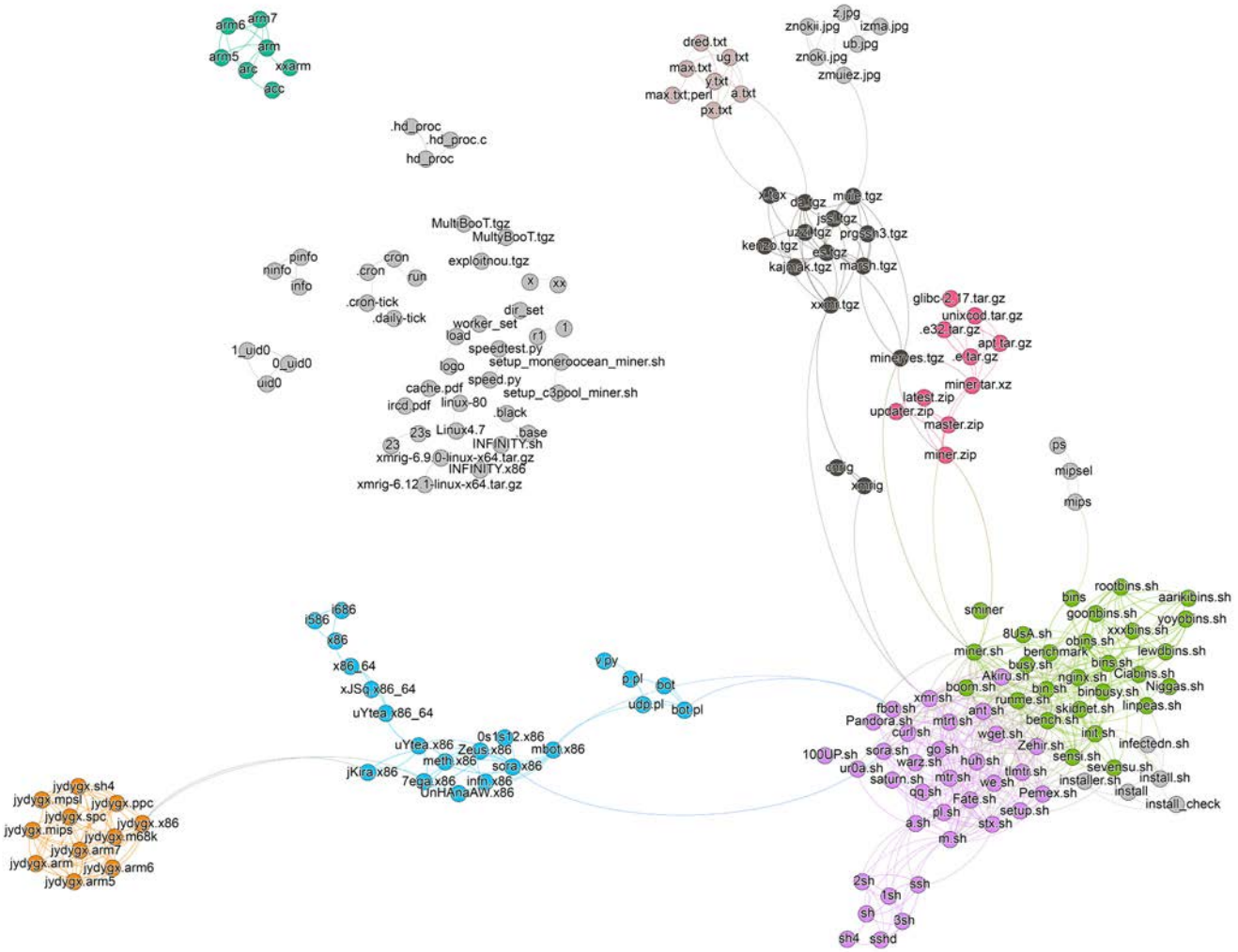


FIGURE 20. Exported network of malware file names used for malware categorization based on file name similarity. The projection visualizes MalwareFileName nodes from the global attack graph and links them using a LevenshteinSimilarity virtual relationship. A force-directed layout is used to visually group nodes with higher name similarity, and network modularity is used to color individual modules. The color scheme differs from the general color scheme used for node design.

However, since the hashes of all served files differ, no inter-community links appear in Fig. 15. To further investigate this discrepancy, the user makes use of another graph projection encompassing all four of the crucial participating nodes: AttackerIP, MalwareServerIP, MalwareFileMD5, and MalwareFileName. Analogous to the malware distribution network visualization, this projection forms a new virtual relationship to directly connect unique AttackerIP and MalwareServerIP nodes. It calculates PageRank for the nodes in the new network and considers the malware download count as a parameter of the virtual relationship. The nodes and edges are then scaled according to the new weights. They reveal multiple attacker groups focused on malware distribution servers and their served file names and payloads (Fig. 17).

Fig. 17 reveals that many attackers in the largest cluster are transferring multiple distinct malware payload variations. These payloads are concealed under three file names (‘1sh’, ‘2sh’, and ‘3sh’), each of which is hosted by both malware servers with the highest rank in the global attack graph. Some of the attackers from the cluster only transferred files from one of the malware servers, whereas others connected to both and revealed a potential hidden link between the servers. The number of different attacker IP addresses using each unique malware file is depicted on the bottom left chart in Fig. 16.

The graph further reveals that each of these three file names, in fact, conceals four different payloads, none of which was ever served by both of the servers. The user might again speculate that this is likely due to the payload obfuscation in an attempt to evade detection. Therefore, while the links between the two servers are revealed by individual attackers, the potential links between the attacker communities are revealed by file name patterns. By uncovering these links in our attack graph model, we managed to unveil the relationships between the payloads, thus increasing the chances of their detection.

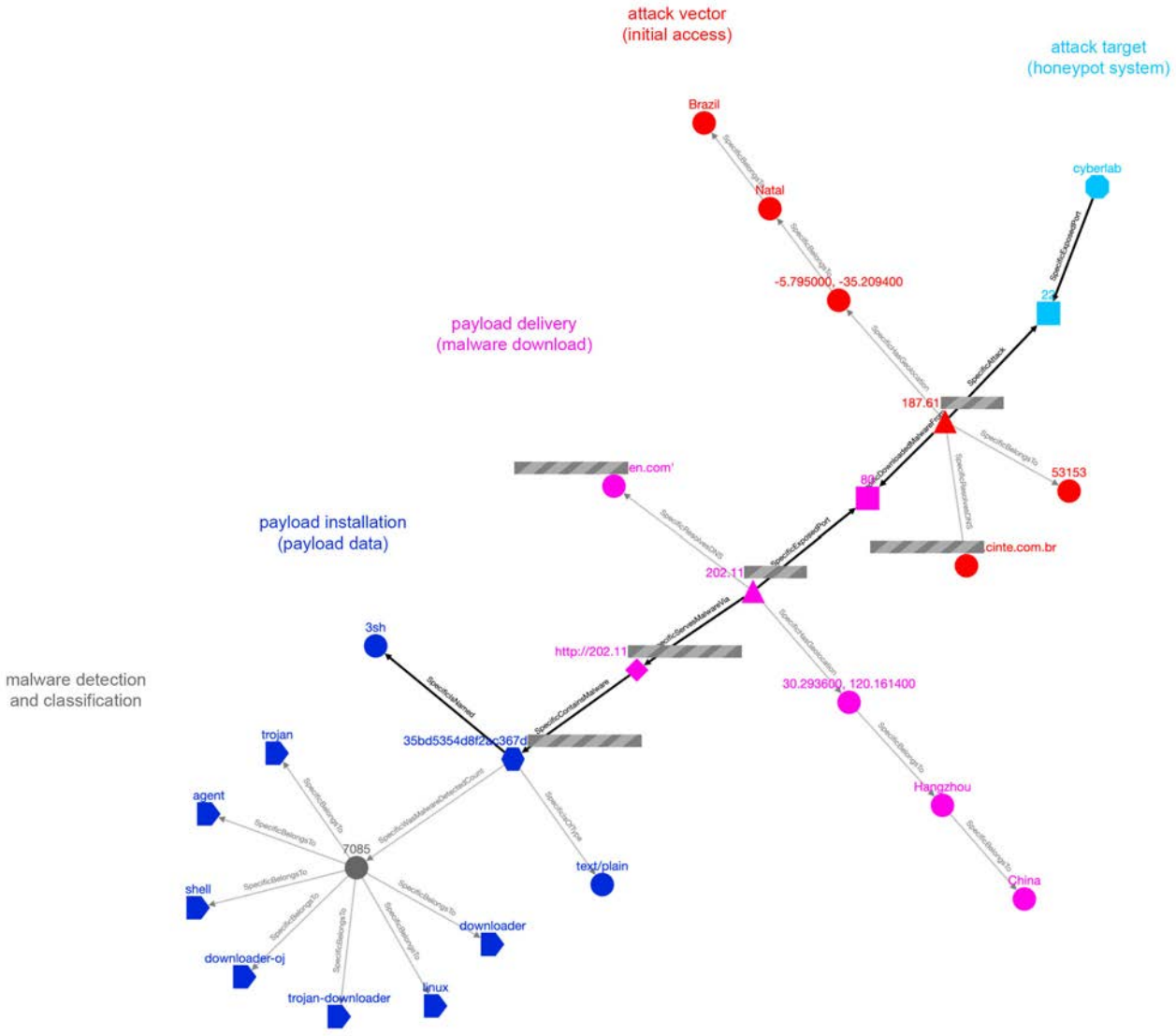


FIGURE 21. Simplified visualization of an individual attack event. The simplified attack graph visualizes multiple stages of the attack, its participating entities, and their metadata. Metadata information is dispersed between node properties (displayed on click) and metadata nodes. Nodes participating in the attack are connected using emphasized edges and form the main attack execution chain. Metadata nodes are connected using light, dotted edges and are positioned perpendicular to the attack chain. They represent crucial node data and serve as graph pivot points for exploring linked entities from other recorded attacks. Sensitive node information has been redacted from the figure and the user interface has been hidden.

Such graph projections also form a basis for botnet detection using multiple graph features (e.g., select links from the global graph and newly formed virtual relationships). Individual attackers may be associated based on their reliance on distinct malware distribution servers (networks) for payload delivery based on malware naming patterns, file hashes, command similarity, execution order, or a combination of these and other factors. Fig. 18 illustrates a rudimentary approach toward botnet detection based on community detection algorithms for the selected projection. In Fig. 18, the user visualizes the results by color-coding the most significant connected components of the network.

Fig. 18 illustrates the potential botnets. The connected component search algorithm applied to the new network results in 57 communities, the largest 15 of which are color-coded. Detection quality scales with the size of the captured attack sample and is amplified by the distributed nature of the underlying honeypot system. While the proposed approach is capable of botnet detection, misclassifications are likely to occur if the data are not complemented by a multitude of other dimensions. Either individual botnets may consist of several smaller groups of attackers or the individual attackers may belong to multiple botnets at the same time.

To automate attacker classification and assort attackers into communities, connected component search [105]

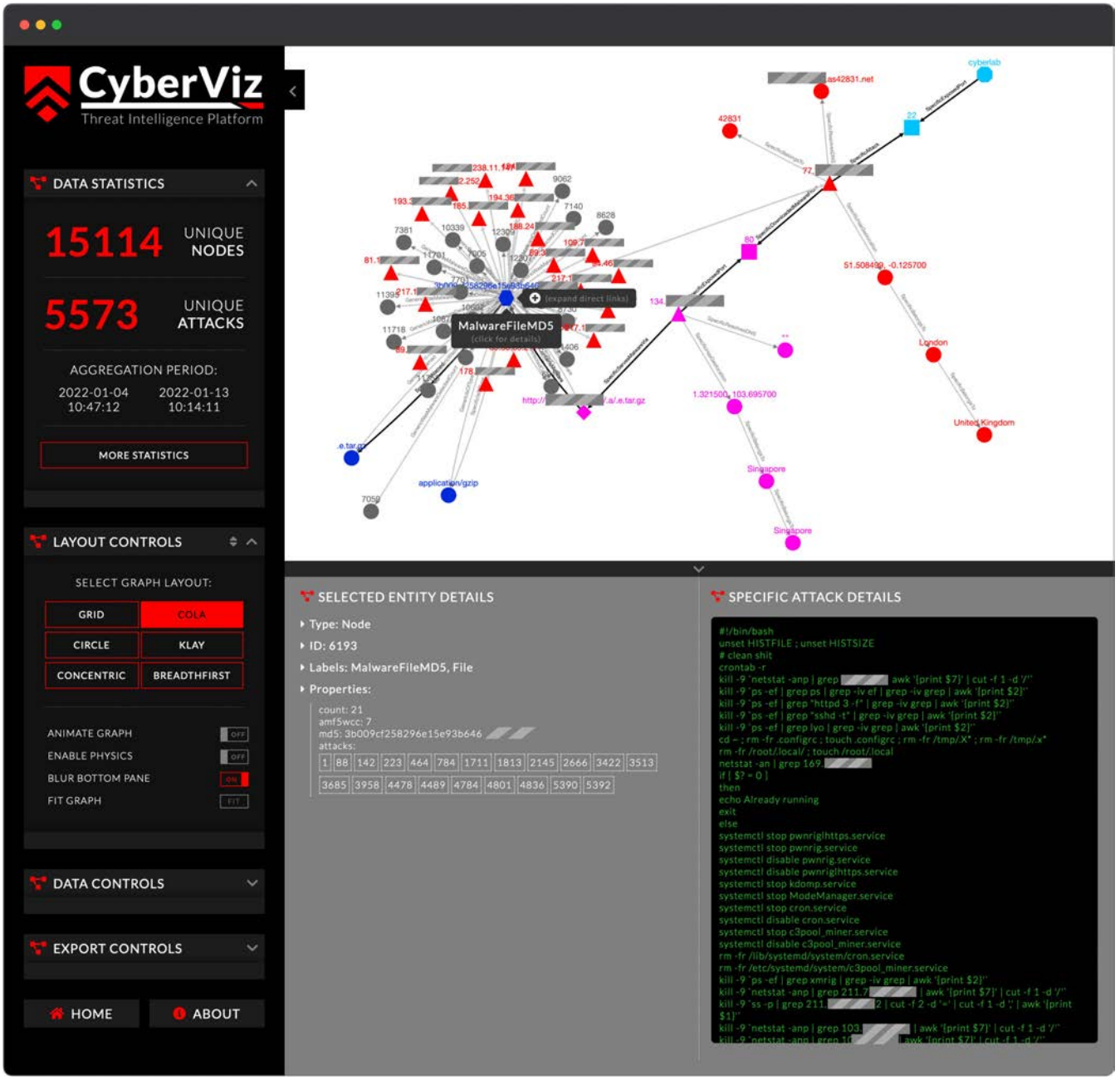


FIGURE 22. Attack exploration using node-link expansion and entity details pane. The graph display pane shows entities related to the selected malware file name '3sh'. These include at least three additional unique files (shown as hashes) sharing the exact same file name. These files were downloaded to various compromised hosts during the other attacks and are linked to the unique file name node in the global attack graph. Besides file-to-name relations, links to other involved malware servers and similarly named files are also shown.

and community detection algorithms, such as the Louvain method [106], are automatically performed for all predefined graph projections during the data import procedure. Detected botnets are ordered by size and charted in the details pane of the interface. Extracted community sizes and malware server PageRank charts for graph projection in Fig. 18 are shown in Fig. 19.

The user may decide to use the graph data to categorize malware into virtual groups based on their file name similarity. They can do so by visualizing the Levenshtein distance between the individual malware file names. The user first applies graph filters to only visualize MalwareFileName nodes and models a new direct relationship with the calculated LevenshteinSimilarity parameter. Lastly, the user may measure the network modularity of the new graph and color code the individual classes. The resulting exported visualization is shown in Fig. 20.

D. INDIVIDUAL ATTACK BROWSER

Our second use case scenario includes the review of the individual captured attacks in isolation from the global attack graph. Instead of applying node and edge label filters on the global graph, the user can employ *DATA CONTROLS* to enter the *SPECIFIC ATTACK* view and only retrieve nodes bound by specific relationships tied to the attack's unique identifier. In this manner, generic graph relationships are not retrieved, and the user can focus on the extracted subgraph (attack) topology.

The user can review individual attacks by directly referencing their identifiers either using manual input, by selecting attack links displayed in various entity properties, or by time-consecutively browsing between the captured attack events. An example of the simplified attack structure is shown in Fig. 21.

Fig. 21 depicts the targeted honeypot system (cyan) and several consecutive stages of the attack execution chain. The attack event flow is depicted from the top right to the bottom left of the graph. It starts with the initial access to the target system over the SSH service on port 22 (attack vector). The attacker's IP address (red) is traced to the town of Natal, Brazil, based on the geographical IP data. It belongs to the Autonomous System number 53153 and resolves a reverse DNS lookup. The attacker then initiates an outgoing connection from the compromised system and retrieves the malware payload from another server in Hangzhou, China, listening on port 80. The payload delivery stage is modeled as an action originating from the attacker server to the malware distribution server, even though the outgoing connection is, in fact, established from the compromised host. This links the attacker and malware servers when visualizing the attack. During the malware download and installation stage, its distribution URL is recorded and decomposed it into URL parts. The latter are not displayed in the individual attack preview. The downloaded files and their properties, including their MIME type and extracted strings, are also stored in the database. The files are represented by their MD5 hash and file name(s). Captured files are then scanned for known malware with an array of malware detection engines and classified with malware category tags. These are linked to the detection result reference and displayed on the graph.

Specific attack preview also enables the user to browse entity relations outside of the scope of the current attack. It allows them to discover entity links without the requirement to display all nodes of the global attack graph. The analysis is conducted inversely to the exploration of the global attack graph: instead of diluting the graph to identify hidden links, the user expands relationships starting from a specific attack to reach related nodes and reveal prominent features.

Since the user is interested in the details of the attacker's malware, they hover over the malware file node to expand its direct links from the global attack graph. With a few clicks, the user reveals two virtual relationship types, the GenericFileDirectlyUsedByAttacker and GenericWasMalwareDetected, thus unveiling all attackers relying on the same malware file to execute the attack and malware scan results at various points in time. This is illustrated in Fig. 22.

Fig. 22 illustrates the visual attack exploration mode on the main graph display and lists the selected entity and attack session details in the bottom pane. Most notably, entity details reveal the total number of times this file was detected in a global attack graph. They also reveal the corresponding identifiers of these attacks and allow the user to visualize them on the graph with a single click. Besides the attack involvement data, the bottom pane displays a replay of the captured SSH session, revealing an automated malware installation script. The topmost part of the figure shows the direct link expansion resulting from hovering over the selected file node. This option visualizes select directly connected nodes from the global attack graph. These include 20 additional attacker IP addresses utilizing the same malware file and indicating a potential correlation between the attackers.

VI. CONCLUSION, LIMITATIONS, AND FUTURE WORK

We presented a novel approach for cyberattack modeling using graph theory and validated its usefulness with a proof-of-concept implementation and analysis of two common use case scenarios. We described the developed model and visualization tool and demonstrated their benefits during a walk-through of their use-case scenario. We presented the use of the designed software for honeynet attack data visualization and analysis. We used the developed attack graph model to explore attack entity relationships and identify and visualize malware distribution networks. We presented statistical data on captured attacks, conducted malware file analysis, and displayed rudimentary botnet detection functionality. Lastly, we deduced the characteristics of the captured attack sample, revealing substantial malware reuse and uncovering hidden links between attackers, malware distribution servers, and malware files. With that, we demonstrated that the developed proof-of-concept aids cyberattack modeling, analysis, and visualization. We believe the results obtained using our tools and methods are significant as they demonstrate the power of interactive visual analytics designed with end users in mind. They highlight the importance of continuous monitoring of the dynamic threat landscape and provide insights into the challenges of addressing a wide variety of cyberattacks with a unified model.

We aimed to design our solution in accordance with the reviewed visualization best practices in an attempt to ease attack cognition. We conducted expert consultations with the potential users of the tool in the cybersecurity domain and identified their needs. We consulted cybersecurity analysts and incident response teams and collected their continuous feedback throughout the solution development. We received both positive and negative feedback. Cybersecurity professionals complimented our graph-based approach to threat hunting and attack analysis. However, they expressed their

concerns over the global attack graph complexity and the requirement of manual attack data mapping to fit or extend the model. They complimented the tool's interactivity and attack visualizations resembling the Cyber Kill Chain model but questioned its coverage and missed a wider array of integration with third-party intelligence data providers. Furthermore, we also identified several occasions of overplotting during the visualization of vast datasets. We concern overplotting may worsen and reduce the visualization performance and the effectiveness of attack data communication with an increase in data amount.

Within our future work, we plan to address the expressed user concerns and propositions regarding the demonstrated solution. We plan to extend the attack graph model with attack reconnaissance data collected from a network telescope (https://telescope.ltfe.org) and implement additional reconnaissance and post-exploitation data analysis tools. We intend to further improve the interactivity and visualizations offered by our tool by introducing timeline-based dynamic graph previews and implementing additional third-party data integrations with the MITRE ATT&CK framework, MISP platform, and the CVE program. Our major research directions entail data dimensionality reduction and prominent attack feature extraction approaches based on machine learning, including using large language models. We aspire that a holistic approach towards cyber incident analysis will enable better attacker profiling, identification of human actors, and detection of advanced persistent threats.

APPENDIX
DATA AVAILABILITY

The collected honeypot metadata that served as a basis for the processing and visualization described in this paper is publicly available on our project's website, https://cyber.ltfe.org, or available upon request. A limited subset of the data are also available in the Zenodo repository under the DOI:10.5281/zenodo.3687527. All published data are de-identified and pseudonymized to protect any sensitive information. All intermediate results and metadata can be made available at the authors' discretion upon request. Requests can be sent to the corresponding author.

A supplementary video demonstration of the developed proof-of-concept solution is available at https://youtube.com/watch?v=-YzH2zSFwbs.

REFERENCES

[1] V. Nicomette, M. Kaâniche, E. Alata, and M. Herrb, "Set-up and deployment of a high-interaction honeypot: Experiment and lessons learned," *J. Comput. Virol.*, vol. 7, no. 2, pp. 143–157, May 2011. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00762596

[2] E. Krokos, A. Rowden, K. Whitley, and A. Varshney, "Visual analytics for root DNS data," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Oct. 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8709205/

[3] T. Yadav and A. M. Rao, "Technical aspects of cyber kill chain," in *Security in Computing and Communications* (Communications in Computer and Information Science), J. H. Abawajy, S. Mukherjea, S. M. Thampi, and A. Ruiz-Martínez, Eds. Cham, Switzerland: Springer, 2015, pp. 438–452.

[4] P. N. Bahrami, A. Dehghantanha, T. Dargahi, R. M. Parizi, K.-K. R. Choo, and H. H. S. Javadi, "Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures," *J. Inf. Process. Syst.*, vol. 15, no. 4, pp. 865–889, 2019. [Online]. Available: https://www.koreascience.or.kr/article/JAKO201925462478086.page

[5] H. Al-Mohannadi, Q. Mirza, A. Namanya, I. Awan, A. Cullen, and J. Disso, "Cyber-attack modeling analysis techniques: An overview," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud Workshops (FiCloudW)*, Aug. 2016, pp. 69–76.

[6] R. Daszczyszak, D. Ellis, S. Luke, and S. Whitley. *TTP-Based Hunting*. Accessed: Jun. 9, 2022. [Online]. Available: https://www.mitre.org/publications/technical-papers/ttp-based-hunting

[7] P. K. Manadhata and J. M. Wing, "A formal model for a system's attack surface," in *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats* (Advances in Information Security), S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds. New York, NY, USA: Springer, 2011, pp. 1–28, doi: 10.1007/978-1-4614-0977-9_1.

[8] K.-W. Lye and J. M. Wing, "Game strategies in network security," *Int. J. Inf. Secur.*, vol. 4, nos. 1–2, pp. 71–86, Feb. 2005. [Online]. Available: http://link.springer.com/10.1007/s10207-004-0060-x

[9] X. Cai, K. Shi, K. She, S. Zhong, and Y. Tang, "Quantized sampled-data control tactic for T–S fuzzy NCS under stochastic cyber-attacks and its application to truck-trailer system," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7023–7032, Jul. 2022.

[10] A. Bose and K. G. Shin, "Agent-based modeling of malware dynamics in heterogeneous environments," *Secur. Commun. Netw.*, vol. 6, no. 12, pp. 1576–1589, Dec. 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/sec.298

[11] X. Cai, K. Shi, K. She, S. Zhong, Y. C. Soh, and Y. Yu, "Performance error estimation and elastic integral event triggering mechanism design for T–S fuzzy networked control system under DoS attacks," *IEEE Trans. Fuzzy Syst.*, vol. 31, no. 4, pp. 1327–1339, Apr. 2023.

[12] D. Dagon, C. Zou, and W. Lee, "Modeling botnet propagation using time zones," in *Proc. 13th Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2006.

[13] S. Caltagirone, A. Pendergast, and C. Betz, "The diamond model of intrusion analysis," Defense Tech. Inf. Center, US DoD, Tech. Rep., Jul. 2013.

[14] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Comput. Sci. Rev.*, vol. 35, Feb. 2020, Art. no. 100219. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013719300772

[15] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *Proc. Workshop New Secur. Paradigms (NSPW)*, 1998, pp. 71–79. [Online]. Available: http://portal.acm.org/citation.cfm?doid=310889.310919

[16] J. Zeng, S. Wu, Y. Chen, R. Zeng, and C. Wu, "Survey of attack graph analysis methods from the perspective of data and knowledge processing," *Secur. Commun. Netw.*, vol. 2019, Dec. 2019, Art. no. e2031063. [Online]. Available: https://www.hindawi.com/journals/scn/2019/2031063/

[17] I. Chokshi, N. Ghosh, and S. K. Ghosh, "Efficient generation of exploit dependency graph by customized attack modeling technique," in *Proc. 18th Int. Conf. Adv. Comput. Commun. (ADCOM)*, Dec. 2012, pp. 39–45.

[18] I. Kotenko and M. Stepashkin, "Attack graph based evaluation of network security," in *Communications and Multimedia Security* (Lecture Notes in Computer Science), H. Leitold and E. P. Markatos, Eds. Berlin, Germany: Springer, 2006, pp. 216–227.

[19] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: Association for Computing Machinery, Oct. 2006, pp. 336–345, doi: 10.1145/1180405.1180446.

[20] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation," in *Proc. ACM Workshop Vis. Data Mining Comput. Secur. (VizSEC/DMSEC)*. New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 109–118, doi: 10.1145/1029208.1029225.

[21] O. M. Sheyner, "Scenario graphs and attack graphs," Ph.D. dissertation, Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2014.

[22] N. Idika, "Characterizing and aggregating attack graph-based security metric," Ph.D. dissertation, Purdue Univ. West Lafayette, West Lafayette, IN, USA, 2011. [Online]. Available: https://docs.lib.purdue.edu/dissertations/AAI3444572/

[23] D. Tayouri, N. Baum, A. Shabtai, and R. Puzis, "A survey of MulVAL extensions and their attack scenarios coverage," 2022, *arXiv:2208.05750*.
[24] S. Noel and S. Jajodia, "Optimal IDS sensor placement and alert prioritization using attack graphs," *J. Netw. Syst. Manage.*, vol. 16, no. 3, pp. 259–275, Sep. 2008.
[25] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Data and Applications Security XXII* (Lecture Notes in Computer Science), V. Atluri, Ed. Berlin, Germany: Springer, 2008, pp. 283–296.
[26] S. Noel, S. Jajodia, L. Wang, and A. Singhal, "Measuring security risk of networks using attack graphs," *Int. J. Next-Gener. Comput.*, vol. 1, no. 1, pp. 135–147, Jul. 2010.
[27] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 1, pp. 61–74, Jan. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/5936075/
[28] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *Computer Security—ESORICS 2008*, vol. 5283, S. Jajodia and J. Lopez, Eds. Berlin, Germany: Springer, 2008, pp. 18–34. [Online]. Available: http://link.springer.com/10.1007/978-3-540-88313-5_2
[29] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.—Stanford InfoLab Publication Server*. Accessed: Oct. 12, 2022. [Online]. Available: http://ilpubs.stanford.edu:8090/422/
[30] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah, "Distilling critical attack graph surface iteratively through minimum-cost SAT solving," in *Proc. 27th Annu. Comput. Secur. Appl. Conf. (ACSAC)*. New York, NY, USA: Association for Computing Machinery, Dec. 2011, pp. 31–40, doi: 10.1145/2076732.2076738.
[31] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "K-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 1, pp. 30–44, Jan. 2014.
[32] X. Ou, S. Govindavajhala, and A. W. Appel, "MulVAL: A logic-based network security analyzer," in *Proc. 14th USENIX Secur. Symp.*, Baltimore, MD, USA, 2005. Accessed: Jan. 16, 2023. [Online]. Available: http://www.usenix.org/events/sec05/tech/ou.html
[33] H. Holm, M. Ekstedt, T. Sommestad, and L. NordstrM, "CySeMoL: A tool for cyber security analysis of enterprises," in *Proc. 22nd Int. Conf. Exhib. Electr. Distrib. (CIRED)*. London, U.K.: Institution of Engineering and Technology, 2013, p. 1109. [Online]. Available: https://digital-library.theiet.org/content/conferences/10.1049/cp.2013.1077
[34] H. Holm, M. Ekstedt, T. Sommestad, and M. Korman, "A manual for the cyber security modeling language (simplified version)," Dept. Ind. Inf. Control Syst., Roy. Inst. Technol., Tech. Rep., 2013. [Online]. Available: https://www.semanticscholar.org/paper/A-Manual-for-the-Cyber-Security-Modeling-Language-Holm-Ekstedt/ffa6f7eb15f1e91e26d266d5d9f9824cbbf36862
[35] M. L. Artz, "NetSPA: A network security planning architecture," M.S. thesis, Massachusetts Inst. Technol., Cambridge, MA, USA, 2002. [Online]. Available: https://dspace.mit.edu/handle/1721.1/29899
[36] A. Nadeem, S. Verwer, and S. J. Yang, "SAGE: Intrusion alert-driven attack graph extractor," 2021, *arXiv:2107.02783*.
[37] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "AttacKG: Constructing technique knowledge graph from cyber threat intelligence reports," 2021, *arXiv:2111.07093*.
[38] *MITRE ATT&CK*. Accessed: Jan. 26, 2023. [Online]. Available: https://attack.mitre.org/
[39] T. Li, Y. Jiang, C. Lin, M. S. Obaidat, Y. Shen, and J. Ma, "DeepAG: Attack graph construction and threats prediction with bi-directional deep learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 740–757, Jan. 2023.
[40] B. Asvija, R. Eswari, and M. B. Bijoy, "Bayesian attack graphs for platform virtualized infrastructures in clouds," *J. Inf. Secur. Appl.*, vol. 51, Apr. 2020, Art. no. 102455. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212619305332
[41] T. Wang, Q. Lv, B. Hu, and D. Sun, "CVSS-based multi-factor dynamic risk assessment model for network system," in *Proc. IEEE 10th Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC)*, Jul. 2020, pp. 289–294.
[42] D. Malzahn, Z. Birnbaum, and C. Wright-Hamor, "Automated vulnerability testing via executable attack graphs," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services (Cyber Secur.)*, Jun. 2020, pp. 1–10.
[43] H. Shiravi, A. Shiravi, and A. A. Ghorbani, "A survey of visualization systems for network security," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 8, pp. 1313–1329, Aug. 2012.
[44] S.-C. Liu and Y. Liu, "Network security risk assessment method based on HMM and attack graph model," in *Proc. 17th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, May 2016, pp. 517–522.
[45] S. Peryt, J. A. Morales, W. Casey, A. Volkmann, B. Mishra, and Y. Cai, "Visualizing a malware distribution network," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Oct. 2016, pp. 1–4.
[46] A. Ulmer, M. Schufrin, D. Sessler, and J. Kohlhammer, "Visual-interactive identification of anomalous IP-block behavior using geo-IP data," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Oct. 2018, pp. 1–8.
[47] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen, "PortVis: A tool for port-based detection of security events," in *Proc. ACM Workshop Vis. Data Mining Comput. Secur. (VizSEC/DMSEC)*, 2004, p. 73. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1029208.1029220
[48] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju, "VisFlowConnect: Netflow visualizations of link relationships for security situational awareness," in *Proc. ACM Workshop Vis. Data Mining Comput. Secur. (VizSEC/DMSEC)*. New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 26–34, doi: 10.1145/1029208.1029214.
[49] L. Williams, R. Lippmann, and K. Ingols, "An interactive attack graph cascade and reachability display," in *Proc. Workshop Vis. Comput. Secur. (VizSEC)* (Mathematics and Visualization), J. R. Goodall, G. Conti, and K.-L. Ma, Eds. Berlin, Germany: Springer, 2008, pp. 221–236, doi: 10.1007/978-3-540-78243-8_15.
[50] A. Xie, Z. Cai, C. Tang, J. Hu, and Z. Chen, "Evaluating network security with two-layer attack graphs," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2009, pp. 127–136. [Online]. Available: https://ieeexplore.ieee.org/document/5380503
[51] M. Chu, K. Ingols, R. Lippmann, S. Webster, and S. Boyer, "Visualizing attack graphs, reachability, and trust relationships with NAVIGATOR," in *Proc. 7th Int. Symp. Vis. Cyber Secur. (VizSec)*. New York, NY, USA: Association for Computing Machinery, Sep. 2010, pp. 22–33, doi: 10.1145/1850795.1850798.
[52] M. Angelini, N. Prigent, and G. Santucci, "PERCIVAL: Proactive and reactive attack and response assessment for cyber incidents using visual analytics," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Oct. 2015, pp. 1–8.
[53] S. Noel, E. Harley, K. Tam, M. Limiero, and M. Share, "CyGraph: Graph-based analytics and visualization for cybersecurity," in *Handbook of Statistics*, vol. 35. Amsterdam, The Netherlands: Elsevier, 2016, pp. 117–167. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0169716116300426
[54] S. O'Hare, S. Noel, and K. Prole, "A graph-theoretic visualization approach to network risk analysis," in *Visualization for Computer Security*, vol. 5210, J. R. Goodall, G. Conti, and K.-L. Ma, Eds. Berlin, Germany: Springer, 2008, pp. 60–67. [Online]. Available: http://link.springer.com/10.1007/978-3-540-85933-8_6
[55] M. Angelini, S. Bonomi, S. Lenti, G. Santucci, and S. Taggi, "MAD: A visual analytics solution for multi-step cyber attacks detection," *J. Comput. Lang.*, vol. 52, pp. 10–24, Jun. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1045926X17301106
[56] L. Leichtnam, É. Totel, N. Prigent, and L. Mé, "STARLORD: Linked security data exploration in a 3D graph," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Oct. 2017, pp. 1–4.
[57] O. Tsigkas, O. Thonnard, and D. Tzovaras, "Visual spam campaigns analysis using abstract graphs representation," in *Proc. 9th Int. Symp. Vis. Cyber Secur. (VizSec)*. New York, NY, USA: Association for Computing Machinery, Oct. 2012, pp. 64–71, doi: 10.1145/2379690.2379699.
[58] J. J. Fowler, T. Johnson, P. Simonetto, M. Schneider, C. Acedo, S. Kobourov, and L. Lazos, "IMap: Visualizing network activity over Internet maps," in *Proc. 11th Workshop Vis. Cyber Secur. (VizSec)*. New York, NY, USA: Association for Computing Machinery, Nov. 2014, pp. 80–87, doi: 10.1145/2671491.2671501.

[59] G. Ikuomenisan and Y. Morgan, "Systematic review of graphical visual methods in honeypot attack data analysis," *J. Inf. Secur.*, vol. 13, no. 4, pp. 210–243, 2022. [Online]. Available: https://www.scirp.org/journal/doi.aspx?doi=10.4236/jis.2022.134012

[60] J. Bertin, *Semiology of Graphics: Diagrams, Networks, Maps*. ESRI Press, 2011.

[61] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychol. Rev.*, vol. 63, no. 2, pp. 81–97, 1956.

[62] M. Petre, "Why looking isn't always seeing: Readership skills and graphical programming," *Commun. ACM*, vol. 38, no. 6, pp. 33–44, 1995. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/203241.203251

[63] D. L. Moody, "The 'physics' of notations: A scientific approach to designing visual notations in software engineering," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. (ICSE)*. New York, NY, USA: Association for Computing Machinery, vol. 2, May 2010, pp. 485–486, doi: 10.1145/1810295.1810442.

[64] L. Graham, "Gestalt theory in interactive media design," *J. Humanities Social Sci.*, no. 2, 2008.

[65] H. Kobayashi, Z. Zhang, H. Ochiai, and H. Esaki, "Probing firewalls of malware-infected networks with honeypot," in *Proc. 14th Int. Conf. Future Internet Technol.*, Aug. 2019, pp. 1–4. [Online]. Available: https://dl.acm.org/doi/10.1145/3341188.3341190

[66] E. Vasilomanolakis, S. Karuppayah, P. Kikiras, and M. Mühlhäuser, "A honeypot-driven cyber incident monitor: Lessons learned and steps ahead," in *Proc. 8th Int. Conf. Secur. Inf. Netw.*, 2015, pp. 158–164. [Online]. Available: https://dl.acm.org/doi/10.1145/2799979.2799999

[67] M. Dumas, J.-M. Robert, and M. J. Mcguffin, "Alertwheel: Radial bipartite graph visualization applied to intrusion detection system alerts," *IEEE Netw.*, vol. 26, no. 6, pp. 12–18, Nov. 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6375888/

[68] M. Aupetit, Y. Zhauniarovich, G. Vasiliadis, M. Dacier, and Y. Boshmaf, "Visualization of actionable knowledge to mitigate DRDoS attacks," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Oct. 2016, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/7739577/

[69] *AlienVault—Open Threat Exchange*. Accessed: Jan. 26, 2023. [Online]. Available: https://otx.alienvault.com/

[70] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "MISP: The design and implementation of a collaborative threat intelligence sharing platform," in *Proc. ACM Workshop Inf. Sharing Collaborative Secur.*, Oct. 2016, pp. 49–56. [Online]. Available: https://dl.acm.org/doi/10.1145/2994539.2994542

[71] *Common Vulnerabilities and Exposures*. Accessed: Jan. 26, 2023. [Online]. Available: https://www.cve.org/

[72] *National Vulnerability Database*. Accessed: Jan. 26, 2023. [Online]. Available: https://nvd.nist.gov/

[73] *Offensive Security's Exploit Database Archive*. Accessed: Jan. 26, 2023. [Online]. Available: https://www.exploit-db.com/

[74] *Introduction to STIX*. Accessed: Jan. 26, 2023. [Online]. Available: https://oasis-open.github.io/cti-documentation/stix/intro

[75] *Introduction to TAXII*. Accessed: Jan. 26, 2023. [Online]. Available: https://oasis-open.github.io/cti-documentation/taxii/intro.html

[76] *CybOX—Cyber Observable Expression*. Accessed: Jan. 26, 2023. [Online]. Available: https://cyboxproject.github.io/

[77] *About MAEC | MAEC Project Documentation*. Accessed: Jan. 26, 2023. [Online]. Available: https://maecproject.github.io/about-maec/

[78] VirusTotal. *VirusTotal/Yara*. Accessed: Jan. 26, 2023. [Online]. Available: https://github.com/VirusTotal/yara

[79] A. Kyriakou and N. Sklavos, "Container-based honeypot deployment for the analysis of malicious activity," in *Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS)*, Oct. 2018, pp. 1–4.

[80] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," 2016, *arXiv:1608.06249*.

[81] *IoT Cybersecurity: Research Challenges and Opportunities Ahead—IEEE Internet of Things*. Accessed: Jun. 7, 2022. [Online]. Available: https://iot.ieee.org/newsletter/may-2020/iot-cybersecurity-research-challenges-and-opportunities-ahead.html

[82] A. Bar, B. Shapira, L. Rokach, and M. Unger, "Identifying attack propagation patterns in honeypots using Markov chains modeling and complex networks analysis," in *Proc. IEEE Int. Conf. Softw. Sci., Technol. Eng. (SWSTE)*, Jun. 2016, pp. 28–36. [Online]. Available: http://ieeexplore.ieee.org/document/7515407/

[83] H. Studiawan, S. Djanali, and B. Pratomo, "Graph-based forensic analysis of web honeypot," *J. Telecommun. Inf. Technol.*, vol. 2016, no. 2, pp. 60–65, 2016.

[84] G. Fernandez, A. Nieto, and J. Lopez, "Modeling malware-driven honeypots," in *Proc. 14th Int. Conf. Trust, Privacy Secur. Digit. Bus. (TrustBus)*, 2017, pp. 130–144.

[85] G. Wagener, "Self-adaptive honeypots coercing and assessing attacker behaviour," Ph.D. dissertation, Univ. Luxembourg, Esch-sur-Alzette, Luxembourg, 2011.

[86] K. Durkota, V. Lisý, C. Kiekintveld, B. Bošanský, and M. Pechoucek, "Case studies of network defense with attack graph games," *IEEE Intell. Syst.*, vol. 31, no. 5, pp. 24–30, Sep. 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7579429/

[87] A. H. Anwar, C. Kamhoua, and N. Leslie, "Honeypot allocation over attack graphs in cyber deception games," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2020, pp. 502–506.

[88] Y. Gao, G. Zhang, and C. Xing, "A multiphase dynamic deployment mechanism of virtualized honeypots based on intelligent attack path prediction," *Secur. Commun. Netw.*, vol. 2021, Oct. 2021, Art. no. e6378218. [Online]. Available: https://www.hindawi.com/journals/scn/2021/6378218/

[89] M. Kaâniche, Y. Deswarte, E. Alata, M. Dacier, and V. Nicomette, "Empirical analysis and statistical modeling of attack processes based on honeypots," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN), Workshop Empirical Eval. Dependability Secur. (WEEDS)*, France, Jun. 2006, pp. 119–124.

[90] S. Noel, "A review of graph approaches to network security analytics," in *From Database to Cyber Security*, P. Samarati, I. Ray, and I. Ray, Eds. Cham, Switzerland: Springer, vol. 11170, 2018, pp. 300–323. [Online]. Available: http://link.springer.com/10.1007/978-3-030-04834-1_16

[91] *Neo4j Graph Data Platform—The Leader in Graph Databases. Neo4j Graph Data Platform*. Accessed: Jun. 15, 2022. [Online]. Available: https://neo4j.com/

[92] *openCypher*. Accessed: Jun. 15, 2022. [Online]. Available: https://opencypher.org/

[93] J. Yang, R. Xu, Z. Qi, and Y. Shi, "Visual anomaly detection for images: A survey," 2021, *arXiv:2109.13157*.

[94] S.-Y. Ji, B.-K. Jeong, and D. H. Jeong, "Evaluating visualization approaches to detect abnormal activities in network traffic data," *Int. J. Inf. Secur.*, vol. 20, no. 3, pp. 331–345, Jun. 2021, doi: 10.1007/s10207-020-00504-9.

[95] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," 2014, *arXiv:1403.2877*.

[96] T. L. Weissgerber, N. M. Milic, S. J. Winham, and V. D. Garovic, "Beyond bar and line graphs: Time for a new data presentation paradigm," *PLOS Biol.*, vol. 13, no. 4, Apr. 2015, Art. no. e1002128. [Online]. Available: https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1002128

[97] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual analytics: Definition, process, and challenges," in *Information Visualization: Human-Centered Issues and Perspectives* (Lecture Notes in Computer Science), A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, Eds. Berlin, Germany: Springer, 2008, pp. 154–175, doi: 10.1007/978-3-540-70956-5_7.

[98] T. Green, "Cognitive dimensions of notations," in *People and Computers*. Cambridge, U.K.: Cambridge Univ. Press, 1989, pp. 443–460.

[99] A. H. S. Chan and A. W. Y. Ng, "Perceptions of implied hazard for visual and auditory alerting signals," *Saf. Sci.*, vol. 47, no. 3, pp. 346–352, Mar. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925753508000908

[100] O. Omojola, "Using symbols and shapes for analysis in small focus group research," *Qualitative Rep.*, vol. 21, pp. 832–847, May 2016.

[101] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proc. Int. AAAI Conf. Web Social Media*, 2009.

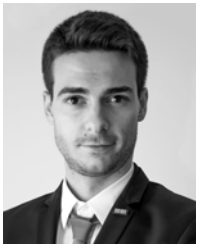
[102] M. Franz, C. T. Lopes, G. Huck, Y. Dong, O. Sumer, and G. D. Bader, "Cytoscape.js: A graph theory library for visualisation and analysis," *Bioinformatics*, vol. 32, no. 2, pp. 309–311, 2016. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv557

[103] *Graphology*. Accessed: Apr. 2, 2023. [Online]. Available: https://graphology.github.io/

[104] *Sigma.js*. Accessed: Apr. 2, 2023. [Online]. Available: https://www.sigmajs.org/

[105] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.

[106] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, 2008, Art. no. P10008.



MATEJ RABZELJ received the B.Sc. degree in electronics engineering and the M.Sc. degree in information and communications technology from the Faculty of Electrical Engineering, University of Ljubljana, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree. He is a Research Associate with the Faculty of Electrical Engineering, University of Ljubljana. His research interests include cybersecurity, systems design, and PPDR solution development. In 2022, he participated as a National Cyberdefense Team Member in NATO's Locked Shields Exercise.

CIRIL BOHAK received the B.Sc., M.Sc., and Ph.D. degrees in computer and information science from the University of Ljubljana. He is a Research Scientist with the Nanovisualization Research Group, Visual Computing Center, King Abdullah University of Science and Technology (KAUST). He is also an Assistant Professor with the Faculty of Computer and Information Science, University of Ljubljana. He works on the development of novel rendering and visualization methods for visualizing data at the nanoscale. He started his career as a Researcher with the Laboratory for Computer Graphics and Multimedia, researching in the fields of music information retrieval, computer graphics, human–computer interaction, game technology, and e-learning. He continued his career as a full-time Teaching Assistant with the Faculty of Computer and Information Science, University of Ljubljana, teaching diverse CS-related courses. His current scientific focus is on volumetric and neural rendering and data visualization. He is developing and applying new rendering and visualization methods at the nanoscale for interactive and intuitive structure visualization.

LEON ŠTEFANIČ JUŽNIČ received the B.Sc. and M.Sc. degrees in telecommunications from the Faculty of Electrical Engineering, University of Ljubljana. He is a Research Associate with the Laboratory for Telecommunications, Faculty of Electrical Engineering, University of Ljubljana. His main research interests include cybersecurity cloud architectures and data analysis.

ANDREJ KOS (Senior Member, IEEE) received the Ph.D. degree in telecommunications from the University of Ljubljana. He is a Full Professor with the Faculty of Electrical Engineering, University of Ljubljana. He is also the Head of the Laboratory for Telecommunications and the Chairman of the Commission for Innovation. Currently, at the center of his work are 5G/6G systems and services and the applications of cyber-physical systems, including the Internet of Things.

URBAN SEDLAR (Member, IEEE) is an Assistant Professor and a Senior Researcher with the Laboratory for Telecommunications, Faculty of Electrical Engineering, University of Ljubljana. His current work focuses on the area of cybersecurity threat assessment using large-scale honeypots. He has been involved in several EC and national research and development projects on the topics of emergency response systems, cloud computing, and the Internet of Things.